Machine Learning for Complex Networks

Prof. Dr. Ingo Scholtes

Chair of Machine Learning for Complex Networks
Center for Artificial Intelligence and Data Science (CAIDAS)
Julius-Maximilians-Universität Würzburg
Würzburg, Germany

ingo.scholtes@uni-wuerzburg.de

Lecture 01
Motivation

April 17, 2024

Notes:

- **Lecture L01: Motivation** 17.04.2024
- **Educational objective:** We introduce our Chair and motivate challenges and opportunities of machine learning in data that exhibit a graph structure. We provide an overview of topics covered in the course and discuss organizational issues.
  - Introducing the Chair of Informatics XV
  - Machine Learning for Graph-Structured Data
  - Interdisciplinary Applications of Graph Learning
  - Course Overview and Organizational Issues

Chair of Informatics XV

Franziska Heeg[1]

Lisi Qarkaxhija[1]

Chester Tan[1]

Dr. Christopher Blöcker[1]

Moritz Lampert [1]

Dr. Vincenzo Perri[1]

Dr. Anatol Wegner[1]

Prof. Dr. Ingo Scholtes[1,2]

[1] *Chair of Informatics XV, CAIDAS, University of Würzburg* / [2] *Data Analytics Group, Department of Informatics, Universität Zürich*

Notes:

- The Chair of Machine Learning for Complex Networks is an international and interdisciplinary team, currently residing at two physical locations. Seven group members are employees at Julius-Maximilians-Universität Würzburg. One additional member of our team is funded by a third-party project still running at the Department of Informatics at the University of Zurich.
- The nationalities of our team include German, Swedish, Italian, Turkish, Kosovarian, and Singaporian. Our team incorporates backgrounds in computer science, data science, civil engineering, control engineering, psychology, mathematics and theoretical physics. We appreciate the different view points that results from this diversity. However, all members of our team speak a common “scientific language”, which is the language of graph theory, machine learning, statistics, and mathematical modelling.

Research interests

Notes:

- Our research profile includes methodological and applied works in data science, network science and machine learning.
- In a first line of works we address the theoretical and methodological foundations of network-based modelling of complex systems and machine learning for complex networks. We are interested in open issues that arise in large, high-dimensional, noisy, and time-resolved data on complex systems and we develop machine learning and statistical modelling techniques to address them.
- In a second line of works, we apply data science, network analysis, and machine learning in domains like, e.g., software engineering, computational social science, scientometrics, or information systems. We specifically use data science and graph learning to better understand teams, companies, and other social organizations based on large-scale behavioral data. We recently started projects in which we combine network science and machine learning to support decision-making in areas like medical diagnostics, educational studies, or logistics.
- We publish our works in international journals addressing natural sciences, (statistical) physics, and complex systems, as well as in leading international conferences on machine learning, data mining, graph learning, network analysis, software engineering, computational social science, and information systems.

Teaching portfolio

Lectures

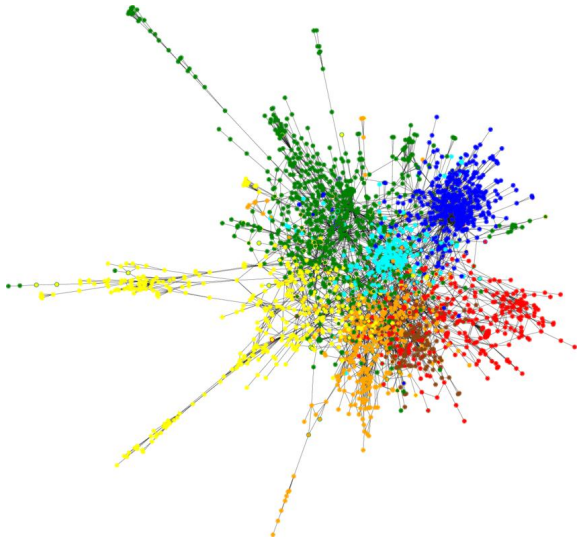
- MSc lecture **Statistical Network Analysis**
  → winter semester
- MSc lecture **Machine Learning for Complex Networks**
  → summer semester
- MSc lecture **Introduction to CS for Jurists**
  → winter semester
- BSc lecture **Algorithms, AI, and Data Science II**
  → summer semester

MSc-Praktika (Labs)

- Statistical Network Analysis
  → summer semester
- Graph Neural Networks
  → summer semester
- eXtAI Lab
  → each semester

BSc/MSc Seminars

- Data, AI, and Society
  → summer semester
- Machine Learning for Complex Networks
  → summer semester
- Computational Astrophotography
  → winter semester



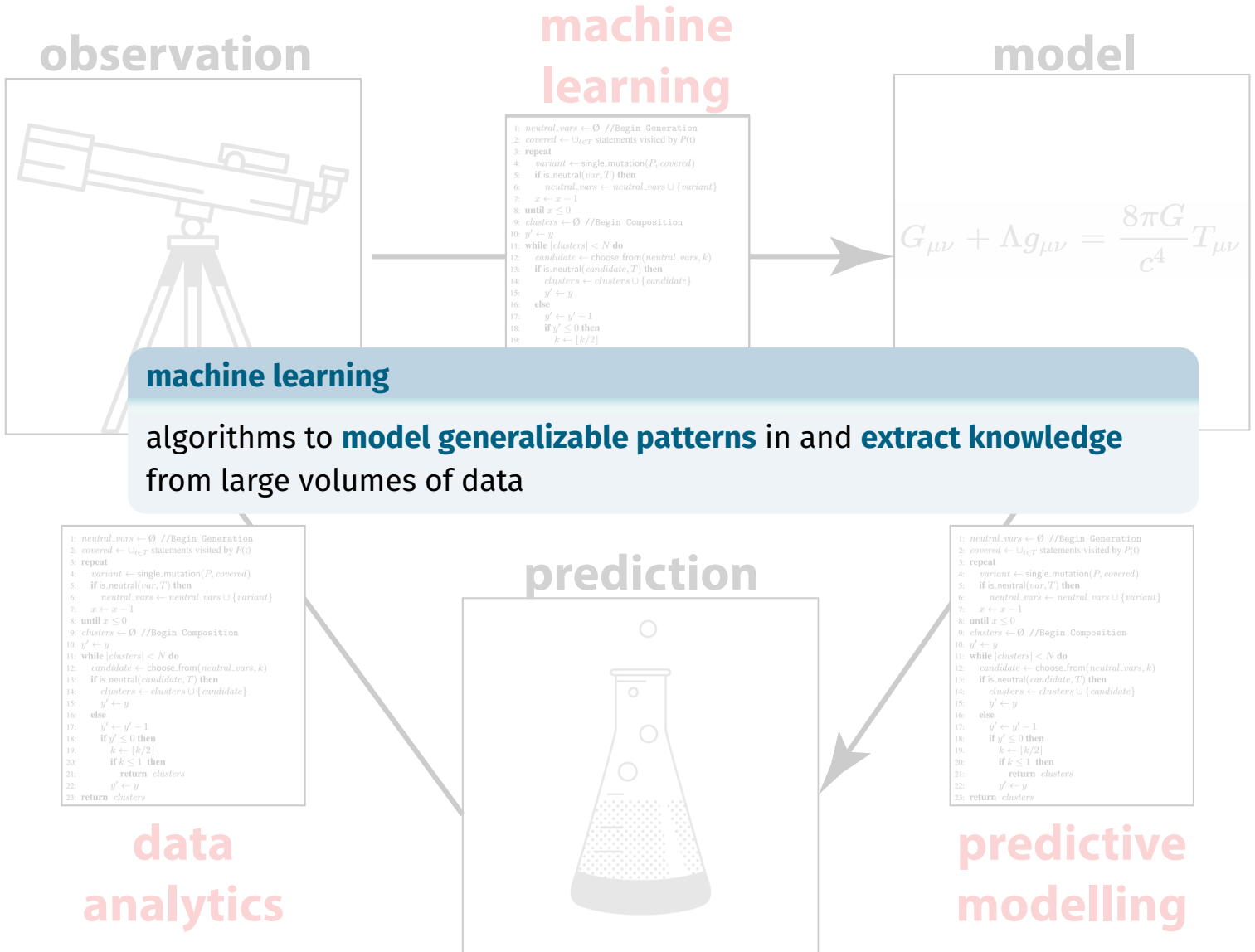
PATHPY

PyTorch

Notes:

- Above, you can find an overview of the lectures, seminars, and labs that we currently offer in the BSc and MSc programs *Informatics* and *eXtended AI* at JMU.
- In this semester we offer several seminars and labs. You can find more information in the “Vorlesungsverzeichnis”, on our website and on WueCampus.

What is machine learning?



machine learning

algorithms to **model generalizable patterns** in and **extract knowledge** from large volumes of data

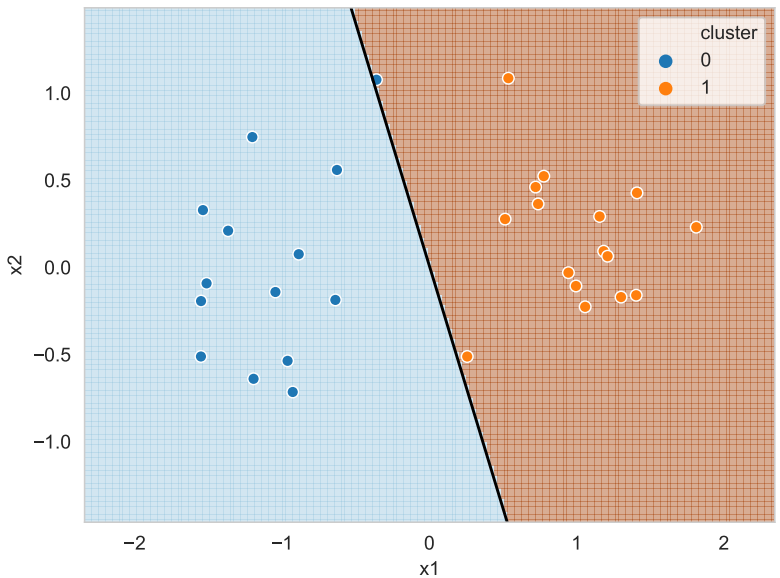
Notes:

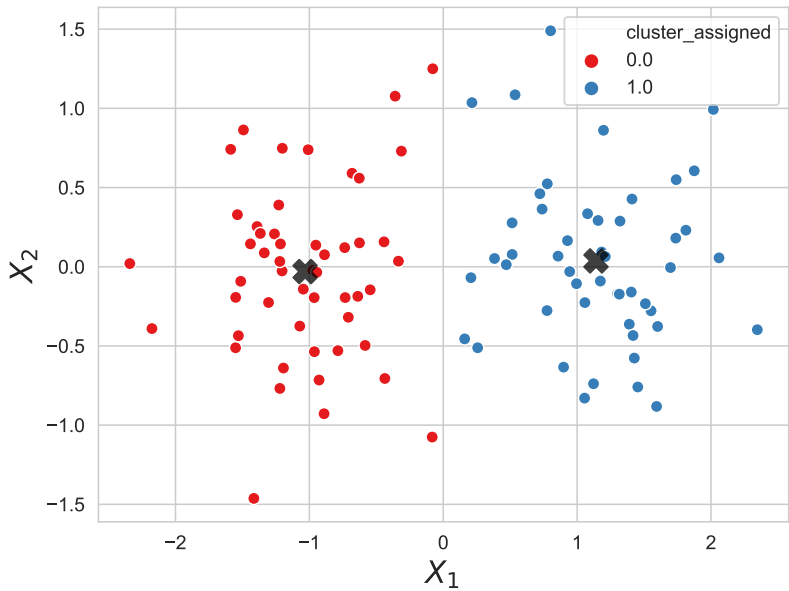
- Before we can understand what “machine learning” is, we first need to understand what “learning” is. Learning refers to the **systematic process by which we extract knowledge from experience** (cf. Greek “empeiría” for experience). This experience is based on the observation of a phenomenon in our environment. Classical examples are astronomical observations (e.g. of stars, planets, or galaxies).
- We are interested in a model that describes (or explains, from Latin “explano” for “make plain” or “make clear”) our observations. A **good model provides a compact summary of our observations and allows to derive predictions** that can be used to falsify our model in future observations.
- We call this systematic process the “scientific method” and digital technologies have revolutionized it, e.g. by improving our ability to derive predictions through large-scale simulations or test predictions in vast amounts of data. However, the task of deriving a good model that can be used to make predictions is largely left to human intuition and creativity.
- Human cognitive abilities are limited, which is why we are interested in **machine learning** techniques that can be used to further automate knowledge generation. Fitting this view, the term “machine learning” refers to methods, processes, and algorithms that can be used to extract knowledge from large volumes of structured and unstructured data.

Supervised and unsupervised learning

learn model in **labeled examples**

detect patterns in **unlabeled data**





example method: support vector machine (SVM)

find d — 1-dim. hyperplane that separates classes such that margin of decision boundary is maximized

→ BSc/MSc Lecture: Data Mining

example method: k-means clustering

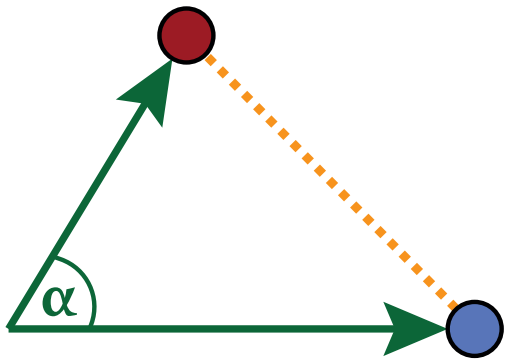
assign data points to k clusters, such that squared distance of points to closest cluster center is minimized → Bsc/MSc Lecture: Data Mining

Notes:

- As you probably know from an introductory machine learning course, we can broadly categorize machine learning into **supervised and unsupervised techniques**. We ignore **reinforcement learning techniques**, which will not be discussed in our course.
- In **supervised learning** techniques, we assume that we have access to a continuous or categorical response or target variable that we can use for training.
- What if we do not have access to labelled examples that we could use to train a classifier? We can still detect patterns, e.g. in the distribution of points in our feature space. In particular, we can identify groups of objects that are close together in the feature space. This is an example for a clustering problem, an important class of problems in **unsupervised learning**.
- We will later also introduce semi-supervised techniques where a response or target variable is only available for small subset of the data, i.e. those combine aspects of supervised and unsupervised methods.
- We illustrate those two classes of machine learning techniques with two popular algorithms for classification (supervised) and clustering (unsupervised): **Support vector machines** find a (linear or non-linear) decision boundary that separates data points in different classes. The **k-means clustering algorithm** assigns data points to a given number of k clusters such that the squared distances of points to the closest cluster center is minimized.

