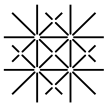


University of Basel

Application of Graph Learning to inverse problems

Master Thesis Preperation

Natural Science Faculty of the University of Basel
Department of Mathematics and Computer Science
Data-Analytics

Examiner: Prof. Dr. Ivan Dokmanić
Supervisor: Dr. Valentin Debarnot

Cédric Mendelin
cedric.mendelin@stud.unibas.ch
2014-469-274

29.11.2021

Table of Contents

1 Introduction

Inverse problems refer to the process, to derive an unknown system from observed data. In Machine Learning (ML), this system can be seen as the model, which produced the data. They are widely used throughout different science directions, such as ML, signal processing, computer vision, natural language processing and many more.
In recent years, Graphs got a lot of attention in ML and are one of the most promising research areas. Graphs are a well suited data structure, simple but with high expressiveness and, therefore, well suited in ML and for inverse problems. For some specific scenarios ordinary ML algorithm fail but Graph ML approach have great success, e.g dimensionality reduction for high-dimensional data. Data can be in a graph structure already, like social networks, or they can be constructed easily.
Cryo-electron microscopy (cryo-EM), where molecules are imaged in an electron microscope, gained a lot of attention in recent years. Due to ground-breaking improvements regarding hardware and data processing, the field of research has highly improved. In the year 2017, the pioneers in the field of cryo-EM even got the Nobel Prize in Chemistry[1] Today, using cryo-EM many molecular structures can be displayed with near-atomic resolution. The big challenge with cryo-EM is enormous noise, which makes calculation challenging, but more to that later in the report. During the Master Thesis, the aim is to exploit Graph Learning on the cryo-EM reconstruction problem.

The following report resulted as the Master Thesis Preparation report. During the six weeks project, the aim was to familiarize with the research area, build up some mathematically foundation needed for the Thesis and define the project content as well as a project plan.

The report is structured the following: In chapter 2, the overall foundation for the Master Thesis will be given, focusing on Graph Learning, Graph Denoising, some mathematically methods and definitions as well as a introduction to cryo-EM. Chapter 3 is dedicated to the problem setup and some preliminaries of the problem. Moreover, the base idea of the Master Thesis is defined. Up to this point, the underlying problem has been defined and

[1] https://www.nobelprize.org/prizes/chemistry/2017/press-release/

some related work can be given in chapter 4. To end the report, the project plan and some work packages are introduced in chapter 6

2
Foundation

Before dealing with the problem setup of the Master Thesis itself, in the following chapter, a broad foundation of graphs and Graph Learning will be given. Moreover, some important mathematically concepts and methods will be introduced as well as cryo-EM.

2.1 Graph Foundations

A graph is defined as $G = \langle V, E \rangle$, where V is a set of vertices (or nodes) and E is a set of edges (or links). Edges are defined as a set of tuples $\langle i, j \rangle$, where i and j determine the index of the vertices in the graph.

Edges of the graph can be either *directed* or *undirected* and has to do with the position of the nodes in the edge. In a directed graph, a edge points explicitly from one node to another, which means that edge $\langle i, j \rangle \neq \langle j, i \rangle$. In undirected graphs the ordering does not matter and $\langle i, j \rangle = \langle j, i \rangle$.

Moreover, edges can have weights, which is a method to define some kind of importance to the neighbours of a node. If edges are dealing with weights, we are talking from a *weighted* graph.

A *dense* graph is a graph, where the number of edges is close to the maximal number of edges. Contrarily, a *sparse* graph only consists of a few edges.

Homophilic and *heterophilic* are properties of the graph and are from importance in link predication. A homophilic dataset leads to the tendency to interact with similar nodes, therefore nodes will be linked with a edge. Whereas a heterophilic dataset contrary has the tendency to not link, if nodes are similar.

2.1.1 Important matrices

To do calculation with graphs, it is common to translate graphs in a well suitable mathematically form, which are matrices. In the following, some important matrices will be introduced.

Adjacency matrix: The (binary) adjacency matrix of graph G is defined as follows:

$$A_{ij} = \begin{cases} 1 & \text{if } \langle i, j \rangle \in E \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

The matrix A has dimension $\mathbb{R}^{N \times N}$ and the indices of the matrix correspond to the nodes of the graph. If there is an edge between two nodes, the entry in the matrix will be set to 1, otherwise to 0. This leads to an unweighted graph, as the weight of all edges will be 1. When the graph is undirected, the resulting matrix will be symmetric and has a complete set of eigenvalues and eigenvectors. The set of eigenvalues are also called *spectrum* of the graph.

K-hop neighbourhood: The adjacency matrix A has many nice properties, and one is referred to the k-hop neighbourhood. When calculating the k-th power of A, on calculates the k-hop neighbourhood of the graph. The resulting matrix gives the number of walks of length k from one node to another. Moreover, with $A[i]$, one can determine the k-hop neighbourhood of node i. Every node in the extracted row, which has a value > 0 can be reached within k hops.