Machine Learning in Euclidean data

- many machine learning techniques assume **Euclidean feature spaces**, e.g. $x_i \in \mathbb{R}^d$
- d-dimensional Euclidean space is **metric space** with **Euclidean distance** metric
- Euclidean vector space = d-dimensional **inner product space** over $\mathbb{R}$



$$\|\vec{y} - \vec{x}\| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

$$\vec{x} \cdot \vec{y} = x_1 y_1 + x_1 x_2 = \|\vec{x}\| \|\vec{y}\| \cos \alpha$$

Euclid of Alexandria as depicted in the fresco “The School of Athens”
born ca. 325 BC

image credit: public domain

Notes:

- You are probably familiar with basic machine learning algorithms like SVM, logistic regression, k-nearest-neighbor classification, or k-means clustering. Such general-purpose techniques often assume a Euclidean feature space, e.g. d-dimensional feature vectors $\mathbb{R}^d$. Such a Euclidean space has special properties that are used by many machine learning techniques.
- A d-dimensional Euclidean space with the Euclidean distance metric

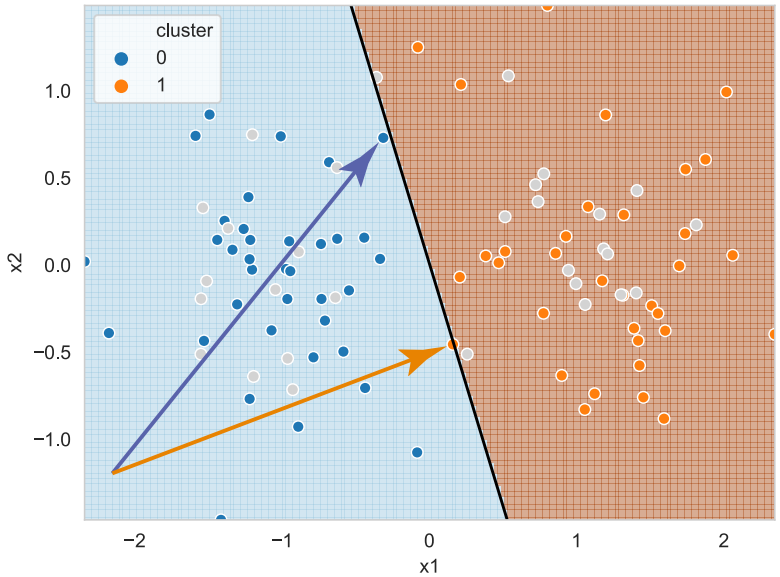
$$dist(\vec{x}, \vec{y}) := \|\vec{y} - \vec{x}\| = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

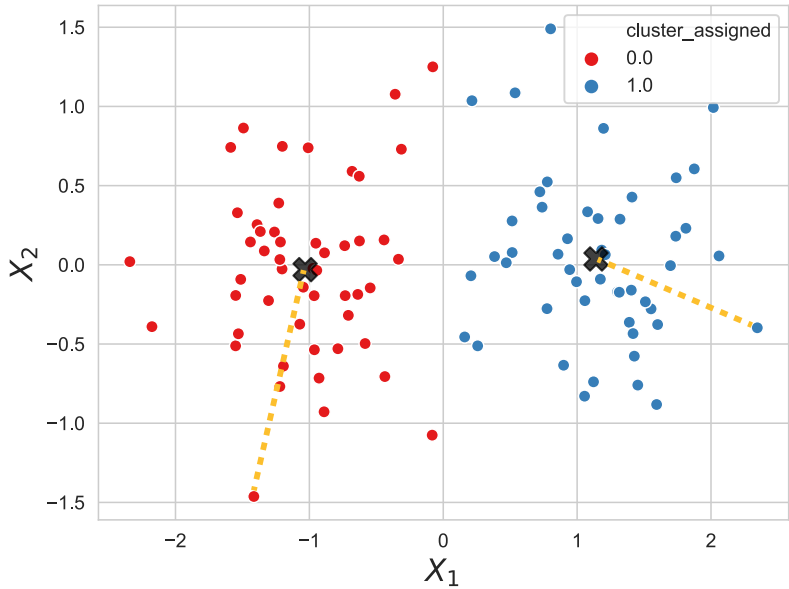
  is a prototypical example of a **metric space**, where the Euclidean distance fulfills the three axioms of a metric (identity of indiscernibles, symmetric, triangle inequality). Many machine learning techniques that calculate distances between data points implicitly use the resulting properties of a metric space.
- Using the dot product

$$\vec{x} \cdot \vec{y} = x_1 y_1 + x_1 x_2 = \|\vec{x}\| \|\vec{y}\| \cos \alpha$$

  the Euclidean space fulfills the properties of an **inner product space**. The dot product can be viewed as a measure of "similarity" between data points, where the similarity depends both on the length of those vectors and the angle α between them.

Machine Learning in Euclidean data





example method: suppport vector machine (SVM)

find $d - 1$-dim. hyperplane that separates classes such that margin of decision boundary is maximized

→ BSc/MSc Lecture: Data Mining

example method: k-means clustering

assign data points to k clusters, such that squared distance of points to closest cluster center is minimized → Bsc/MSc Lecture: Data Mining

dot product between $x_i \in \mathbb{R}^2$, i.e. we use property of **inner product space**

distance between $x_i \in \mathbb{R}^2$, i.e. we use property of **metric space**

Notes:

- Let us have a look at the two exemplary applications of machine learning from before.
- k-means clustering seeks to minimize the squared distance between the cluster centers and the data points assigned to the closest cluster center. Due to the use of a distance measure, this algorithm implicitly assume that the feature space fulfills **metric** properties.
- Other machine learning techniques are based on “similarities” between feature vectors that can be calculated based on a dot product between vectors (e.g. in Euclidean space this yields the cosine of the angle between vectors, which is zero if they are orthogonal). This dot product is directly used by a support vector machine (which is named after support vectors that are orthogonal to the decision boundary). With this, we use the fact that our feature space is an **inner product space**
- Note that the property of an inner product space is more general than a metric space. We can use vector norms to obtain a metric space, but we can define inner product spaces that are non-Euclidean. Consider, for instance support vector machines. Using the kernel trick, we can define an inner product that implicitly maps features into a (high-dimensional) inner product space. Consider, e.g., the radial basis function kernel that helps us to fit linear decision boundaries in a high-dimensional inner product space, which correspond to a complex non-linear decision boundaries in our original Euclidean space.

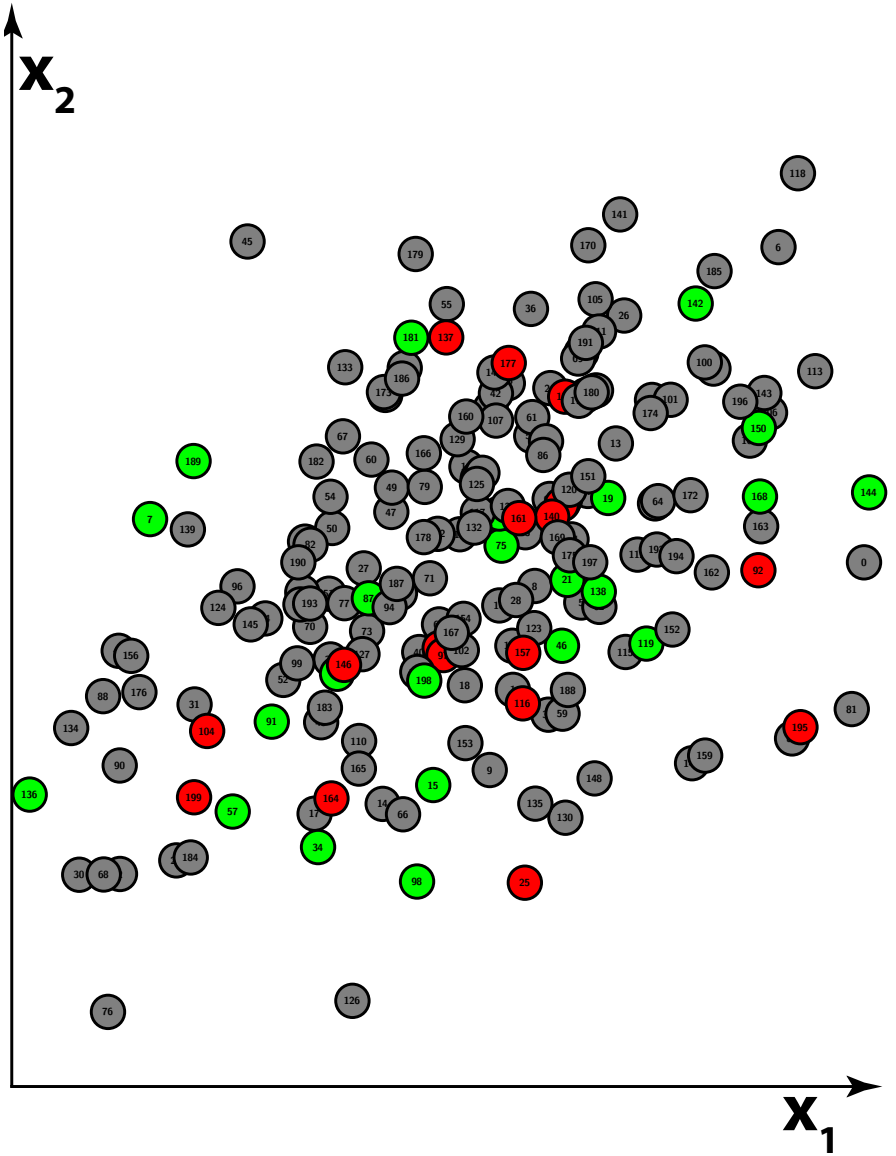
A more complex example

- consider synthetic data $x_i \in \mathbb{R}^2$ with classes $y_i \in \mathbb{C} = \{0, 1\}$
- split data in **training and test set**
- train **support vector machine** with **non-linear RBF kernel**

example: kernel SVM

- classification accuracy ≈ 0.57
- kernel SVM in Euclidean feature space unable to classify data

- problem: data often exhibits **complex relationships** not captured by Euclidean features



correctly and wrongly classified data points using **SVM with radial kernel** (accuracy = 0.57)

Notes:

- Despite the remarkable successes of machine learning in recent years, there are application domains where general purpose machine learning techniques reach their limits. This is due to the fact that most standard machine learning techniques assume data in a Euclidean domain. For many real-world data sets, this is not sufficient to capture the complex patterns that enable us to address supervised or unsupervised learning tasks.
- To give a concrete example, consider the synthetic data set shown on the right, which contains two (Euclidean) features X_1 and X_2 and binary classes 0 and 1.
- Those classes are clearly not separable by a simple linear model, which is why we adopt a kernel SVM with a non-linear radial basis function kernel. We train the SVM in a training set and then apply it to a test set. We find that the classification accuracy in this example is very low, approx. 0.57. Since we have balanced classes, this is not much better than a random guess.
- You could now think that this bad performance is simply due to the absence of a pattern that can be learned by the model. Such a pattern exists in this data set, but we cannot capture this in a simple two-dimensional Euclidean space.
- The point is that the observations in this data set, like many other real-world data, exhibits complex relationships that are not captured by the Euclidean features.

Learning in graph-structured data?

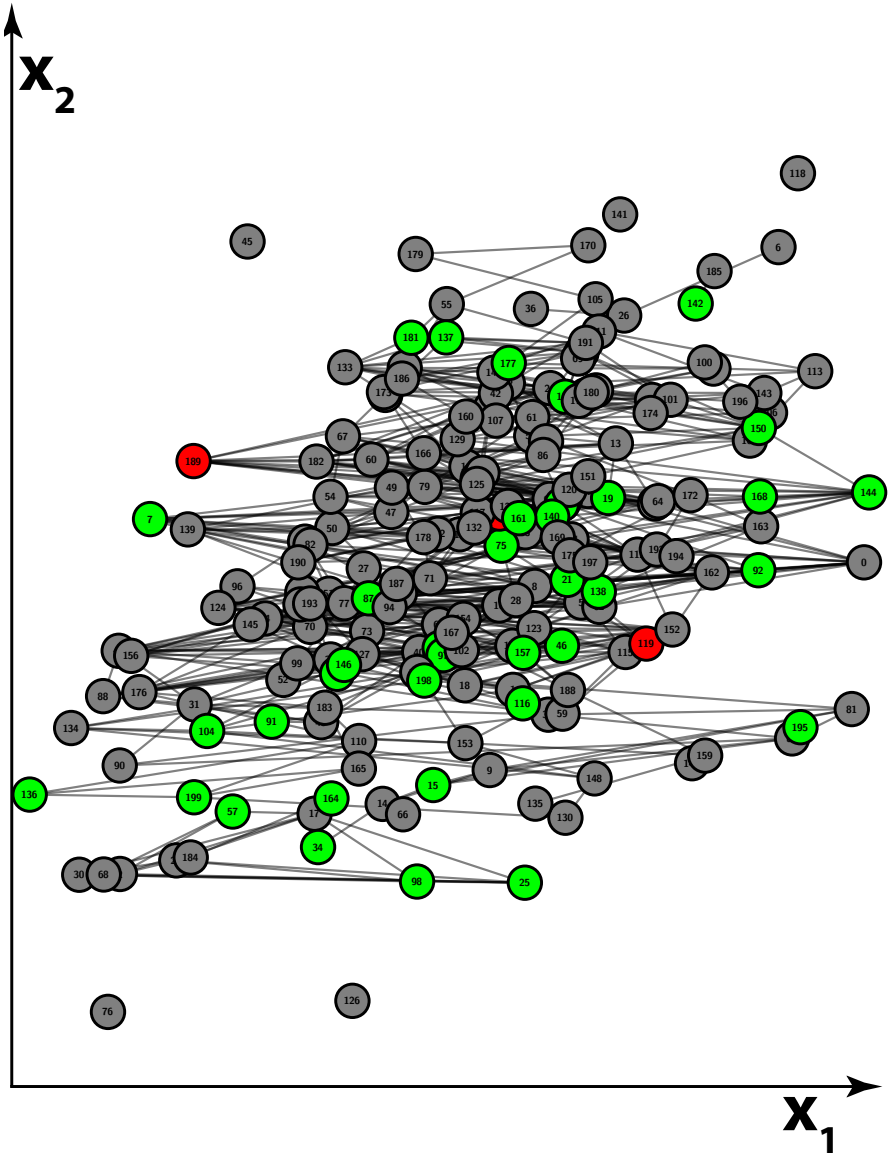
definition

- **graph or network** $G = (V, E)$ consists of a set of nodes (vertices) V and links (edges) $E \subseteq V \times V$
- nodes can have features/classes, e.g. $x : V \to \mathbb{R}^d$ and $y : V \to \mathbb{C}$

- how can we apply machine learning to such **graph-structured data**?

exemplary approaches

1. apply statistical learning to **ensembles of networks** → statistical network analysis
2. map graph to **vectorial feature space**
   → graph embedding techniques
3. machine learning for graphs with **additional** vectorial features
   → graph neural networks



correctly and wrongly classified nodes using **graph neural network** (accuracy = 0.93)

Notes:

- We often have additional information on relationships that can not be represented in a (low-dimensional) Euclidean space. In our example, additional relationships exist between data points, which can be represented as links in a graph.
- These relationships contain a pattern that we can use for machine learning. To highlight those patterns, we can assign positions to nodes based on a **force-directed layout algorithm** (rather than assigning positions based on Euclidean features of nodes). This shows that the graph topology constitutes an independent dimension of information on our problem.
- Note that graphs are one of the most expressive data structures. Interestingly, most machine learning problems can be viewed as special cases of graph-structured problems (e.g. features in a Euclidean space, image data) but not the other way round. There are different approaches to leverage this (additional) graph structure:
- First, we can model patterns based on **statistical models for graphs** like those discussed in our course *Statistical Network Analysis*. This approach yields statistical learning techniques that we discuss in a first chapter of the course.
- Second, we could embed the nodes in the graph into a (Euclidean) feature space and then apply standard machine learning algorithms. This embedding should preserve the information/patterns contained in the topology. We will discuss such **graph embedding** techniques in a second chapter.
- A third approach uses new techniques that leverage both (vectorial) features of nodes/links as well as the graph topology. **Graph neural networks** are one important way to achieve this, which we will discuss in a final chapter of the course.

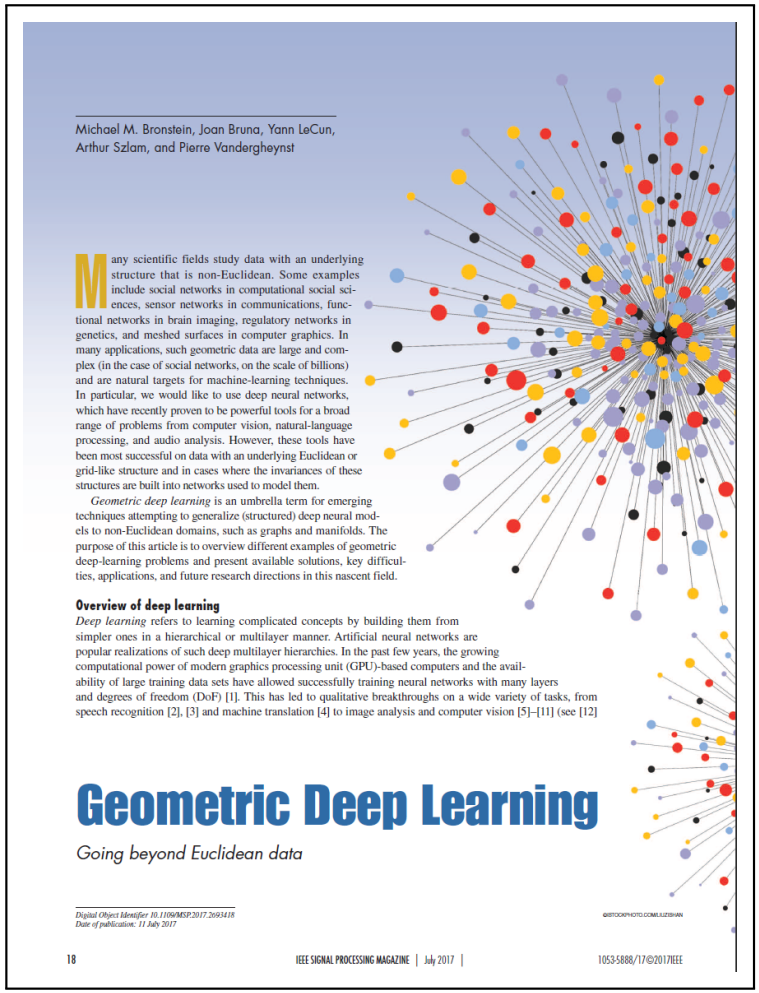
Geometric Machine Learning

- graph learning = special class of **geometric machine learning techniques** for non-Euclidean domains

> "Geometric deep learning is an umbrella term for emerging techniques attempting to generalize (structured) deep neural network models to non-Euclidean domains, such as graphs and manifolds." → Michael M Bronstein et al., 2017

geometric learning domains

- 3D molecular structures
- two-dimensional image/video data
- tree structures (e.g. knowledge graphs/phylogenetics)
- non-Euclidean manifolds (e.g. surface of 3D shape)



Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst

Many scientific fields study data with an underlying structure that is non-Euclidean. Some examples include social networks in computational social sciences, sensor networks in communications, functional networks in brain imaging, regulatory networks in genetics, and meshed surfaces in computer graphics. In many applications, such geometric data are large and complex (in the case of social networks, on the scale of billions) and are natural targets for machine-learning techniques. In particular, we would like to use deep neural networks, which have recently proven to be powerful tools for a broad range of problems from computer vision, natural-language processing, and audio analysis. However, these tools have been most successful on data with an underlying Euclidean or grid-like structure and in cases where the invariances of these structures are built into networks used to model them.

Geometric deep learning is an umbrella term for emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains, such as graphs and manifolds. The purpose of this article is to overview different examples of geometric deep-learning problems and present available solutions, key difficulties, applications, and future research directions in this nascent field.

Overview of deep learning

Deep learning refers to learning complicated concepts by building them from simpler ones in a hierarchical or multilayer manner. Artificial neural networks are popular realizations of such deep multilayer hierarchies. In the past few years, the growing computational power of modern graphics processing unit (GPU)-based computers and the availability of large training data sets have allowed successfully training neural networks with many layers and degrees of freedom (DoF) [1]. This has led to qualitative breakthroughs on a wide variety of tasks, from speech recognition [2], [3] and machine translation [4] to image analysis and computer vision [5]–[11] (see [12]

Geometric Deep Learning

Going beyond Euclidean data

Digital Object Identifier 10.1109/MSP.2017.2693418
Date of publication: 11 July 2017

 1053-5888/17©2017IEEE

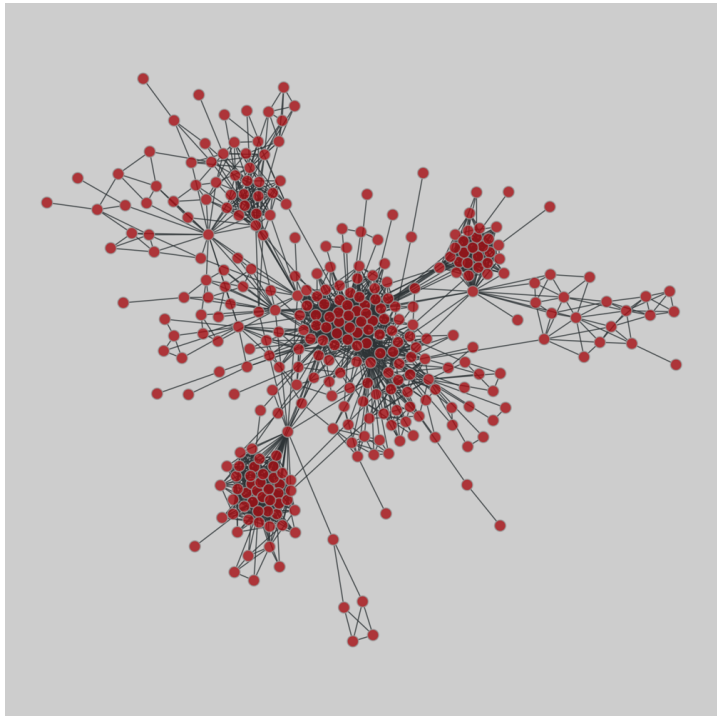
→ MM Bronstein et al. IEEE Sig. Proc. Mag., 2017

Notes:

- Machine learning techniques for graph-structured data are a special class of **geometric machine learning techniques** for non-Euclidean domains, i.e. for data that exhibits structures that cannot be represented in a simple vector space.
- Data with graph structures are only one example for non-Euclidean data. Other examples include three-dimensional molecular structures, where atoms can exhibit complex properties, bindings, and relationships based on the laws of physics. Network models of molecular are one increasingly popular approach to address machine learning problems in such data.
- Image or video data are another example for data that require specialized machine learning methods that account for the special structure of pixel data arranged in a lattice. Note that we can again think of an image as a lattice network of pixels, where adjacent pixel coordinates (e.g. at Hamming distance one) are connected by an edge.
- We also often have data that can naturally be arranged in a hierarchy or tree structure. Consider, e.g. certain types of knowledge graphs or trees formed by genetic inheritance in phylogenetics. We note that such trees are naturally represented as a special type of graphs that exhibit not cycles.
- Finally, an important example for non-Euclidean data are all sorts of manifolds, e.g. surfaces of higher-dimensional objects. Such manifolds locally resemble a Euclidean space, but have a more complex topology. Consider, e.g. a machine learning problem that can be mapped to the surface of the 3D rabbit above. Interestingly, we can again map such problems to a graph topology, which underpins the fact that they provide us with a very general approach to study problems in non-Euclidean domains.

Example 1: Social Networks

consider an **online social network**, where nodes represent **users** and links represent **friend/follower relationships**



question

can we automatically recommend interesting accounts to (new) users?

graph learning task

we can cast this as an **link prediction** problem

→ week 07/08

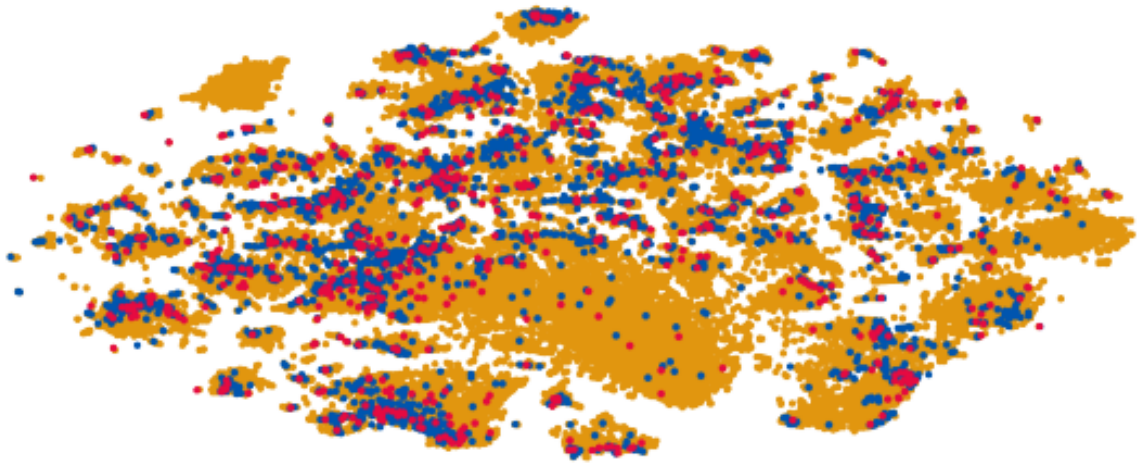
image credit: image from `https://networks.skewed.de/net/facebook_friends` based on data from BF Maier and D Brockmann

Notes:

- Let us now study motivating examples for applications of machine learning in data with an (additional) graph structure. Those examples highlight the rich interdisciplinary applications of graph learning. They also exemplify different supervised and unsupervised graph learning tasks that we address in more detail in the coming weeks.
- As a first example, consider online social networks like, e.g. Instagram, Twitter, or TikTok. Each node in the network is a user account, and (directed or undirected) links represent relationships between two accounts, e.g. one user following another user or two users being friends with each other. The example network depicted above shows an undirected network of friend relationships in a sample of Facebook users.
- Consider the case where a new user joins the online social network, initially adding links to some of his/her acquaintances to the profile. Can we use the network of existing relationships to automatically recommend accounts that could be of interest for the new user?
- This question can be cast as a **link prediction problem**, i.e. given a set of links we predict which unobserved links are likely to exist in the network. We cover different techniques to address this in week seven and eight of our course.
- On the one hand, such techniques add value both for the new user and existing users as we help to discover accounts they may wish to interact with. On the other hand, this can have negative implications for privacy, i.e. platforms can learn about relationships that users did not wish to disclose or they can create so-called shadow profile, i.e. they can collect information on users that do not even use the platform.