Degree matrix: The degree matrix of G is defined as follows:

$$D_{ij} = \begin{cases} deg(v_i) & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

Where $deg(v_i)$ is the degree of the node, formally the number of incoming edges of node v_i.

Normalization: When starting calculating with matrix A, it is sometimes necessary to normalize. With the degree matrix D and adjacency matrix A, all information for normalization are present. The normalization can be achieved in a row, column or symmetric way:

$$A_{row-norm} = D^{-1}AA_{col-norm} \quad = AD^{-1}A_{sym} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \tag{2.3}$$

The normalization of the row and column will normalize the row and column to sum to 1 respectively. The symmetric normalization is well suited for undirected graphs, as it preserve the nice symmetric structure matrices.

Graph Laplacian The graph Laplacian is a matrix that represents the graph and can be used to find many important properties of the graph. It is defined as follows:

$$L = D - A \tag{2.4}$$

Normalized Graph Laplacian: During computation, it is often needed to have a normalized version of the Graph Laplacian:

$$L_{sym} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}L_{rw} \quad = I - D^{-1}A, \tag{2.5}$$

where L_{sym} is a symmetric normalization and L_{rw} is called a random walk normalization.

2.1.2 Graph Construction

When data is not available as a graph, it can be constructed from the data. First of all, every sample can be seen as a node and only the decision of how edges will be constructed is necessary. One popular approach is k-nearest neighbour (KNN) graph construction. The parameter k defines how many edges every node will have at the end. The neighbourhood of node i is defined as $\mathcal{N}_i$ and consists of the nodes, with the k smallest similarity measure.

$$A_{ij} = \begin{cases} 1 & \text{if } j \in \mathcal{N}_i \\ 0, & \text{otherwise} \end{cases} \tag{2.6}$$

Instead of using a fixed parameter k as in KNN approach, one could define a threshold π and connection nodes if the similarity measure is smaller than π. This is another common way to define graphs and results in a graph, where not all nodes have the same amount of edges.

2.2 Graph Denoising

Data acquired by Real-world observations are often noisy, which can lead to poor performance on data analysis tasks. This observed data can already be in the form of a graph, or a graph can be easily constructed. This resulting graph is what we call a noisy graph, as it includes the noise from the observation.
Graph denoising is the task to reconstruct the original graph from a noisy one. Therefore, graph denoising can be seen as a pre-processing step, where noisy data is filtered.
Denoising in general has often to do with averaging and graphs are a well suited data structure for this task[4].

Noise: A noisy observation is defined as: $y_n = y + \eta$, where η is the observation noise and y the noiseless observation.

Denoising: When we talk from denoising, we want to reconstruct the true observation from a given noisy observation. This reconstruction is done via averaging, which can be performed locally, by the calculus of variations or in the frequency domain[4].

Noisy Graph : For every noisy graph, there exists an original graph $G = \langle V, E \rangle$.
The noisy graph can be defined as follows:

$$\begin{aligned} G_{noisy} &= \langle V, E_{noisy} \rangle, \\ \text{with } E_{noisy} &= E \setminus E^- \cup E^+, \\ E^- &\subseteq E, \\ E^+ \cap E &= \emptyset \end{aligned} \tag{2.7}$$

The noisy graph consists of the same vertices as the original graph. From the original graphs edges, some are removed (denoted by E^-) and some new edges are added (denoted by E^+).

The adjacency matrix of G_{noisy} is denoted by $\bar{A}_{ij}$. The task of graph denoising, can therefore be written as:

$$\bar{A} \xrightarrow[method]{Graph-denoising} \tilde{A} \approx A \tag{2.8}$$

Where $\bar{A}$, $\tilde{A}$, A denotes the adjacency matrix from the noisy input graph, the denoised graph and the original graph respectively.

Connection to link prediction Link prediction is a task in Graph Learning. The idea is to predict existence of a link (edge) between two nodes. The task can be formulated as a missing value estimation task. A model M_p is learned from a given set of observed edges. The model finally maps links to probabilities:

$$M_p : E' \rightarrow [0, 1], \tag{2.9}$$

where E' is the set of potential links.
We define U as the set of all possible vertices of G, therefore $E \subseteq U$. Obviously, graph denoising can be seen as a link prediction problem.
The difference is, that in link prediction a model from a set of observed links is learned $E_{observed} \subseteq E$ and in graph denoising the model is learned from $E_{observed} \subseteq U$.
On could also say that link prediction problems are a subset of graph denoising problems.

2.3 Math Foundation

In the following section, some mathematically concepts and methods will be explained.

2.3.1 Embedding

Mathematically, an embedding $f : X \rightarrow Y$ is defined as a structure-preserving mapping from one domain to another.
In graph theory, a graph embedding in the mapping from the graph G to a surface structure Σ.