Example 2: eCommerce

consider a **product-product co-purchasing network**, where nodes represent **products** and links represent **products frequently bought together**



question

can we automate the assignment of **products to categories**?

graph learning task

we can cast this as an **node classification** problem

→ week 10/11

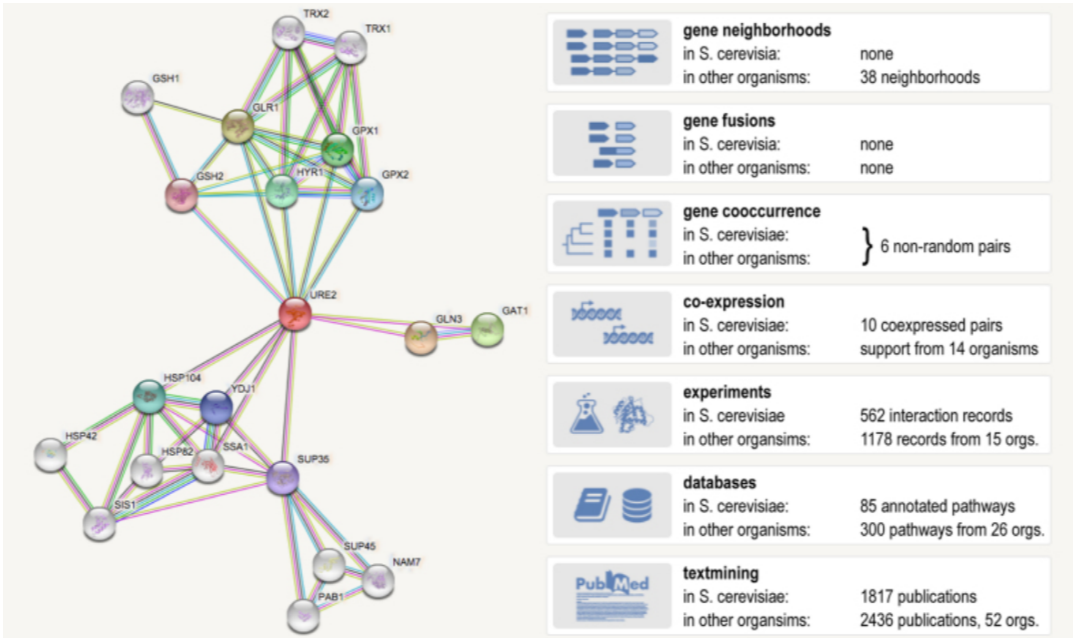
image credit: T-SNE visualization of network from → W Hu et al., NeurIPS 2020 based on data from → WL Chiang et al., SIGKDD 2019

Notes:

- As a second example, consider a network where nodes represent products offered by an eCommerce platform. Those products that are frequently purchased together are connected by a (weighted) link. The resulting graph structure is likely to capture thematic relationships between products, i.e. we can hypothesize that groups of products that are often bought together have something in common. The visualization shown above shows an Amazon co-purchasing network from the Open Graph Benchmark data set (see references below).
- Can we use such data to automatically augment the product catalogue of the platform, e.g. by automatically assigning products to product categories?
- This can be phrased as a node classification problem, i.e. we seek to assign nodes to a discrete set of classes. In week 10 an 11 of our course, we will cover techniques that can be used to address this task.
- Such techniques can help providers of eCommerce platforms to maintain and augment their product catalogue. It helps to limit the amount of boring manual categorization work and can assist us in finding inconsistencies in the catalogue.

Example 3: Protein-protein networks

consider a **protein-protein association network**, where nodes represent **proteins in a cell** and links represent **functional associations**



question

can we identify groups of proteins that jointly perform a **biological function**?

graph learning task

we can cast this as a **community detection** problem

→ week 02/04/05

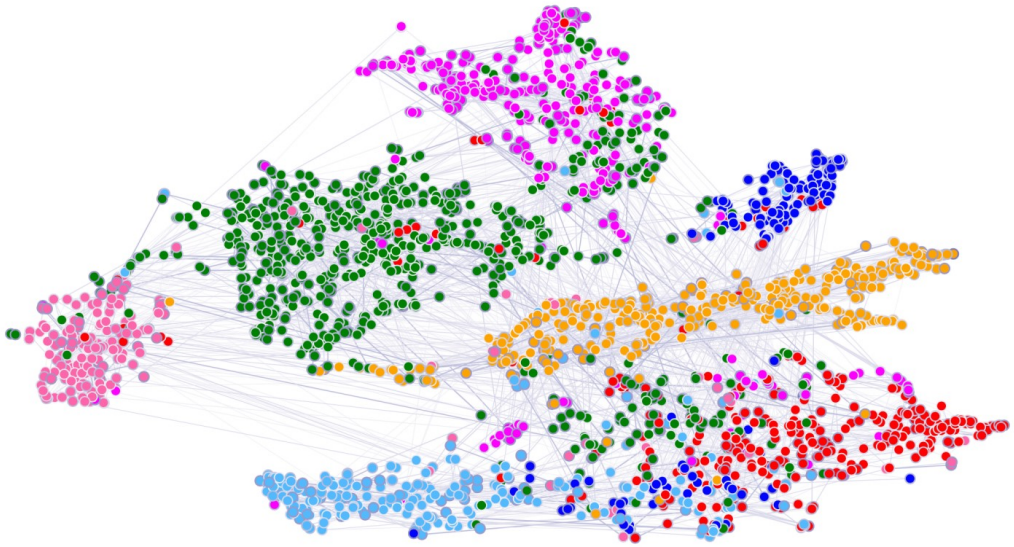
image credit: image from → D Szklarczyk, Nucl. Acids. Res., 2019

Notes:

- Leaving the online world, consider a network of protein-protein interactions within a (human) cell. Here nodes represent proteins and a link between two proteins captures that they are engaged in a specific biological function. Such functional associations between proteins can be inferred based on various measurements in the lab, e.g. through electron microscopy, spectroscopy, fluorescence methods, etc. The network depicted above shows possible functional associations between the yeast protein URE2 (in the center) and ten other proteins based on the STRING database. This database covers different selectable sources for functional associations (including gene co-expression or text mining on scientific articles).
- Can we use the resulting network to identify groups of proteins that jointly perform specific biological functions?
- We can cast this question as a **cluster or community detection** problem, i.e. we seek to identify groups of nodes in the network that are more strongly connected (or – in a network-sense – "similar") to each other than to nodes belonging to other groups. We have already covered some basic community detection techniques in our course *Statistical Network Analysis*. We will repeat one of those approaches in week two. In week four and five, we will introduce more advanced techniques, which directly build upon concepts introduced in this course.
- Methods to detect clusters in protein-protein interaction networks can help us to identify targets for therapeutic interventions, better understand gene functions, or identify genes associated with certain diseases.

Example 4: Citation networks

consider a **citation network** where nodes are **scientific articles** and links are **citations**



question

can we use **network structure and node features** (e.g. term frequencies in articles) to infer a **latent semantic space** of scientific articles?

machine learning task

we can cast this as a **node embedding** or **representation learning** problem → weeks 08, 10, 11

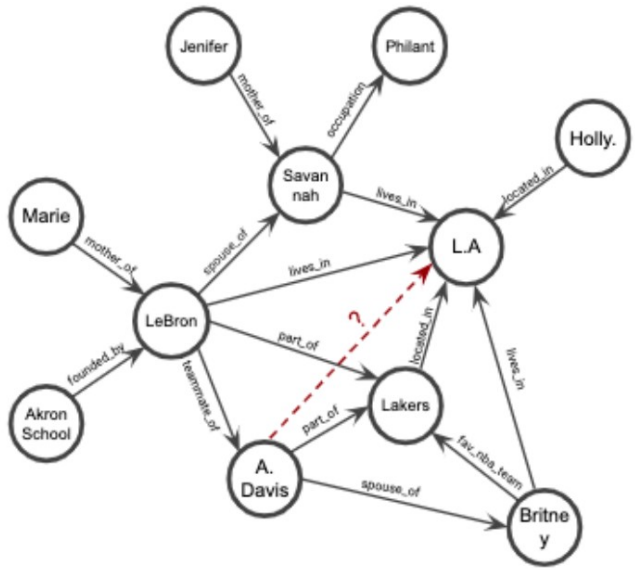
image credit: CORA citation network, image from → P Velickovic et al., ICLR 2018

Notes:

- Let us now consider a citation network, where nodes are published scientific articles and directed links indicate that one articles cites another article. In an idealized setting where articles cannot cite articles published in the future, we obtain a directed acyclic graph. Due to preprints, cycles appear in real citation networks, e.g. two articles can cite each other. The figure above shows the article-article citation network of the CORA data set, where node colors represent topic areas.
- Apart from citations, we can consider **additional vectorial features** of articles. For instance, we can use the full text of articles and count how often certain terms occur. This yields vectors of term-frequencies that we can assign to nodes. Both the topology of citations and the term frequencies contain information about the topics addressed by the articles.
- Can we use such data to **learn a latent semantic space** in which we can place the articles? That is, we are interested in a positioning of nodes in a vector space such that articles placed in close proximity are “similar” in terms of the topic that they address.
- This question can be cast as a **node embedding or graph representation learning** problem. We cover different methods to address this in week 08, 10 and 11.
- Techniques to learn vector space representations of networks are very important in practice. They can also help us to address the three examples mentioned before: We collaboration predict links based on distances between nodes in a vector space, apply classification algorithms like SVMs or neural networks to classify nodes, or use clustering methods like k-means to the resulting vector space to identify, e.g., scientific communities or disciplines.

Example 5: Knowledge Graphs

consider a **knowledge graph** where nodes are entities and links are relationships



question

can we use **network structure and entity data** to predict relationships between entities?

graph learning task

we can cast this as a **link prediction/classification** problem → weeks 07, 12

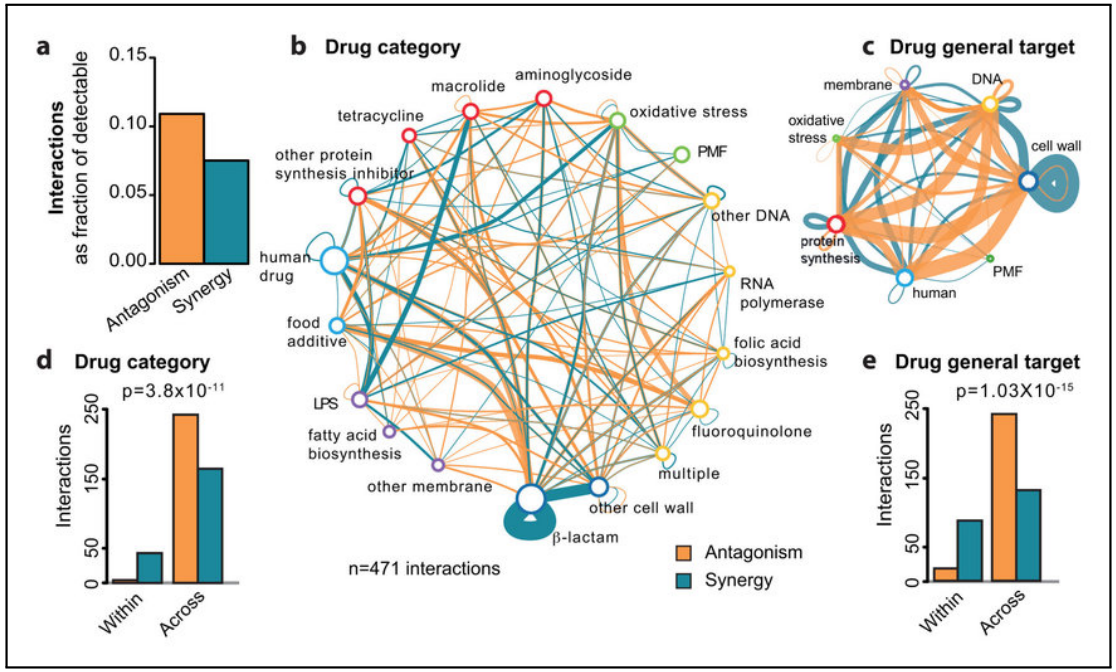
image credit: toy example for knowledge graph from → Teru et al. 2020

Notes:

- In the previous examples, we either exclusively used the topology of links (e.g. to predict links or find clusters) or we assumed that nodes have additional features or classes. Let us consider an example for a knowledge graph, where both nodes and links can have additional features or labels that we may want to use in the learning task. A knowledge graph consists of so-called triplets. Each triplet consists of two entities (represented as nodes) and the type of relationship between those entities. A simple example for a knowledge graph where entities include people, sports teams, places, etc. is shown above.
- A common problem with this approach to model knowledge is that it requires a lot of (manual) effort. Assuming that we are given an (incomplete) set of triplets on a certain set of entities, we are thus interested in methods that help us to infer additional relationships. In the example above, given that A. Davis is a part of “Lakers”, which is located in L.A., what can we say about the relationship between A. Davis and L.A.?
- If we were only interested in the presence of a relationship, we could phrase this as a link prediction problem. However, here we have additional labels that we seek to predict, i.e. we can either cast this as multiple link prediction problems (one for each type of relationships) or as a **link classification problem**.
- A method to automatically predict and classify links in knowledge graphs can help us to augment incomplete triplet data. This can be done using **inductive or transductive graph learning techniques**. The latter are are helpful for **semi-supervised learning problems**, where we may not have labels for all training data.

Example 6: Drug networks

consider a **drug-drug interaction network** where nodes are **drugs** and links are **joint prescriptions**



research question

can we use **network topology and node features** (e.g. drug agent/target, indication, etc.) to predict which joint prescriptions lead to **adverse side-effects**?

graph learning task

we can cast this as a **link classification** problem

→ week 12

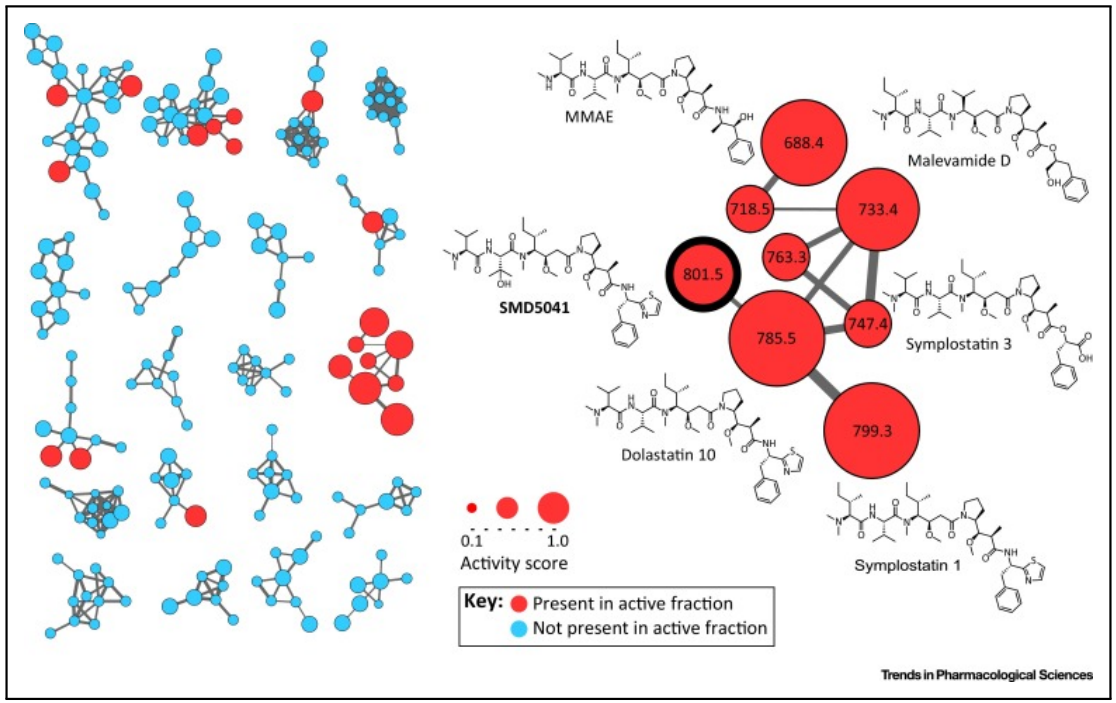
image credit: figure from → AR Brochado et al., Nature, 2018

Notes:

- A similar link-centric problem emerges in the study of drug-drug interaction networks, where nodes are drugs and links represent joint prescriptions of those drugs. Here we often have additional features at the level of nodes, e.g. the type of indication leading to the prescription or the drug target. We can further assign adverse side-effects resulting from a joint prescription to the links. The example network above shows antagonistic and synergetic effects observed in the joint administration of different types of antibacterial drugs in E. Coli.
- Given a set of nodes and (labelled) links, can we predict possible side effects for existing joint prescriptions for which we do not yet have information on adverse side-effects? Can we predict which new joint prescriptions could be safe?
- This question can be cast as a link classification problem, where we want to incorporate the topology of links, link features/labels as well as node features.
- Methods to address this problem can help us to uncover adverse side-effects that may not have been related to the joint-prescription of drugs, or identify new combinations of drugs, whose joint administration may exhibit synergetic effects.