2.3.2 Manifolds

A manifold is a topological space, where locally Euclidean distances make sense. More formally, a n-dimensional manifold is a topological space where each point has e neighbourhood, that is homeomorphic (mapping which preserves topological properties) to an subset of a n-dimensional Euclidean space.
Some example for a 1-D manifold is a line or a circle. 2-D manifolds can already become pretty complex and are basically any surfaces like planes, sphere but also the torus, Klein bottle or others.

Manifold assumption: The manifold assumption is a popular assumption for high-dimensional datasets. Even if a given dataset is in high-dimension and consists of many features, one can assume, that these data points are samples from a low-dimensional manifold, which embeds the high-dimensional space.

Therefore, if one can approximate the underlying Manifold, one solved the dimensionality reduction as one can embed the data points in the low-dimensional manifold space.
There is a complete area of research devoted to this manifold assumption called Manifold Learning[5].

2.3.3 Power Iterations

Power iteration (also called power method) is a iteratively method, which approximates the biggest eigenvalue of a diagonalizable matrix A.
The algorithm starts with a random vector b_0 or an approximation of the dominant eigenvector.

$$b_{k+1} = \frac{Ab_k}{||Ab_k||} \tag{2.10}$$

The algorithm not necessarily converges. The algorithm will converge, if A has an eigenvalue strictly grater than its other eigenvalues and the initial vector b_0 has a component in direction of an eigenvector, associated with the dominant eigenvector.

2.3.4 Folded spectrum Method

Calculation of eigenvalues and eigenvectors of a given Hamiltonian matrix H is a fundamental mathematical problem. Often, we are interested in just the smallest values, which can be efficiently computed. But if we are interested in selected values, this can be hard. H is needed to be diagonalized (bring matrix H into diagonal form) which is computationally expensive and for big matrices impossible.
Currently, the best way to solve such problems is the Folded spectrum (FS)[23] method, which iteratively solves the problem. During calculation, the eigenvalue spectrum will be folded around a reference value ϵ.

$$v^{t+1} = v^t - \alpha(H - \epsilon I)^2 v^t, \tag{2.11}$$

with $0 < \alpha < 1$. When $t \to \infty$, then v^∞ will be the eigenvector with respect to the reference value ϵ.

2.3.5 Wasserstein metric

The Wasserstein metric is a distance measure between two probability distributions and it is used in ML as a loss function[10]. Intuitively, it can can be understood as the minimum cost to transfer the mass of one distribution to the other. Therefore, it is also known as the *earth mover's distance*.
As Arjovsky et al. [1] could show, ordinary distance measures like *Total Variation*, *Kullback-Leibler divergence* and *Jensen-Shannon divergence* are not sensible when learning with distributions supported by manifolds On the contrary, Wasserstein metric does a good job as loss function in such scenarios.

2.3.6 Fourier Transform

Fourier Analysis is the overall field of study, which deals with representing (or approximating) functions as sums of trigonometric functions. When the function is defined in such a way, we are talking from the *Fourier Domain*.
Fourier transform (FT) is the way of transforming signals to the Fourier Domain, which is popular in ML. Basically, with the Fourier transform, a signal can be decomposed to a *Fourier series*, which consists of many weighted sinusoids.

2.3.6.1 Fourier-slice theorem

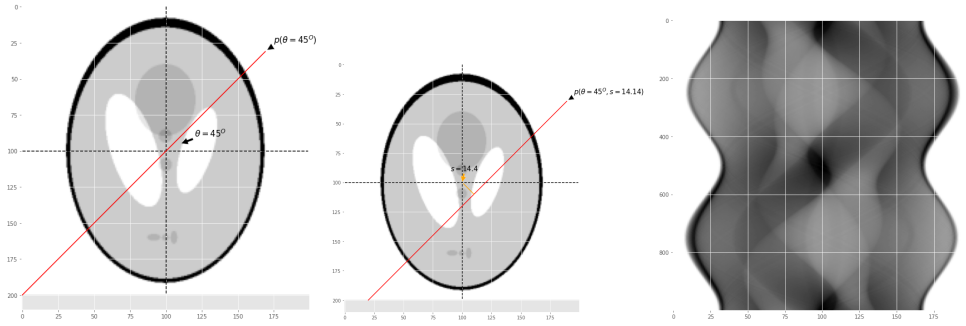
The Fourier-slice theorem [14] in 3D is defined as follows:

$$F_2 P_2 = S_2 F_3, \tag{2.12}$$

where $F2$ and $F3$ are FTs in 2D and 3D respectively, P2 is a projection operator ($P_2 : 3D \rightarrow 2D$) and $S2$ is the restriction operator.
As pointed out by [3], the Fourier-slice theorem is the foundation of the reconstruction problem in computerized tomography (CT), which will be explained in section 3.1. It states, that the 2D FT of the tomographic projection is the same as the 3D FT restricted to a 2D plane through the origin. Basically, for the CT reconstruction problem, acquiring samples from known viewing directions is the same as sampling the 3D Fourier-space. This concept is exploited by the filter BackProjection algorithms, see section 3.1.2.

2.3.7 Radon Transform

The radon transform[21] is the main mathematically concept of tomographic reconstruction. It is an integral transformation of a function $f(x, y)$, which is defined on the plane. In tomographic reconstruction the function f will be the observed tomographic image. The radon transform then transforms f to a function Rf, which corresponds to the line integral of the line defined by the two parameters θ and s, where θ is a angle and s the distance to the origin.
In Figure 2.1(a) and Figure 2.1(b) on can see two plots of different values for θ and s, where $f(x, y)$ is the Shepp-Logan phantom. The complete $Rf(\theta = 45, s = 0)$, which is also called *sinogram*, can be see in Figure 2.1(c)



(a) Radon Transform $Rf(\theta = 45, s = 0)$

(b) Radon Transform $Rf(\theta = 45, s = 14.14)$

(c) Shepp–Logan phantom sinogram of $Rf(\theta = 45, s = 0)$

Figure 2.1: Examples, where the original object x is the Shepp-Logan phantom.

2.4 Graph Learning

As already mentioned, Graph Learning is a popular research area and got a lot of attention in recent years. It is a new way of applying Machine Learning (ML) with graphs as a data structure and many algorithms emerged from ML.
A lot of real data can be modelled as graphs. The data could have graph structure, like social networks. Or a graph can be artificially constructed with methods like k-nearest neighbours (KNN) or with some other similarity measure.

Graph Learning Tasks: When a graph is available, one can start using Graph Learning algorithms for solving tasks. Popular tasks are node classification or link prediction within a graph. One tries to learn from node and edge features as well as the topology of the graph and tries to map information to a model, which allows prediction or classification.
Another popular task in Graph Learning is community detection, where the aim is to identify cluster of nodes within the input graph.
Further, graphs are highly popular for dimensionality-reduction. In higher dimensions, the euclidean distance is not helpful and therefore, algorithms for reducing the dimensionality, such that euclidean distance make sense again. are needed. Graph algorithms provide a helpful tool in such scenarios, as ordinary algorithms like principle component analysis (PCA) fail.

Algorithm categories There are many algorithmic approaches, how to exploit graphs.
Graph Deep Learning is a derivation from Deep Learning. Basically, in Graph Deep Learning, Deep Learning algorithms are extended for the usage with graphs. After all, a model or some feature will be learned within a neural network, suitable for working with graphs.
Spectral graph theory[19] deals with learning properties and characteristics of graphs, in regard to the graphs eigenvalues and eigenvectors.
Manifold Learning [5] is a popular approach for dimensionality reduction on graphs. Using the manifold assumption section 2.3.2, an embedding of the graph for lower dimension is calculated, which can preserve most of the information, of the original graph.

Further, *Random Walks* is a concept, which is often used in Graph Learning. It is used to exploit topological information of a graph by randomly "walking" (use edges to move from on node to another) over the graph. With sampling a lot of these walks, one can infer information about the graphs topology.

2.5 Cryo-EM

TODO

3

Preliminaries and Problem Setup

In the following chapter, the problem setup handled by the Master Thesis will be explained. Further, preliminaries regarding assumptions and other decisions are defined.

3.1 Reconstruction problem

Tomographic reconstruction is a popular inverse problem [6]. The aim is to reconstruct an object x from its observed projections $P = [p_0, p_1, \ldots, p_N]$. More formally, the aim is to recover some density function f from overserved samples y, taken from the line-integral $p(\cdot)$. The problem can be defined as a two-dimensional (2D) problem but also as a three-dimensional (3D) problem. In the following, we focus on the 2D case. In 2D, also called classical tomography reconstruct problem, the underlying density function is in two dimensions and the measurements lines lie on a plane.
The problem automatically gets harder, if we deal with incomplete datasets (subset of measured lines, limited angle data) but also with noisy observations. Moreover, the angle θ of the projections are not always known.

3.1.1 Computer tomography

First of all, lets define the line integral p of our unknown density function f in the 2D case:

$$\begin{aligned} p(\theta, s) &= Rf(\theta, s) \\ Rf(\theta, s) &= \int_{-\infty}^{\infty} f(x(z), y(z))dz \\ &= \int_{-\infty}^{\infty} f((z\sin\theta + s\cos\theta), (-z\cos\theta + s\sin\theta))dz \end{aligned} \tag{3.1}$$

where p is the line integral of the density function f, θ the projection angle and s the distance from the origin.
In the 2D case, the line integral corresponds to the Radon-Transform [21]. When data of the line integral is presented as a 2D image, we are speaking from the *sinogram*. With the 2D Radon transform, we can map the density function f to the sinogram p.

3.1.2 Filter Backprojection

Filter Backprojection (FBP) is a reconstruction method, which allows to solve for p. It is equivalent to the inverse of the Radon Transform and is related to the Fourrier transform. Basically, it maps sinograms of p back to the density function f.
The problem with the algorithm is, that it only works for complete data and without noise.