Example 7: Molecular networks

consider **molecular networks**, where nodes represent **atoms** and links represent **chemical bonds**



question

can we use network structure and node/link features to automatically **discover new drugs**?

graph learning task

we can cast this as a **graph classification** problem

→ week 12/13

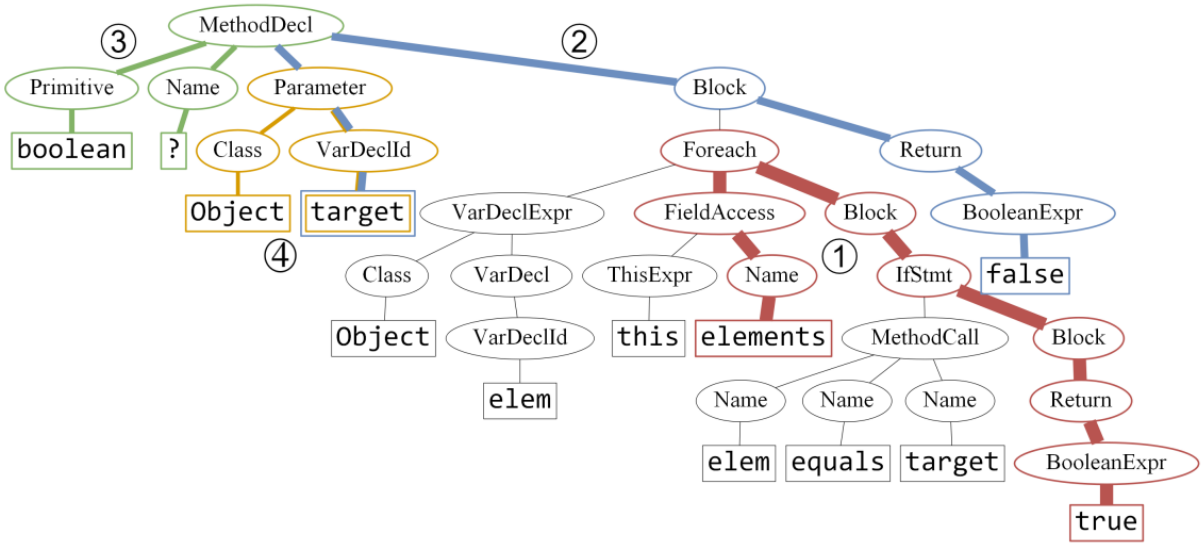
image credit: figure from → RA Quinn et al., Trends in Pharmacological Sciences, 2017

Notes:

- For the examples considered so far, we were interested in either the nodes or links within a single graph that represented our data set. But what if we data where a graph is only one data point? What if we have a collection of graphs and want to answer a question about one specific graph? Consider a data set of chemical compounds, where each chemical compound is represented as a molecular graph. Within such a graph, the nodes represent atoms and links represent chemical bonds between atoms. Both nodes and links can have additional features (e.g. the type of atom or properties of the chemical bond). In addition, we can have graph-level properties that relate to the chemical compound as a whole, e.g. the toxicity of the substance, therapeutic effects, or its solubility in water.
- Given a set of molecular networks representing different chemical compounds, where some of those chemical compounds have known properties, can we use the topology and node/link properties of molecular networks to discover new promising drugs?
- This can be cast as a **graph classification problem**, where we wish to assign classes to each instance of a molecular network. Hence, different from the examples before, here the whole graph represents one observation in our data set, rather than the collection of nodes or edges.
- Network-based methods to predict properties of chemical compounds have developed into an important approach in computer-assisted drug discovery and precision medicine. See the reference above for more details.

Example 8: Code networks

consider **abstract syntax trees**, where nodes represent **programming language constructs** and links represent the **syntactic structure of code snippets**



question

can we infer a **latent semantic space** that captures the meaning of code snippets?

graph learning task

we can cast this as a **graph embedding** problem

→ week 12/13

image credit: figure from → U Alon, M Zilberstein, O Levy, E Yahav, POPL 2019

Notes:

- With our final example of a graph-level learning task, we return to the domain of computer science. For this, consider an abstract syntax tree, i.e. an acyclic network where nodes are programming language constructs (e.g. variables, operators, control structures) and links represent the syntactic structure of a code snippet (e.g. one source code file, a class, or a method). Each code snippet is represent by a network, and nodes and links may have additional features. The example above shows an abstract syntax tree of a single Java method declaration (see reference for mode details).
- Can we use such a graph representation of a code snippet to position the code in a latent vector space? And can we use the resulting latent space to assist software developers, e.g. by automatically suggesting meaningful method names, discover code clones, or help in refactoring code?
- We can cast this problem as a **graph-level embedding** problem, e.g. rather than using the topology of links to embed nodes in a vector space, here we use a collection of graphs to embed each graph into a latent space.
- This example is one instance of **machine learning applications in software engineering**, which are becoming increasingly sophisticated and important in practice. You will find that some of those applications have already been implemented in modern IDEs (integrated development environments) like Eclipse or Visual Studio Code.

A first Graph Learning taxonomy

	(semi-)supervised	unsupervised
node level	node classification → example 2	node embedding community detection → examples 3, 4
edge level	link prediction link classification → examples 1,5,6	link prediction network reconstruction → example 1
graph level	graph classification graph regression → example 7	graph clustering graph embedding → example 8

Notes:

- The eight examples introduced on the previous slides highlight the interdisciplinary relevance and appeal of graph learning beyond computer science. They further showcase different types of graph learning problems, which we can roughly categorize as depicted above.
- We have seen examples for learning problems at the level of nodes, links, or whole graphs. We have further seen examples both for **(semi-)supervised and unsupervised machine learning in graphs**. Embedding and clustering tasks are examples for unsupervised problems, as we seek to model patterns in data without ground truth labels. Classification tasks are a common example for a supervised problem, where ground truth labels (either on nodes, links or graphs) are available. Note that we can also have semi-supervised problems, where labels may only be available for a (small) subset of the training examples. Finally, note that some problems, like link prediction, can be addressed both in a supervised or unsupervised fashion.
- A note on graph regression problems: While we have not considered an example for this, we can consider the supervised learning problem of predicting a numerical quantity that is associated with a given graph (i.e. similar to linear regression, where a graph is considered the independent variable).
- Moreover (and not depicted in the necessarily incomplete taxonomy above) in some of the examples we have additional features at the level of nodes, links or graphs, which we can use to improve the performance of our algorithms.

Challenges in Graph Learning

- (semi-)supervised and unsupervised tasks at level of **nodes, links, and graphs**
- we can have **additional features** for nodes, links, and graphs
- we often have **incomplete/unreliable data**, e.g. graph reconstruction, link prediction, node disambiguation
- **small networks** with dozens vs. **large networks** with hundreds of millions of nodes
- large network = high-dimensional object $\Rightarrow$ **curse of dimensionality**

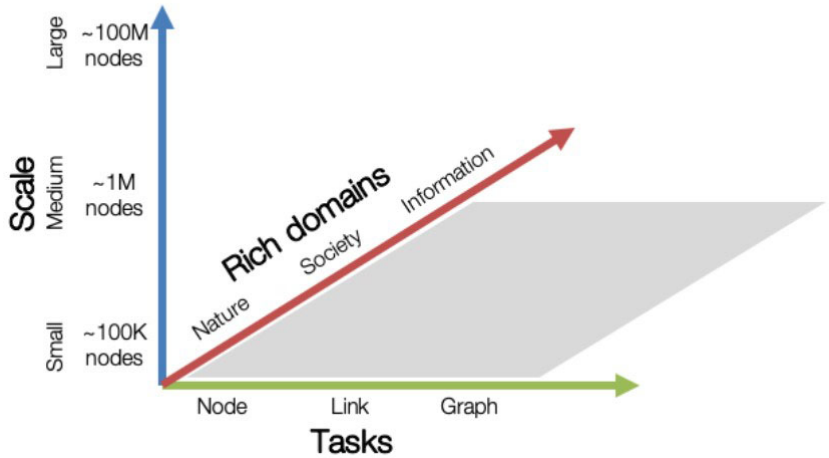


image credit: OGB Dataset Overview, `https://ogb.stanford.edu/docs/dataset_overview/`

Notes:

- Apart from exemplifying supervised and unsupervised learning tasks, the examples also illustrate challenges that we face when we apply machine learning to data on complex networks.
- First, as mentioned before, we can address **supervised and unsupervised learning tasks at the level of nodes, links, and whole graphs**. This generates a diverse set of interesting learning challenges that can be applied in various domains.
- In addition to the graph topology, which tells us which nodes are directly or indirectly connected to which other nodes, we also often have **additional (Euclidean) features**. Those features can again relate to nodes, edges, or the graph as a whole, which generates problems that require different methods.
- We have seen that graph learning problems occur in **many different domains**, be it in information systems, social systems, natural systems, or engineered infrastructures. In many of those domains, data are either incomplete or unreliable, which introduces interesting machine learning challenges by itself. We are often confronted with challenges that are either due to **small or large networks**. In small networks, it may be difficult to reliably reason about patterns. In large networks with hundreds of millions of nodes we face computational challenges.
- While many data sets in machine learning have millions of observations, an additional complication is due to the fact that a network with n nodes is – in general – an n-dimensional object. The resulting **curse of dimensionality** creates challenges for generalizability and requires dimensionality reduction techniques.

The present and future of AI

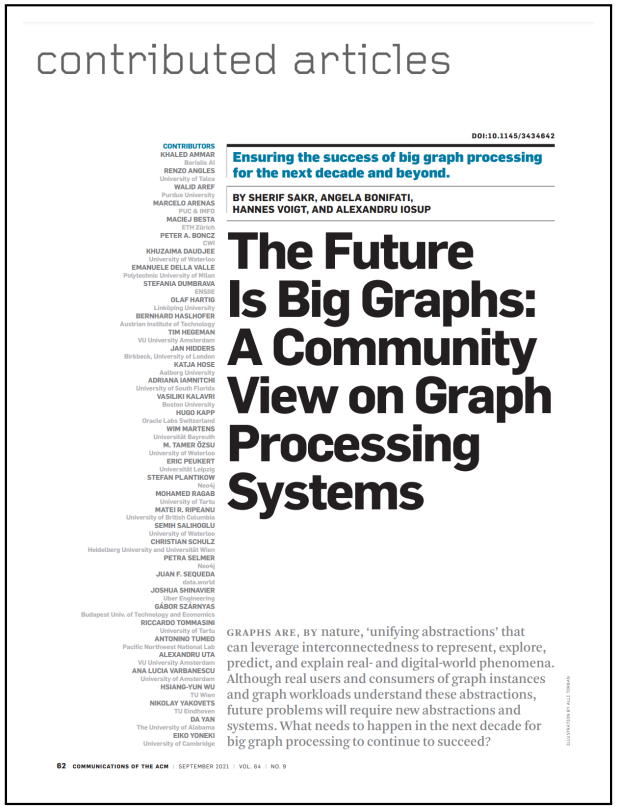
graphs and networks

graphs and networks = **universal mathematical abstraction for complex systems** consisting of many interacting elements

- **graph-structured data are ubiquitous**

> "We are witnessing an unprecedented growth of interconnected data, which underscores the vital role of graph processing in our society. Instead of a single, exemplary ("killer") application, we see big graph processing systems underpinning many emerging [...] data management ecosystems, in many areas of societal interest." → S Sakr et al. Comm. of the ACM, 2021

- vibrant, interdisciplinary and international **research community**
- growing **interest from industry**


contributed articles

DOI:10.1145/3434642

Ensuring the success of big graph processing for the next decade and beyond.

BY SHERIF SAKR, ANGELA BONIFATI, HANNES VOIGT, AND ALEXANDRU IOSUP

The Future Is Big Graphs: A Community View on Graph Processing Systems

GRAPHS ARE, BY nature, 'unifying abstractions' that can leverage interconnectedness to represent, explore, predict, and explain real- and digital-world phenomena. Although real users and consumers of graph instances and graph workloads understand these abstractions, future problems will require new abstractions and systems. What needs to happen in the next decade for big graph processing to continue to succeed?