3.2 Cryo-EM

3.2.1 Single particle multireference alignment (MRA)

3.2.2 Extension for PSF

Why exactly? Convolving (point spread function) PSF with observation.

3.3 General form

In the following, we want to defined a form, which is valid for the 2D reconstruction problem of CT as well as the 3D cryo-EM problem.

$$y_i = g_i A(x, \theta) + noise \tag{3.2}$$

where y_i is the observed sample, x our original object and $A(x, \theta)$ a non-linear operator.

3.3.1 Extension

Again, we can extend the formula to:

$$y_i = h_i \times g_i A(x, \theta) + noise \tag{3.3}$$

3.3.2 2D example

$a_i \in SO(1)$ $A := N \times N$ $B := N$
$f(x) = \int_{\theta=0}^{\pi} p(\theta, s)|_{s=x\cdot(-\sin\theta, cos\theta)} d\theta$

3.3.3 3D example

$a_i \in SO(2)$ $A := N \times N \times N$ $B := N \times N$

3.4 Manifold assumption

In the reconstruct problem, we can apply the Manifold assumption from section 2.3.2. Moreover, in the none-noisy case, we can even assume how this Manifold look like.
The manifold, and therefore, a low-dimensional embedding, can be calculated the following:

1. Construct the knn-graph from our line integral (sinogram).
2. Calculate the normalized Graph Laplacian

3. Extract the second, third (and fourth) smallest eigenvectors

todo: Show with Shepp-Logan-phantom that we get unit-cirlce
The showed example can be extended to 3D, where the underlying manifold corresponds to the Sphere. Therefore, we can derive, that our angles have the following property: In the 2D case, $\theta \in SO(1)$ and the in 3D case, $theta \in SO(2)$.
Again, the circle and sphere can be computed and for the none-noisy, the underlying Manifold can be seen as known.

3.5 Thesis problem

During the Master Thesis, the reconstruction problem with unknown angles is considered. Moreover, the observed samples are considered to be noisy.
The resulting proposed algorithm should work in the 2D and 3D scenario.
The main idea is, to exploit the fact that the underlying Manifold is known (Circle in 2D and Sphere in 3D).
From our noisy observations, we can compute the approximated manifold and compare it with the original manifold. The comparison between the manifold enables the possibility of a Loss function and Learning in general.
Simple Case in

4
Related Work

In the following section, related work will be introduced.

4.1 Graph Deep Learning

Graph deep learning is a fast evolving field in research. With Graph Neural Networks (GNN) [11] [22]

4.1.1 Graph feature extraction with GCN

4.1.2 Graph Convolutional Network

Graph Convolutional Networks (GCN) [12] can be used for many tasks in the field of Graph Learning, such as node classification or link prediction. Basically, with GCN, a new feature representation is iteratively learned for the node features.
The basic concept is as follows: For a given graph $G = \langle V, E \rangle$, with node features X^{NxD} and adjacency Matrix A where N denotes the number of nodes and D the number of node input attributes,
a novel node representation Z^{NxF} will be learned, where F is the number of output features. Z will be learned within a neural network, and every layer can be written by the following, non-linear function:

$$\begin{aligned} H^{l+1} &= f(H^l, A), \\ \text{with } H^0 &= X, \\ H^L &= Z, \end{aligned} \tag{4.1}$$

where L is the number of layers in the neural network. The model only differ in the choice of $f(\cdot, \cdot)$.
We are ready to define our first GCN. To keep it simple, $f(\cdot, \cdot)$ will be defined as the following:

$$f(H^l, A) = \sigma(AH^lW^l) \tag{4.2}$$

Where $\sigma(\cdot)$ is a non-linear activation function, such as ReLU and W^l is a weight Matrix of the layer l of the neural network. As Kipf and Welling [12] could show during experiments,

this choice of $f(\cdot,\cdot)$ is already very powerful and leads to state-of-the-art results.

4.1.2.1 Renormalization trick

With this model, we do have two problems and need to refine it further. First of all, with the multiplication of A, we average over the neighbour nodes but will ignore the node itself. Therefore, self-loops will be added to A. The second problem is, that A is not normalized and if therefore, when multiplying with A, the features of the nodes will change it scale. Therefore, we need to normalize A such that all rows sum to one. This can be done with a simple multiplication with the D.
These two steps are called the Renormalization trick[12]. First of all, we can simple add the self-loops by adding the Identity Matrix to A, $\hat{A} = A + I$ and $\hat{D}$ is the degree Matrix of $\hat{A}$. Now, we can achieve a symmetric normalization by multiplying $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$.
And finally, we can put all things together, and replace A in the original equation:

$$f(H^l, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^l W^l) \tag{4.3}$$

4.1.2.2 Simple Graph Convolutional Network

Basically Power method with normalization
Simple Graph Convolutional Network (SGC) [25] proposed a simplified version of GCN. They could verify their hypothesis, that GCN is dominated by the local averaging step and the non-linear activation function between layers do not contribute to much to the success of GCN.
This makes the calculation simpler. We denote $S = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ and can use the fact that in every layer of the neural network, the same computation will take place.

$$\begin{aligned} Z &= S \dots SXW^1W^2 \dots W^L \\ Z &= S^L XW^1W^2 \dots W^L \\ Z &= S^L XW \end{aligned} \tag{4.4}$$

where W is the matrix of all vector weights.

4.1.2.3 Link to Graph Laplacian:

In the section, we will have a look at the connection between SGC and Graph Laplacian.
We can define $x \in R^n$ as our signals and define the Fourier transform as $\hat{x} = U^T x$ and the inverse as $x = U\hat{x}$. With the transform, we can easily switch between spatial and Fourier(spectral) domain.
Further, we can define the graph convolution operation between signal x and filter g.

$$g \star x = U((U^T g)(U^T x)) = U\hat{G}U^T x, \tag{4.5}$$

where $\hat{G}$ is a diagonal matrix where the elements are the spectral filter coefficients (eigenvalues?)
The graph convolution can be approximated by the k-th order polynomials of Laplacians:

$$\approx \sum_{i=0}^{k} \Delta^i x = U \left(\sum_{i=0}^{k} \Theta_i \Lambda^i \right) U^T x, \tag{4.6}$$

where $\Delta = D - A$ and Θ_i are filter coefficients which correspond to polynomials of the Laplacian eigenvalues, $\hat{G} = \sum_i \Theta_i \Lambda^i$
In the original [12] paper, the approximation is done with $k = 1$

$$g \star x = \Theta(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x, \tag{4.7}$$

, where Kipf and Welling [12] further applies the renormalization trick, ending up replacing $I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ with $\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}}$.
$I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is also called first-order Chebyshev filter.
[12] [25] [24]

4.2 Manifold Learning

[20] [16] [17]

4.3 Random Walk approaches

[7] [18] [9]

4.4 Denoising

4.4.1 Image Denoising

4.4.1.1 Non local means

Non local means is a state-of-the-art image denoising method [4]. In the name of the method are two important concepts, namely the *mean* and *non local*.
For a given noisy image v, the denoised image is defined as:

$$NL[v](i) = \sum w(i,j)\ v(j) \tag{4.8}$$

where $w(i,j)$ is the weight between pixel i and j and fulfils two conditions:

- $0 \leq w(i,j) \leq 1$
- $\sum_j w(i,j) = 1$

The weight can be seen as a similarity measure of the two pixels. Moreover, these similarities are calculated over square neighbourhoods of the two pixels, where the l2 norm of the neighbourhood is used. Similar pixel neighbourhoods have a large weight and different neighbourhoods have a small weight.
More general, the denoised image pixel i is computed as an weighted average of all pixels in the image, therefore, in a non local way. Image Denoising Graph Denoising

4.4.2 Graph Denoising

[4] [13]

4.4.3 cryo-EM calculation

4.5 Graph Laplacian Tomography From Unknown Random Projections

Coifman et al. [8] introduces a Laplacian-based algorithm, with which reconstruction of a planar object from projects at random unknown directions is possible.
Overall, in computerized tomography (CT), reconstruction of an object with only samples of its projections is a standard problem. In Coifman et al. [8] the problem was extended by the fact, that the projection angle to the object is unknown.
Formally: Given N projection vectors $(P_{\theta_i}(t_1), P_{\theta_i}(t_2), \ldots, P_{\theta_i}(t_N)$ at unknown angles $\{\Theta_i\}_{i=1}^N$ which are drawn from the uniform distribution of $[0, 2\pi]$ and $t_1, t_2, \ldots, t_n$ are fixed n points (all equally spaced due to uniform distribution) find the underlying density function $\rho(x, y)$ of the object.

4.5.1 Radon transform

The radon transform $P_\Theta(t)$ is the line integral of ρ along parallel lines L at angle Θ and distance t from the orign.

$$\begin{aligned} P_\Theta(t) &= \int_L \rho(x, y) ds \\ &= \int_{-\infty}^{\infty} \rho(x, y)\ \delta(x \cos\Theta + y \sin\Theta - t) dx\ dy \end{aligned} \tag{4.9}$$

An algorithm for estimating angles from given projections have been introduced by [2]. The introduced algorithm consists of three steps:

1. Angle estimation
2. Angle Ordering
3. Joint maximum likelihood refinement of angles and shifts

Step 2 was implemented by some nearest neighbour algorithm. In the work of [8], they introduced a new way of ordering the angles, using Graph Laplacian.