62 COMMUNICATIONS OF THE ACM | SEPTEMBER 2021 | VOL. 64 | NO. 9

image credit: Sakr et al., Comm. of the ACM, 2021

Notes:

- Besides those challenges, there are many opportunities to apply machine learning for complex networks in practice. This is due to the fact that graphs/networks are universal and powerful mathematical abstractions that is widely used to study complex systems that consist of many interacting elements. For more details on this **complex systems perspective on networks** – and its intriguing relations to statistical physics – I recommend our course **Statistical Network Analysis**. Note, however, that it is not mandatory that you completed this earlier course. You can also take those complementary courses in reverse order, i.e. first **Machine Learning for Complex Networks** and then **Statistical Network Analysis**.
- Graph-structured data are ubiquitous across disciplines, which results in the fact that graph mining, graph learning, and graph processing techniques have recently become one of the most popular topics in machine learning and data science. Graph learning techniques techniques are developed by a vibrant, interdisciplinary and international research community. The latest advances in the field can be found both in top venues in machine learning or data mining, as well as in leading journals in physics or the life sciences. Exemplary conferences include: *The Web Conference* (WWW), *KDD*, *NeurIPS*, *ASONAM*, *ICLR*, or the newly established *Learning on Graphs* conference. You can also find relevant foundational works in natural science journals like *Physical Review Letters*, *Nature*, *Science* or interdisciplinary venues like *NetSci*.
- There is **large interest from industry** as well, which is confirmed by a growing number of startups that focus on network analysis and graph learning.

Weekly Lectures

Theoretical Concepts

We introduce key concepts and methods in machine learning for networks.

→ handout of slides with notes on WueCampus

Practice Session

In integrated practice sessions, we demonstrate how these concepts can be implemented in `python`.

→ jupyter notebooks in gitLab

script, lecture handouts and link to git repository available on WueCampus

`https://wuecampus.uni-wuerzburg.de/moodle/course/view.php?id=66132`
`https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_sose_ml4nets_notebooks`

Notes:

- We will adopt a similar concept as for our course “Statistical Network Analysis”, which some of you have attended. We combine a series of lectures (i.e. traditional lectures with slides) with integrated practice sessions, in which we interactively show how to implement and apply machine learning techniques.
- Different from the lecture *Statistical Network Analysis* in winter, which had seen multiple prior editions at ETH Zürich, University of Zurich and University of Wuppertal, **this is a new course** that is not based on an existing book or lecture. So you are the very first generation of students and we appreciate your feedback!
- We will provide **handouts of lecture slides** with basic comments on each slide. Since this is the first edition, these comments are likely to be less complete than what you are used to from the previous course. You should rather see them as a basis that you complement with your individual notes.
- The material of the integrated practice sessions consists of `jupyter` **notebooks**, which you can run yourself and which you can use as a seed for your own experiments. The practice session should illustrate and complement the contents of the lectures. They allow you to practically explore the theoretical concepts and models. The accompanying notebooks and data sets will be made available in a `git` repository hosted on the university’s `gitLab`. The link is available on WueCampus (and on the slide above).

I - Foundations of Graph Learning

L01 Motivation **17.04.2024**

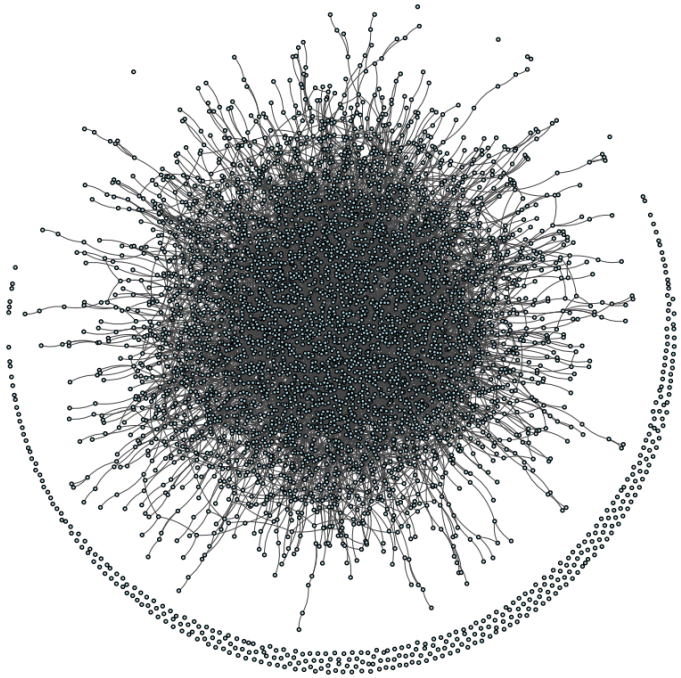
We introduce our Chair and motivate challenges and opportunities of machine learning in data with a graph structure. We further give an overview of topics covered in the course and discuss organizational issues.

L02 Graph Theoretic and Algorithmic Foundations **24.04.2024**

We show how we can mathematically represent graphs and networks. We introduce basic graph-theoretic concepts and show how we can use the graph Laplacian to detect communities in networks.

L03 Generative Models and Inference **08.05.2024**

We introduce generative models of random graphs and explain how we can use them to define statistical ensembles of random graphs. We then illustrate how we can infer model parameters based on empirical data on complex networks.



random network generated by
$G(n, p)$ **Erdös-Rényi model**

Notes:

- We present the general outline of the course, which consists of four thematic chapters. A first chapter introduces the **foundations of graph theory and statistical learning in networks**. Apart from basic definitions and concepts, we first introduce the graph clustering problem and show how it can be addressed, e.g. using methods that were covered in more detail in the course Statistical Network Analysis.
- Linking machine learning in graphs to general purpose machine learning, we show how cluster detection in Euclidean spaces can be reformulated as a component analysis and community detection problem.
- In a final lecture of the first chapter we introduce key generative models and explain how we can use them to define statistical ensembles of random graphs. Linking complex networks and statistical inference, we finally show how we can learn the parameters of those models based on a given empirical network.

II - Community Detection & Link Prediction

L04 Stochastic Block Model 15.05.2024

We illustrate ensemble-based statistical learning with the stochastic block model, a generative approach to detect communities in networks.

L05 Entropy and Description Length 22.05.2024

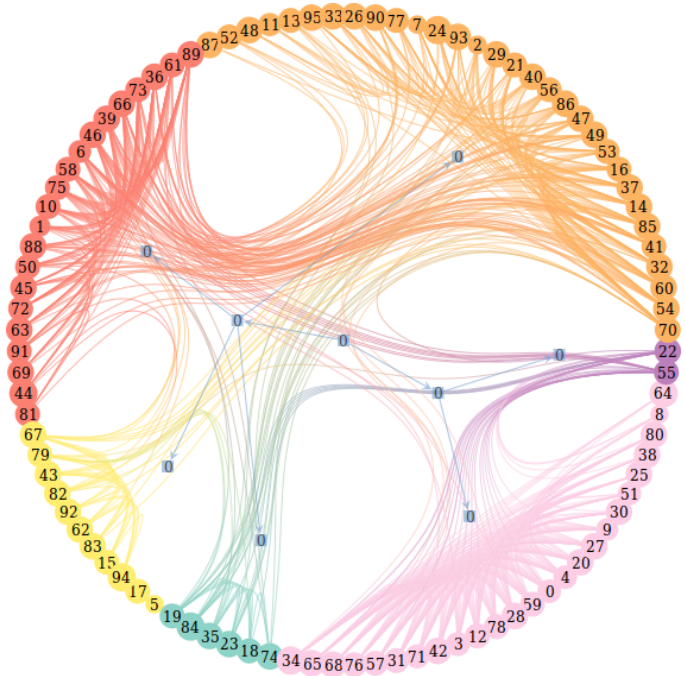
We show how the entropy of ensembles can be used to evaluate the description length of a model. We demonstrate how we can use this to infer parsimonious community structures.

L06 Random Walks and Flow Compression 29.05.2024

We introduce an information-theoretic method to detect cluster structures based on the compression of random walks in networks.

L07 Node Similarities and Link Prediction 05.06.2024

We introduce statistical and heuristic approaches for the supervised and unsupervised prediction of links.



natural communities detected in a social network

image credit: figure from → G Casiraghi, V Nanumyan, I Scholtes, F Schweitzer, SocInfo 2017

Notes:

- A second chapter of the course is devoted to **community detection and link prediction**. Building on the statistical ensembles and inference techniques introduced in Lecture 03, we first introduce the stochastic block model, which can be used to detect cluster patterns in networks.
- We then define the entropy of random graph ensembles, which enables us to quantify the information content of a network realization. This is the basis to evaluate the description length of the stochastic block model, which can be used to infer parsimonious community structures in networks.
- Completing the statistical inference view, we introduce random walks in networks. We then show how description length minimization can be applied to compress trajectories of random walks. We introduce the MapEquation and demonstrate how it enables us to detect optimal hierarchical community structures in networks.
- In a final lecture of this chapter, we turn our attention to link prediction in networks. We cover both statistical and heuristic approaches to address this problem, which has important applications in recommender systems.

III - Deep Learning in Graphs

L08 Graph Representation Learning **12.06.2024**

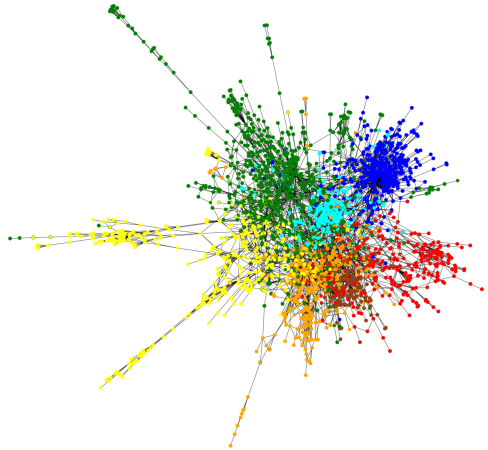
We introduce basic techniques to learn low-dimensional representations of data. We then show how matrix decomposition can be applied to learn latent vector-space representations of graphs.

L09 Supervised Graph Learning **26.06.2024**

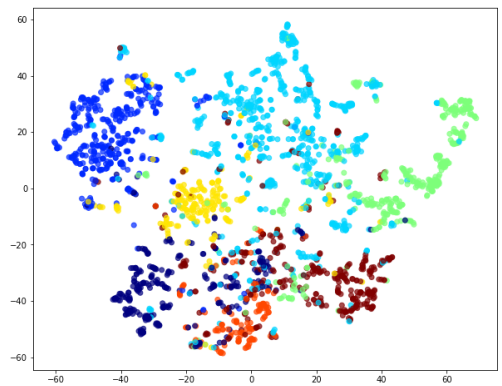
Starting from logistic regression we show how we can use neural networks to address classification problems. We show how we can learn the parameters of deep neural networks using stochastic gradient descent.

L10 Neural Graph Representation Learning **03.07.2024**

Building on the random walk model introduced in L06, we show how random walks and deep neural networks can be used to learn node embeddings for downstream graph learning tasks.



CORA citation network



vector space embedding of articles generated by `node2vec`

image credit:
`https://stellargraph.readthedocs.io`

Notes:

- In a final chapter of the course, we cover state-of-the-art **deep learning techniques for data with a graph structure**.
- We first show that it can be useful to map networks to (low-dimensional) vector spaces. Learning such vector space representations enables us to apply standard, Euclidean machine learning techniques to graph-structured data.
- Linking to dimensionality reduction techniques like PCA or SVD, we introduce the graph Laplacian and show how its eigenvectors and eigenvalues can be used to map nodes to vectors.
- We then consider logistic regression and highlight its relationship to the perceptron model and feed-forward neural networks.
- We next show how deep neural networks can be applied to random walk trajectories to embed the nodes of a graph into a vector space.

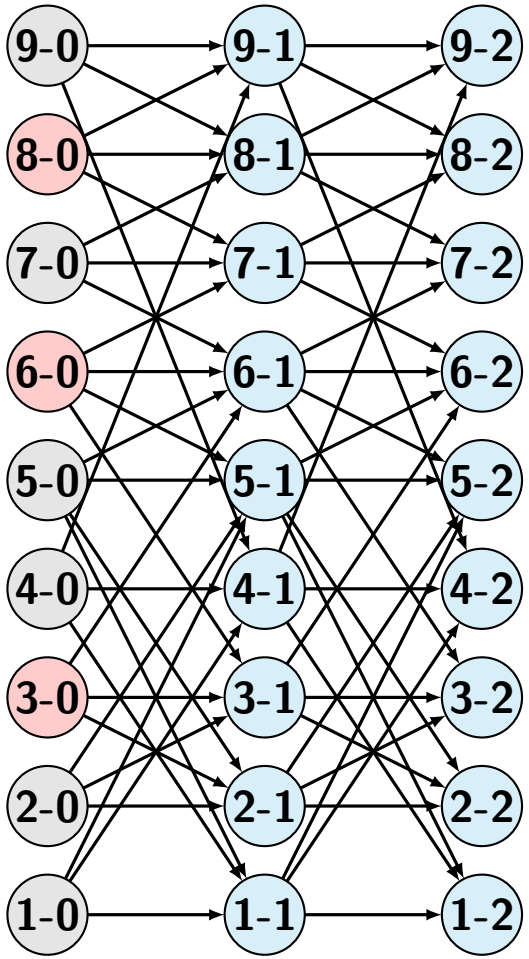
III - Deep Learning in Graphs

L11 Graph Neural Networks **10.07.2024**

We explain how we can use message passing in neural networks to calculate graph convolutions, which allow to apply deep neural networks to graph-structured data.

L12 Repetitorium **TBD**

In the last week of our course, you have the chance to address open questions and we will repeat key sections of the course upon request.



computation graph of a graph convolutional neural network

Notes:

- We show how message passing can be used to generalize the idea behind convolutional networks to graphs.
- In the final week of the course, you will get a chance to address open questions and we offer to repeat parts of the course based on your request.