4.5.2 Laplace-Beltrami operator

[8] could show, that the graph Laplacian approximates the Laplace-Beltrami operator, if data points are uniformly distributed over the manifold.
Further, they showed that in the case of non-uniformly distributed data points, the Laplacian approximated the backward Fokker-Planck operator (which is a generalization of the Laplace-Beltrami operator). With that, at least the ordering of the angles can be estimated. Finally, with a small normalization of the Laplacian, the Laplace-Beltrami operator can also be approximated in the non-uniform distributed case.

4.5.3 Algorithm

For a given set of projections vector $x_i = (P_{\Theta_i}(t_1), \ldots, P_{\Theta_i}(t_n))$ for $i = 1, 2, \ldots, mN$
The algorithm proposed in [8] consists of five steps:

1. Double the number of projections to 2mN (due to the fact that projections are symmetric)
2. Construct the co-called density invariant Graph Laplacian $\tilde{L}$
3. Compute $\theta_1(i)$ and $\theta_2(i)$ the first two nontrivial eigenvector of $\tilde{L}$
4. Sort x_i according to $\phi_i = \tan^{-1}(\theta_1(i) \ / \ \theta_2(i))$
5. Reconstruct image using the sorted projections and estimated angles.

Where $\tilde{L}$ can be constructed by the following way:

$$W_{ij} = k\left(\frac{\|x_i - x_j\|^2}{2\epsilon}\right), \quad i,j = 1,\ldots,N \tag{4.10}$$

where $||\cdot||$ is the euclidean-norm, k a semi-positive kernel and $\epsilon > 0$ the bandwidth of the kernel. As mentioned in [8], the kernel $k(x) = \exp(-x)$ is a popular choice.
With the newly computed weight Matrix W and the degree Matrix D corresponding to W, we can finally define $\tilde{L}$.

$$\begin{aligned} \tilde{W} &= D^{-1}WD^{-1} \\ \tilde{D} &= \text{Degree matrix corresponding to } \tilde{W} \\ \tilde{L} &= \tilde{D}^{-1}\tilde{W} - I \end{aligned} \tag{4.11}$$

4.5.4 Walk Pooling

Homophilic and heterophilic are properties of the underlying dataset in link predication. A homophilic dataset leads to the tendency to interact with similar nodes. Whereas a heterophilic dataset contrary has the tendency to not link, if nodes are similar.
Pan et al. [15] proposed a new way of link prediction, which works with a random-walk-based pooling method called WalkPool.
Based on the assumption, that link presence can be predicted only based on the information on its neighbours within a small radius k, they calculate subgraphs and extract information from these subgraphs.

4.5.4.1 Feature extraction

Based on the adjacency matrix and some node attributes, feature will be extracted with a GNN f_θ

$$Z = f_\theta(A, X) \tag{4.12}$$

This step could be achieved by the earlier introduced GCN.

4.5.4.2 Subgraph classification

For candidate link E^c, the k-hop enclosing subgraphs will be constructed, which allows the transformation from a link predication problem to a graph classification problem.
From these subgraphs, so called random-walk profiles are calculated:

4.5.4.3 Random-walk profiles

First of all, the node correlation is calculated with two multilayer perceptrons (MLP)

$$w_{x,y} = \frac{Q_\theta(z_x)^T K_\theta(z_y)}{\sqrt{F''}} \tag{4.13}$$

Then, from all neighbouring nodes ($N(x)$), probabilities are calculated:

$$p_{x,y} = \begin{cases} [softmax(w_{x,z})_{z \in N(x)}]_y & \text{if } \langle x, y \rangle \in E \\ 0, & \text{otherwise} \end{cases} \tag{4.14}$$

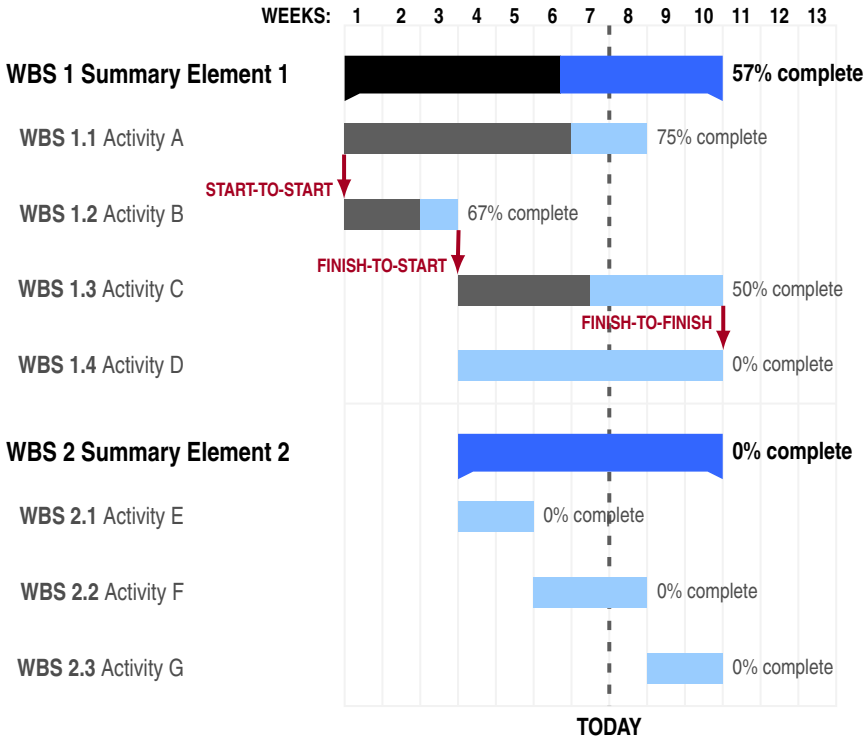
From these calculated probabilities P and its powers, we can derive information for graph classification:
Node level features $node^\tau$ describe loop structures around the candidate link. Further, link features $link^\tau$ give a probability, about a random walk with length τ from the two nodes (ending in a loop). And finally, graph features $graph^\tau$ are related to the total probability of length τ loops.
TODO: As we do not know about if the candidate link is present of not, we consider the subgraphs, consisting of the candidate link and without the candidate link.
We finally can describe WalkPool as:

$$WP_\theta(G, Z) = w_{1,2}(node^{\tau,+}, node^{\tau,-}, link^{\tau,+}, link^{\tau,-}, \delta graph^\tau)_{\tau=2}^{\tau_c} \tag{4.15}$$

5
Evaluation

Create artificial data to test algortihm and add hand crafted noise.
Use CT tomography for simple 2D tomography case. Shepp-Logan phantom.
Long view, application to cryoEM, but probably not possible during MS Thesis.
Baseline Papers
Multifrequency Vector diffusion maps [9]
and [8]

6
Project Plan


WEEKS: 1 2 3 4 5 6 7 8 9 10 11 12 13
WBS 1 Summary Element 1
57% complete
WBS 1.1 Activity A
75% complete
START-TO-START
WBS 1.2 Activity B
67% complete
FINISH-TO-START
WBS 1.3 Activity C
50% complete
FINISH-TO-FINISH
WBS 1.4 Activity D
0% complete
WBS 2 Summary Element 2
0% complete
WBS 2.1 Activity E
0% complete
WBS 2.2 Activity F
0% complete
WBS 2.3 Activity G
0% complete
TODAY

Bibliography

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/arjovsky17a.html.

[2] Samit Basu and Yoram Bresler. Feasibility of tomography with unknown view angles. *IEEE Transactions on Image Processing*, 9(6):1107–1122, 2000.

[3] Tamir Bendory, Alberto Bartesaghi, and Amit Singer. Single-particle cryo-electron microscopy: Mathematical theory, computational challenges, and opportunities. *IEEE signal processing magazine*, 37(2):58–76, 2020.

[4] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

[5] Lawrence Cayton. Algorithms for manifold learning. *Univ. of California at San Diego Tech. Rep*, 12(1-17):1, 2005.

[6] Rolf Clackdoyle and Michel Defrise. Tomographic reconstruction in the 21st century. *IEEE Signal Processing Magazine*, 27(4):60–80, 2010.

[7] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[8] Ronald R Coifman, Yoel Shkolnisky, Fred J Sigworth, and Amit Singer. Graph laplacian tomography from unknown random projections. *IEEE Transactions on Image Processing*, 17(10):1891–1899, 2008.

[9] Yifeng Fan and Zhizhen Zhao. Multi-frequency vector diffusion maps. In *International Conference on Machine Learning*, pages 1843–1852. PMLR, 2019.

[10] Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya-Polo, and Tomaso Poggio. Learning with a wasserstein loss. *arXiv preprint arXiv:1506.05439*, 2015.

[11] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[12] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[13] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 779–787, 2021.

[14] Frank Natterer. *The mathematics of computerized tomography*. SIAM, 2001.

[15] Liming Pan, Cheng Shi, and Ivan Dokmanić. Neural link prediction with walk pooling. *arXiv preprint arXiv:2110.04375*, 2021.

[16] Lawrence K Saul and Sam T Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Departmental Papers (CIS)*, page 12, 2003.

[17] Lawrence K Saul and Sam T Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Departmental Papers (CIS)*, page 12, 2003.

[18] Amit Singer and H-T Wu. Vector diffusion maps and the connection laplacian. *Communications on pure and applied mathematics*, 65(8):1067–1144, 2012.

[19] Daniel Spielman. Spectral graph theory. *Combinatorial scientific computing*, 18, 2012.

[20] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[21] Peter Toft. The radon transform. *Theory and Implementation (Ph. D. Dissertation)(Copenhagen: Technical University of Denmark)*, 1996.

[22] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[23] Lin-Wang Wang and Alex Zunger. Solving schrödinger's equation around a desired energy: Application to silicon quantum dots. *The Journal of Chemical Physics*, 100(3): 2394–2397, 1994.

[24] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[25] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.