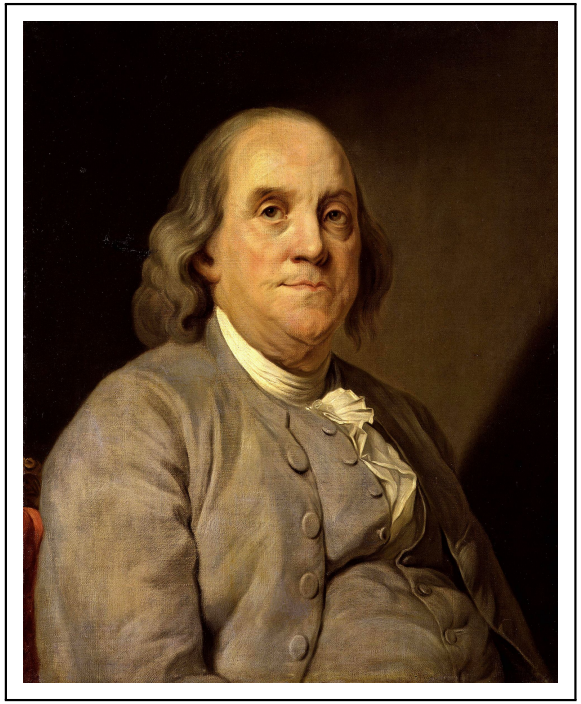
Exercises

weekly exercise sheet

- deepen your knowledge of theoretical concepts
- practically apply machine learning analysis in `python`
- gain bonus points for the exam

preliminary exercise schedule

exercise sheet E01	**24/04/24 – 01/05/24**
exercise sheet E02	**08/05/24 – 15/05/24**
exercise sheet E03	**15/05/24 – 22/05/24**
exercise sheet E04	**22/05/24 – 29/05/24**
exercise sheet E05	**29/05/24 – 05/06/24**
exercise sheet E06	**05/06/24 – 12/06/24**
exercise sheet E07	**12/06/24 – 19/06/24**
exercise sheet E08	**19/06/24 – 26/07/24**
exercise sheet E09	**26/06/24 – 03/07/24**
exercise sheet E10	**03/07/24 – 10/07/24**



Benjamin Franklin

1706 – 1790

quote

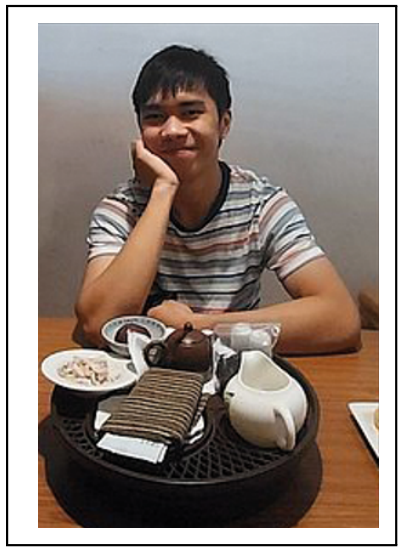
"Tell me and I forget, teach me and I may remember, involve me and I learn" → Benjamin Franklin

Notes:

- The exercises consist of weekly assignments that cover both theoretical concepts, proofs, calculations as well as practical assignments that can be solved in `python`.
- The solutions to the assignments must be uploaded to WueCampus before the deadline given on the exercise sheet. Correct solutions will yield bonus points that help you pass the exam. You also have the chance to present your solutions in the exercise session, which yields additional bonus points that can improve your grade (but will not help you to pass the exam). The concept of the exercises (and bonus points) is identical to our course **Statistical Network Analysis**.
- The exercise schedule is given above and we plan for a total of ten exercise sheets. The first sheet will be released in week two and it will be due one week later. The first exercise session will take place in week three, i.e. one week after the submission deadline of the first sheet. A preliminary schedule of exercise sheets can be found above, we reserve the right to change this schedule.
- In case you have not worked with `python` before, we provide material of an introductory tutorial that introduces the basics that you need in the course. You can find this material in the gitLab repository linked on WueCampus.

Meet the tutors

Dr. Vincenzo Perri



Chester Tan

exercise slots

- **Monday 10:15 - 11:45, SE 8**
- **Thursday 10:15 - 11:45, ÜR 1**
- **Friday 10:15 - 11:45, SE III**

date selection on WueCampus opens April 17th

Notes:

- The exercise sessions will be organized by my assistants. Depending on the number of students, we offer slots on Monday, Thursday or Friday. Apart from organizing the exercise sessions, the tutor will also be responsible for the correction and grading of your submissions.
- A date selection will open on WueCampus on April 17th. Please choose your group until April 24th.

Self-study and exam

final exam

written, closed-book exam of 120 minutes

- no `python` skills required
- exercises = bonus points
- preliminary exam date: **July 17 (TBC)**

exam registration

mandatory exam registration via WueStudy

- 16.04.2024 – 15.07.2024 (Z3)
- “Ausgewählte Kapitel der Informatik” (325653, Z3)
- “Ausgewählte Kapitel d. Intell. Systeme” (318914, Z3)
- “Selected Topics in AI Methods 2” (310859, Z3)
- “Machine Learning for Networks I” (340836, Z3)

assisting self-study at home

each lecture includes 8 – 10 **self-study questions**

- repeat most important issues
- test your comprehension
- useful for exam preparation

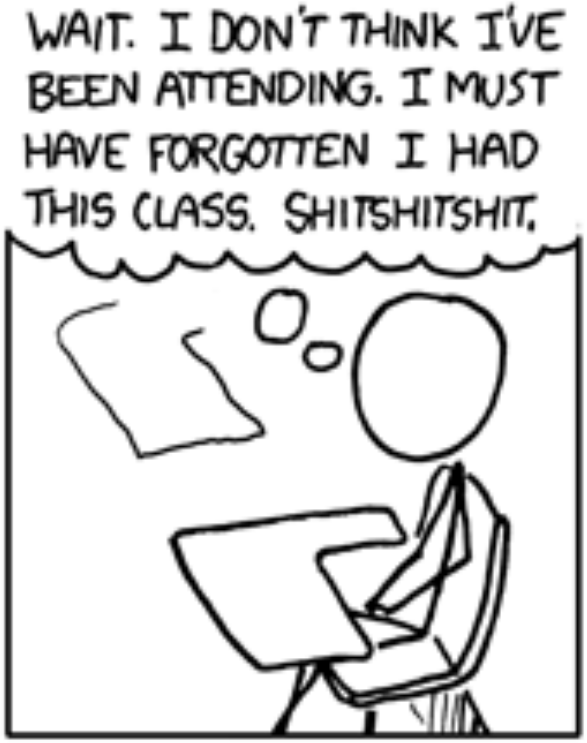
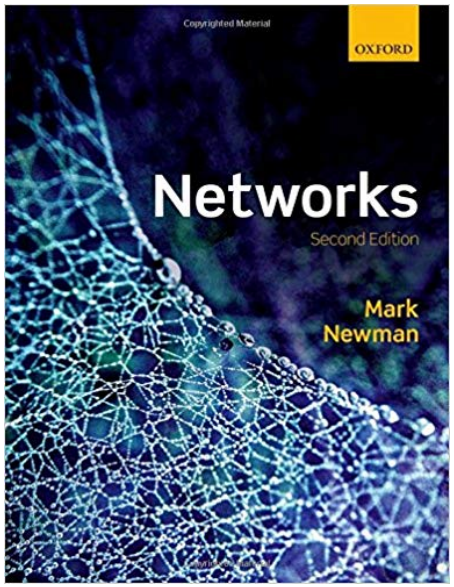


image credit: Randall Munroe, CC-BY-SA

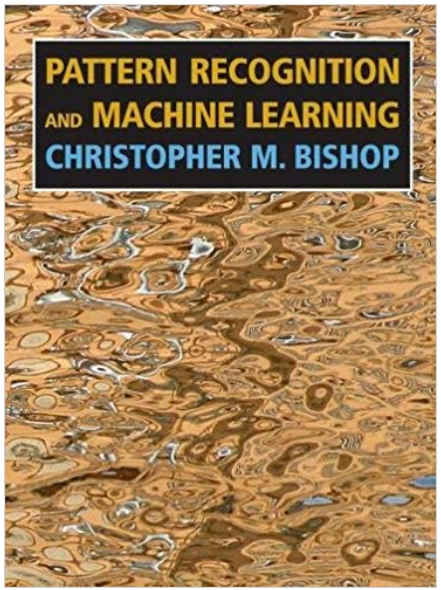
Notes:

- The final exam will be a two hour closed book exam, i.e. no books or lecture slides are allowed. We may allow you to use a single, hand-written sheet during the exam but we will finally decide about this later. For the exam, you will not have to program, so the emphasis is on a conceptual understanding of the methods rather than on programming skills.
- We further assist you in the self-studies at home, which are an important component of any course and which we account for in the ECTS credits as well. As a reminder, this course gives 5 credit points, which corresponds to a workload of 125 - 150 hours. The 14 lectures and 10 exercise sessions amount to 48 hours, which leaves an additional six to seven hours per week to (i) complete the exercise sheets, (ii) brush up your python and programming skills, and (iii) work through the slides, jupyter notebooks, and literature that we distribute.
- To assist you in the self-study and exam preparation, we include a set of self-study questions at the end of each lecture. You should be able to answer them based on the slides and you can see them as a proxy for the questions asked in the exam (along with the tasks in the exercise sheets).
- Note that it is mandatory to register for the exam. We suggest to register for the two modules mentioned above. For those modules, you should be able to register until July 15th. Please contact us if you cannot use any of the modules listed above, we may find another solution.

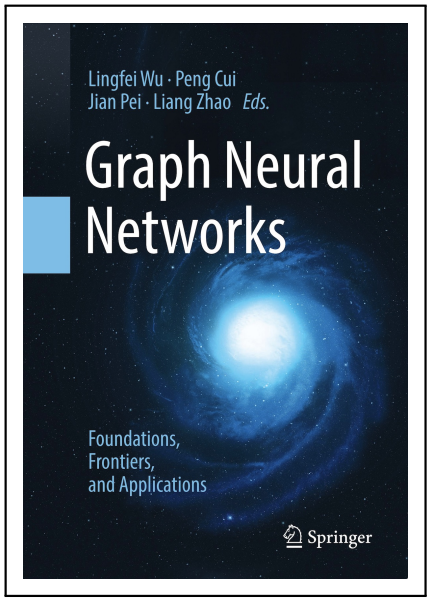
Literature



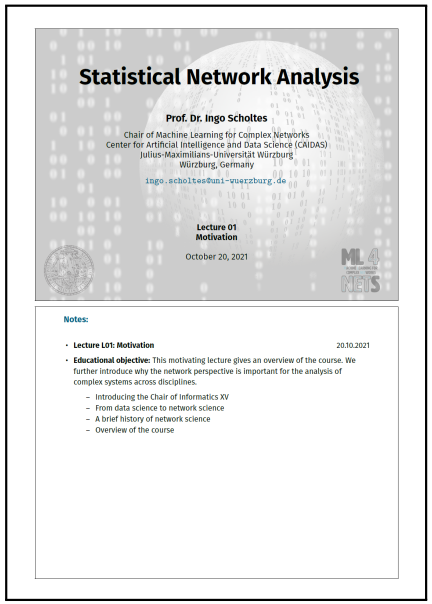
Mark Newman: **Networks**, 2nd edition, Oxford University Press, 2018



Christopher M. Bishop: **Pattern Recognition and Machine Learning**, Springer, 2006



Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao: **Graph Neural Networks: Foundations, Frontiers, and Applications**, Springer, 2021



Ingo Scholtes: **Statistical Network Analysis**, lecture notes available on WueCampus

Notes:

- As mentioned before, this course has been designed from scratch and we do not follow a specific book. However, since this course relies on basic concepts in graph theory, network analysis, and machine learning we refer to some books that cover those topics. In addition, we also recommend to study the lecture notes of our course **Statistical Network Analysis**, which we have made available on WueCampus.

Literature

- last slide of lecture includes **reading list** with related book chapters, articles, or web resources
- we provide **related full-text research papers** for each week of the course on WueCampus
- you are welcome to **suggest additional resources** that you encounter during your studies

reading list

- M Newman: **Networks**, Oxford University Press, 2010 → Chapter 1
- MM Bronstein, J Burna, Y LeCun, Arthur Szlam, P Vandergheynst: **Geometric Deep Learning: Boing beyond Euclidean data**, IEEE Signal Processing Magazine, July 2017
- Z Wu et al.: **MolecuNet: a bechmark for molecular machine learning**, Chemical Science, Vol. 9, 2018
- D Szklarczyk et al.: **STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental data sets**, Nucl. Acids Res., Vol. 47, 2019
- J Gilmer, S Schoenholz, PF Riley, O Vinyals, GE Dahl: **Neural Message Passing for Quantum Chemistry**, Proc. of PMLR, 2017
- A Fout, J Byrd, B Shariat, A Ben-Hur: **Protein Interface Prediction using Graph Convolutional Networks**, Prof. of NeurIPS, 2017
- ZM Zhang, L Liang, L Liu, MJ Tang: **Graph Neural Networks and Their Current Applications in Bioinformatics**, Front. in Genetics, Vol. 12, 2021
- T Hamaguchi, H Oiwa, M Shimbo, Y Matsumoto: **Knowledge Transfer for Out-of-Knowledge-Base Entities: A Graph Neural Network Approach**, XXX,
- W Fan et al.: **Graph Neural Networks for Social Recommendation**, Proc. of WWW, 2019
- WL Chiang et al.: **Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks**, Proc. of SIGKDD, 2019

exemplary “reading list” included at the end of each lecture

Notes:

- We will cover state-of-the-art techniques that have recently been published in machine learning outlets and we will provide links to the articles in which those techniques are explained in more detail. For this purpose, each lecture includes a reference list on the last slide. We will further upload a folder with full-text research papers that are related to the topics covered in each week of our course. We highly recommend to check those articles in order to understand the details and subtleties of the covered techniques.
- If you are interested in more details of a specific method, we invite you to participate in our seminar or lab, where you have the chance to work through some of those papers in detail.

Tools, Data, and Benchmarks

PATHPY

PyTorch

OGB

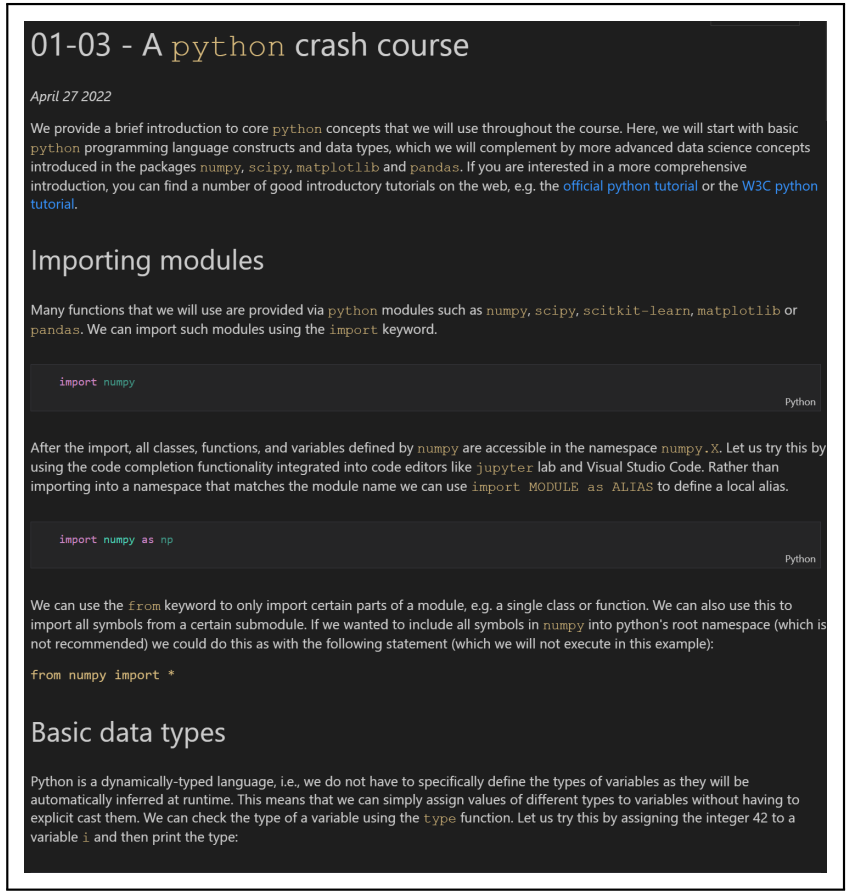
OPEN GRAPH BENCHMARK

Notes:

- For the practice sessions of our course, we will provide `jupyter` notebooks that heavily rely on certain tools, frameworks, and data sets. For the generation and visualization of networks, we will use `pathpyG`, a torch-based reimplementation of `pathpy`, which you may know from **Statistical Network Analysis**.
- The last two chapters of our course build on the popular deep learning framework `pyTorch` and its extension `torch-geometric` (pyG) for geometric deep learning.
- We finally mention **Open Graph Benchmark**, an open platform that provides node-, link-, and graph-level learning problems in realistic data sets. You can check existing solutions for those problems, implement your own models in `pyTorch` and check how their performance compares to state-of-the-art methods.

Practice Sessions - week 01

- we provide **12 tutorial notebooks** introducing data science with `python`
- important basis/reference for future **practice sessions and exercises**



01-03 - A `python` crash course

April 27 2022

We provide a brief introduction to core `python` concepts that we will use throughout the course. Here, we will start with basic `python` programming language constructs and data types, which we will complement by more advanced data science concepts introduced in the packages `numpy`, `scipy`, `matplotlib` and `pandas`. If you are interested in a more comprehensive introduction, you can find a number of good introductory tutorials on the web, e.g. the official python tutorial or the W3C python tutorial.

Importing modules

Many functions that we will use are provided via `python` modules such as `numpy`, `scipy`, `scitkit-learn`, `matplotlib` or `pandas`. We can import such modules using the `import` keyword.

```
import numpy
```
Python

After the import, all classes, functions, and variables defined by `numpy` are accessible in the namespace `numpy.X`. Let us try this by using the code completion functionality integrated into code editors like `jupyter` lab and Visual Studio Code. Rather than importing into a namespace that matches the module name we can use `import MODULE as ALIAS` to define a local alias.

```
import numpy as np
```
Python

We can use the `from` keyword to only import certain parts of a module, e.g. a single class or function. We can also use this to import all symbols from a certain submodule. If we wanted to include all symbols in `numpy` into python's root namespace (which is not recommended) we could do this as with the following statement (which we will not execute in this example):

```
from numpy import *
```

Basic data types

Python is a dynamically-typed language, i.e., we do not have to specifically define the types of variables as they will be automatically inferred at runtime. This means that we can simply assign values of different types to variables without having to explicit cast them. We can check the type of a variable using the `type` function. Let us try this by assigning the integer 42 to a variable `i` and then print the type:

Self-study tutorials

- 01-01 Setup of python environment
- 01-02 `git` and Visual Studio Code
- 01-03 A `python` crash course
- 01-04 Lists, Sets and Collections
- 01-05 Reading and writing data
- 01-06 Data management with `pandas`
- 01-07 Numerical computing with `numpy`
- 01-08 Scientific computing with `scipy`
- 01-09 Data visualization: `seaborn` & `matplotlib`
- 01-10 Representing networks with `pathpyG`
- 01-11 Reading and writing network data
- 01-12 Visualizing networks with `pathpyG`

`https://gitlab2.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_sose_ml4nets_notebooks/-/tree/master/week01`

Notes:

- Due to time constraints, in this course we cannot introduce basics of (interactive and collaborative) data science with `python`. For those of you who do not already have experience with `python`, data science packages like `pandas`, `seaborn`, `numpy` or `scipy`, or the network analysis and visualization package `pathpy`, we have prepared 12 tutorial notebooks that introduce the basics.
- We will not discuss these notebooks in the lecture, but you can find them in the gitLab repository accompanying this course. Please check the directory associated with week 01.
- We will provide more notebooks for future weeks of our course, in which we demonstrate how you can practically implement and apply the theoretical concepts covered in the lecture.

Related courses at JMU

AI and machine learning

- Lecture: **Data Mining**, Prof. Dr. Hotho
- Lecture: **Programmieren mit neuronalen Netzen**, Prof. Dr. Puppe
- Lecture: **Künstliche Intelligenz**, Prof. Dr. Puppe
- Lecture: **Sprachverarbeitung und Text Mining**, Prof. Dr. Hotho
- Lecture: **Time Series Analysis and Forecasting**, Prof. Dr. Bauer
- Lecture: **Information Retrieval**, Prof. Dr. Hotho
- Lecture: **Machine Learning for Natural Language Processing**, Prof. Dr. Hotho
- Lecture: **Computer Vision**, Prof. Dr. Timofte
- Praktikum: **Machine Learning**, Prof. Dr. Hotho
- Praktikum: **Machine Learning for Time Series Analysis**, Prof. Hotho
- Praktikum: **Computer Vision**, Prof. Timofte
- Praktikum: **Graph Neural Networks**, Prof. Dr. Scholtes
- Seminar: **Ausgewählte Kapitel des Machine Learning**, Prof. Hotho
- Seminar: **Aktuelle Trends in der Künstlichen Intelligenz**, Prof. Puppe
- Seminar: **Computer Vision**, Prof. Timofte
- Seminar: **Machine Learning for Complex Networks**, Prof. Dr. Scholtes

network analysis / graph theory

- Lecture: **Statistical Network Analysis**, Prof. Dr. Scholtes
- Praktikum: **Statistical Network Analysis**, Prof. Dr. Scholtes
- Lecture: **Visualisierung von Graphen**, Prof. Dr. Wolff
- Lecture: **Algorithmische Graphentheorie**, Prof. Dr. Wolff

Notes:

- The focus of this course is on machine learning techniques for complex networks. While it will certainly help, the course does not require prior completion of our course *Statistical Network Analysis*. We assume that you have a basic understanding of key machine learning techniques, even though we provide a short introduction of key techniques within the course.
- Artificial intelligence and data science has become a strategic focus of the University of Würzburg. You will thus find a number of additional courses that naturally complement this lecture. Above we provide a selection of teaching offers at JMU that cover (general-purpose) AI and machine learning. You can also find complementary courses addressing topics like network analysis, graph theory and graph algorithms.
- Please check for additional courses as the list is constantly growing!

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Center for Artificial Intelligence and Data Science

CAIDAS

Artificial intelligence as a key technology with data as a resource for creating value in business and society

6	+	12	+	7	=	C
existing chairs with AI expertise (computer science , law, economics, philosophy)		additional AI chairs in the next months		(additional) chairs with AI application (economics, geography, biology, medicine, …)		

Fundamentals of Data Science and Machine Learning

Data Science for selected scientific fields

Transfer in Business and Society

images: pixabay.com

Notes:

- The rapid growth of courses on AI and data science is largely thanks to the recently founded Center for Artificial Intelligence and Data Science (CAIDAS), which includes several existing chairs as well as new professors who are currently being hired.
- Existing collaborations between computer science groups and groups in other fields like business, economics, biology, medicine, physics, or philology provide you with excellent opportunities to apply machine learning methods to practical problem from other disciplines.

Let's learn from each other!

Humboldtsches Bildungsideal

University education should not be job-focused, but educational training that is independent of economic interests. [...] University should be a place of [...] **exchange between all involved in the scientific process**. → Wikipedia

Wilhelm von Humboldt
1767 – 1835

- we need a **pleasant learning environment**
  - golden rule: there are **no** stupid questions
  - feel free to **ask anything at any time**
- **consultation hours** and **advice**
  - contact me in case of problems/concerns
  - 24/7 via E-Mail :)
  - personal meetings upon request

ingo.scholtes@uni-wuerzburg.de

Notes:

- An important rule for this course: you should not be afraid to ask questions, no matter how simple or “stupid” you think they are. There will be no laughing about you and there will be no public shaming. We will ensure that you can learn in an environment that is free of fear.
- Please consider posting your questions to the Q&A forum on WueCampus rather than sending them via E-Mail. This will allow other students to benefit from the answer to your question, and other students may have an answer even before I managed to reply.
- If you have problems that you prefer to not raise in the plenum, feel free to contact me via E-Mail or ask for an appointment for a personal meeting.

Self-study questions

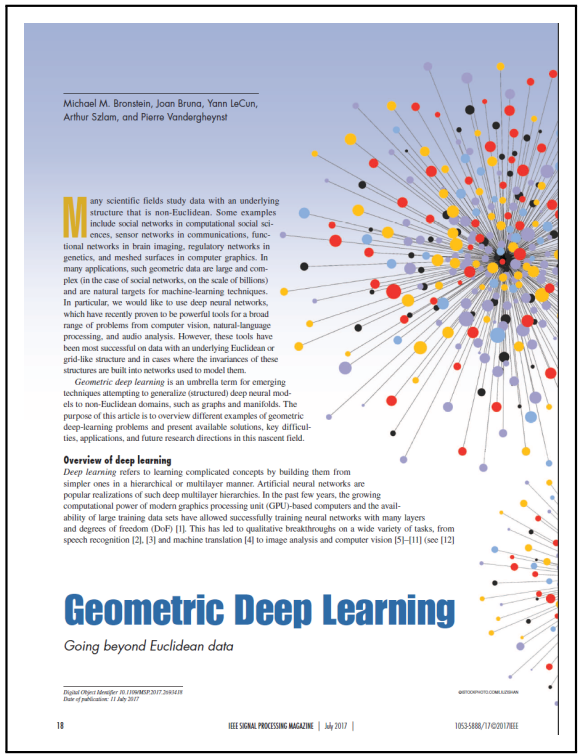
1. Give one example each for a system that can be modelled by a network with undirected or directed links.
2. Explain why we cannot directly apply standard machine learning techniques like k-means clustering or SVMs to data on networks.
3. Can you define a distance measure for the nodes in a graph that fulfills the properties of a metric?
4. Explain whether your definition from above works both for directed and undirected networks?
5. Give one example each for a problem that can be phrased as a node-, edge-, or graph-level machine learning task in a complex network.
6. Explain and exemplify three key challenges in the application of machine learning to graph-structured data.
7. Use the service `DBLP.org` to check proceedings of a recent machine learning conference (e.g. SIGKDD, NeurIPS or ICML). Use paper titles to identify works that address machine learning in networks.
8. Check the proceedings of the last Learning on Graphs conference (Log). Use paper titles to identify current research trends.

Notes:

References

reading list

- M Newman: **Networks**, Oxford University Press, 2010 → Chapter 1
- M Bronstein et al.: **Geometric Deep Learning: Boing beyond Euclidean data**, IEEE Signal Processing Magazine, July 2017
- Z Wu et al.: **MolecuNet: a bechmark for molecular machine learning**, Chemical Science, Vol. 9, 2018
- D Szklarczyk et al.: **STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental data sets**, Nucl. Acids Res., Vol. 47, 2019
- J Gilmer et al.: **Neural Message Passing for Quantum Chemistry**, Proc. of PMLR, 2017
- A Fout et al.: **Protein Interface Prediction using Graph Convolutional Networks**, Prof. of NeurIPS, 2017
- ZM Zhang, L Liang, L Liu, MJ Tang: **Graph Neural Networks and Their Current Applications in Bioinformatics**, Front. in Genetics, Vol. 12, 2021
- T Hamaguchi et al.: **Knowledge Transfer for Out-of-Knowledge-Base Entities: A Graph Neural Network Approach**, IJCAI 2017
- W Fan et al.: **Graph Neural Networks for Social Recommendation**, Proc. of WWW, 2019
- WL Chiang et al.: **Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks**, Proc. of SIGKDD, 2019
- Sakr et al.: **The Future is Big Graphs: A Community View on Graph Processing Systems**, ACM Comm., 2021



Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst

Many scientific fields study data with an underlying structure that is non-Euclidean. Some examples include social networks in computational social sciences, sensor networks in communications, functional networks in brain imaging, regulatory networks in genetics, and meshed surfaces in computer graphics. In many applications, such geometric data are large and complex (in the case of social networks, on the scale of billions) and are natural targets for machine-learning techniques. In particular, we would like to use deep neural networks, which have recently proven to be powerful tools for a broad range of problems from computer vision, natural-language processing, and audio analysis. However, these tools have been most successful on data with an underlying Euclidean or grid-like structure and in cases where the invariances of these structures are built into networks used to model them.

Geometric deep learning is an umbrella term for emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains, such as graphs and manifolds. The purpose of this article is to overview different examples of geometric deep-learning problems and present available solutions, key difficulties, applications, and future research directions in this nascent field.

Overview of deep learning

Deep learning refers to learning complicated concepts by building them from simpler ones in a hierarchical or multilayer manner. Artificial neural networks are popular realizations of such deep multilayer hierarchies. In the past few years, the growing computational power of modern graphics processing unit (GPU)-based computers and the availability of large training data sets have allowed successfully training neural networks with many layers and degrees of freedom (DoF) [1]. This has led to qualitative breakthroughs on a wide variety of tasks, from speech recognition [2], [3] and machine translation [4] to image analysis and computer vision [5]–[11] (see [12]

Geometric Deep Learning

Going beyond Euclidean data

Digital Object Identifier 10.1109/MSP.2017.2693418
Date of publication: 11 July 2017

Notes: