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

Webpage

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
Assessment criteria

Written report including:

- Contents of the Master's Thesis project

- Project plan

- summary of relevant related work

2

Questions

Organization

- Draft and switch of meeting.

- Presentation, date and content?

- Contract of Thesis

- Project plan

Overall problem:

2.0.1 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. [11] 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.

2.0.1.1 Feature extraction

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

$$Z = f_\theta(A, X) \tag{2.1}$$

This step could be achieved by the earlier introduced GCN.

2.0.1.2 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:

2.0.1.3 Random-walk profiles

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

$$w_{x,y} = \frac{Q_\theta(z_x)^T K_\theta(z_y)}{\sqrt{F''}} \tag{2.2}$$

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

$$p_{x,y} = \begin{cases} [softmax(w_{x,z})_{z \in N(x)}]_y & \text{if } \langle x, y \rangle \in E \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

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:

$$WP_\theta(G, Z) = w_{1,2}(node^{\tau,+}, node^{\tau,-}, link^{\tau,+}, link^{\tau,-}, \delta graph^\tau)_{\tau=2}^{\tau_c} \tag{2.4}$$

3

Paper notes

3.1 Diffusion Maps:

Coifman and Lafon [5] [5]

Dimensionality reduction: In essence, the goal is to change the representation of data sets, originally in a form involving a large number of variables, into a low-dimensional description using only a small number of free parameters.

meaningful structures in data sets: Analogous to the problem of dimensionality reduction is that of finding meaningful structures in data sets. The idea here is slightly different and the goal is to extract relevant features out of the data in order to gain insight and understanding of the phenomenon that generated the data.

Markov Chain:

Random walk:

PageRank: Stationary distribution of random walk

Kernel eigenmap methods: - local linear embedding - Laplacian eigenmaps, - hessian eigenmaps - local tangent space alignment

The remarkable idea emerging from these papers is that eigenvectors of Markov matrices can be thought of as coordinates on the data set. Therefore, the data, originally modelled as a graph, can be represented (embedded) as a cloud of points in a Euclidean space. two major advantages over classical dimensionality reduction (PCA, MDS): The first aspect is essential as most of the time, in their original form, the data points do not lie on linear manifolds. The second point is the expression of the fact that in many applications, distances of points that are far apart are meaningless, and therefore need not be preserved.

Unnormalized Graph Laplacian: $L = D - W$

Normalized Graph Laplacian construction: $L_{sym} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2}$
$L_{rw} = D^{-1}L = I - D^{-1}W$

Markov chain has a stationary distribution. If graph is connected, stationary is unique. If X is finite, is ergodic.

Diffusion distance: Diffusion map ψ embeds the data into the Euclidean space so that in this space, the Euclidean distance is equal to the diffusion distance.

Laplace–Beltrami operator on manifolds

What are diffusion maps

3.1.1   Vector Diffusion Maps (VDM)

[14] VDMis a mathematical and algorithmic generalization of diffusion maps and other non-linear dimensionality reduction methods, such as LLE, ISOMAP, and Laplacian eigenmaps. While existing methods are either directly or indirectly related to the heat kernel for functions over the data, VDM is based on the heat kernel for vector fields.

Main concept: Edge consists of weight and linear orthogonal transformation. If linear orthogonal transformation is big, nodes are more like to be equal. If small, there are different Diffusion is calculated on vectors fields, where tangents are mapped to the manifold. A way to globally connect Local PCAs.

SNR: signal-to-noise-ratio

LLE: ISOMAP: Laplacian eigenmaps:

3.1.2   Riemannian Manifold Assumption:

One of the main objectives in the analysis of a high-dimensional large data set is to learn its geometric and topological structure. Even though the data itself is parametrized as a point cloud in a high-dimensional ambient space R^p, the correlation between parameters often suggests the popular "manifold assumption" that the data points are distributed on (or near) a single low-dimensional Riemannian manifold Md embedded in Rp, where d is the dimension of the manifold and $d << p$.

3.1.3   Multi-Frequency Vector Diffusion Maps (MFVDM)

[7] **For a direct link between manifold embedding and tomography, very close to what Ivan explained this morning. If we have a graph denoising method, we will need to compare with this approach (or the original vector diffusion maps).** Basically same as VDM, but with multiple frequencies per edge.

Diffusion maps (DM) only consider scalar weights over the edges and the vector diffusion maps (VDM) only take into account consistencies of the transformations along connected edges using only one representation of $SO(2)$, i.e. $e^{ia_{i,j}}$. In this paper, we generalize VDM and use not only one irreducible representation, i.e. $k = 1$, but also higher order k up to k_{max}.

3.2   Learning to Drop

Graph denoising

3.3   Point Clouds

3.3.1   Dynamic graph Cnn for learning on point clouds

One of the few reference related to graph neural network and learning of graph structure. Adjusts the graph topology layer by layer.

3.3.2   CryoEm and related

2. Estimation of Orientation and Camera Parameters from Cryo-Electron Microscopy Images with Variational Autoencoders and Generative Adversarial:
learning framework where the manifold embedding is estimated.

3. Computational Methods for Single-Particle Cryo-EM: review around cryo-EM.
This reference doesn't talk about manifold embedding, but it is a nice one if you want to know more about the acquisition system and standard approaches to solve the cryo-EM problem.

3.bis) Single-Particle Cryo-Electron Microscopy: another review similar to the previous one. The section "Mathematical frameworks for cryo-EM data analysis" and especially the subsection MRA (multireference alignement) introduce a toy model that is related to cryo-EM and where the symmetries are of importance.

4. Bispectrum Inversion with Application to Multireference Alignment: for a paper that introduce several algorithms to solve MRA.

4

Introduction

Learning on Graphs is one of the most promising research area in Machine Learning. In the past years, a lot of new encouraging methods have been developed, e.g Graph Deep Learning.

Cryo-electron microscopy (EM)

Inverse problems. **TODO**

5

Foundation

Before digging into the problem setup of the Master Thesis, in the following chapter, a broad foundation of graphs and Graph Learning will be given. Moreover, some important mathematically concepts will be introduced. A more detailed explanation to some sections will follow in the Section Related Work

5.1 Graph Learning

As already mentioned, Graph Learning is a highly popular research area and got lot of attention in recent years. It is a new way of applying Machine Learning (ML) and many algorithms emerged from ML.

A lot of real data can be modelled as a graph. The data could have a 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 concept.

5.1.1 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 properties as well as the topology of the graph and tries to map the 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.

5.1.2 Algorithm categories

5.1.2.1 Spectral graph theory

Spectral graph theory [15] deals with learning properties and characteristics of graphs, in regard to the graphs eigenvalues and eigenvectors.

5.1.2.2 Graph 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.

5.1.2.3 Random walk based methods

TODO

5.2 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.

5.2.1 Graph Properties

5.2.1.1 Directed vs. undirected vs. weighted

5.2.1.2 Dense and sparse Graph

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

5.2.2 Node Properties

5.2.3 Edge Properties

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.

5.2.4 Adjacency Matrix

The adjacency matrix of G is then defined as follows:

$$A_{ij} = \begin{cases} 1 & \text{if } \langle i, j \rangle \in E \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

5.2.4.1 k-hop neighbourhood

The adjacency matrix A has a nice property. When calculating the k-th power of A^k, on calculates the k-hop neighbourhood of the given matrix. Which can be used for many

application.

5.2.5   Degree Matrix

The degree Matrix of G is defined as follows:

$$D_{ij} = \begin{cases} deg(v_i) & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

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

5.2.5.1   Normalization

When starting calculating with Matrix A, it is sometimes necessary to normalize. With the degree Matrix D and Adjacency Matrix A, we have all information we need. Mostly, we want to normalize, such that our rows sum to 1.

$$A_{row-norm} = D^{-1}A \tag{5.3}$$

But we can achieve the same for columns, we just need to swap the two matrices:

$$A_{col-norm} = AD^{-1} \tag{5.4}$$

And a final, a probably the most useful normalization, is the symmetric normalization:

$$A_{sym} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \tag{5.5}$$

5.2.6   Graph Laplacian

The graph Laplacian is defined as follows:

$$L = D - A \tag{5.6}$$

5.2.6.1   Normalized Graph Laplacian

Symmetric normalized: $L_{sym} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ Random walk normalized: $L_{rw} = I - D^{-1}A$

5.2.6.2   Normalized Graph Laplacian eigen decomposition

$$\begin{aligned} L_{sym} &= U\Lambda U^T \\ U &= [u_0, \ldots, u_{N-1}] \in R^{N \times N} \\ \Lambda &= diag\left([\lambda_0, \ldots, \lambda_{N-1}]\right) \in R^{N \times N} \end{aligned} \tag{5.7}$$

In this scenario, eigenvectors are also known as *graph Fourier modes* and eigenvalues are known as the *spectral frequencies*.

Moreover, with the Graph Fourier Transform, we can calculate these values from a symmetric graph Laplacian.

5.2.7 Graph Construction
TODO: KNN

5.3 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[2].

5.3.1 Noise

A noisy observation is defined as: $y_n = y + \eta$

5.3.2 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.

5.3.3 Noisy Graph

For every noisy graph, there exists an original graph $G = \langle V, E \rangle$.

The noisy graph can be defined as follows:

$$
\begin{aligned}
G_{noisy} &= \langle V, E_{noisy} \rangle, \\
\text{with } E_{noisy} &= E \setminus E^- \cup E^+, \\
E^- &\subseteq E, \\
E^+ \cap E &= \emptyset
\end{aligned}
\tag{5.8}
$$

Basically, 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 then defined as follows:

$$
\bar{A}_{ij} = \begin{cases} 1 & \text{if } \langle i, j \rangle \in E_{noisy} \\ 0, & \text{otherwise} \end{cases}
\tag{5.9}
$$

The task of graph denoising, can therefore be written as:

$$
\bar{A} \xrightarrow[method]{Graph-denoising} \tilde{A} \approx A
\tag{5.10}
$$

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

5.3.3.1   Connection to link prediction

Link prediction is a task in Graph learning. The idea is to predict the 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:

$$M_p : E' \rightarrow [0, 1] \tag{5.11}$$

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, one could see Graph denoising as a link prediction problem.

The difference is, that in link prediction, we learn a model from a set of observed links $E_{observed} \subseteq E$ and in Graph denoising we learn the model from $E_{observed} \subseteq U$.

On could also say that link prediction problems are a subset of graph denoising problems.

5.4   Math Foundation

5.4.1   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 a approximation of the dominant eigenvector.

$$b_{k+1} = \frac{Ab_k}{||Ab_k||} \tag{5.12}$$

The algorithm not necessarily converge. 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.

5.4.2   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) [19] method, which iteratively solves the problem. During calculation, the eigenvalue spectrum will be folded around a reference value ϵ.

$$v^{t+1} = v^t - \alpha(H - \epsilon I)^2 v^t, \tag{5.13}$$

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

5.4.3 Embedding

An embedding $f : X \to Y$ 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.

5.5 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 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.

5.5.1 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 a dimensionality reduction as we can embed the data points in the low-dimensional manifold space.

There is a complete area of research devotes to this manifold assumption, and basically we then talk from Manifold Learning.

TODO: write more [3]

5.5.2 Line integral

Integral where function is evaluated along a curve

5.5.3 SO3

5.5.4 Lie Group

5.5.5 Fourrier Transform

5.5.6 Radon Transform

5.6 CT tomography and cryp-EM

5.7 Wasserstein Loss-function

6

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.

6.1 Reconstruct problem

Tomographic reconstruction is a popular inverse problem [4]. The aim is, to reconstruct an object, from its observed projections. More formally, the aim is to recover some density function f from overserved samples, taken from the line-integral $p(\cdot)$.

The problem can be defined as a two-dimensional (2-D) problem but also as a three-dimensional (3-D) problem. In 2-D, also called classical tomography reconstruct problem, the underlying density function is in two dimensions and the measurements lines lie on a plane. In 3-D, the density function deals within three dimensions and the measurements are lines with arbitrary orientation in space.

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. During some observation methods, (cryoEm), the underlying object

6.1.1 Notation

Let start with some notations. First of all, lets define the line integral p of our unknown density function f in the 2-D case:

$$
\begin{aligned}
p(\theta, s) &= Rf(\theta, s) \\
Rf(\theta, s) &= \int_{-\infty}^{\infty} f(x(z), y(z)) dz \\
&= \int_{-\infty}^{\infty} f((z \sin \theta + s \cos \theta), (-z \cos \theta + s \sin \theta)) dz
\end{aligned}
\tag{6.1}
$$

where p is the line integral of the density function f, θ the projection angle and s the distance from the origin.

In the 2-D case, the line integral corresponds to the Radon-Transform [17]. With the 2-D

Radon transform, we can map the density function f to the sinogram p. When data of the line integral is presented as a 2-D image, we are speaking from the *sinogram*. The concept of Radon transform was extended as well and can be applied to higher dimensions as well.

6.1.2   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 work for complete data and without noise.

6.1.3   Manifold assumption

In the reconstruct problem, we can apply the Manifold assumption from section 5.5.1. 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 3-D, where the underlying manifold corresponds to the Sphere. Therefore, we can derive, that our angles have the following property: In the 2-D case, $\theta \in SO(1)$ and the in 3-D 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.

6.2   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 2-D and 3-D scenario.

The main idea is, to exploit the fact that the underlying Manifold is known (Circle in 2-D and Sphere in 3-D).

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.

6.2.1   Overall notation

$$y_i = p(\theta, s) + noise$$
$$y_i = A(x, \theta) + noise \tag{6.2}$$
$$p(\theta, s) = A(x, \theta)$$

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

6.2.2 2D example

$a_i \in SO(1)$ $A := N \times N$ $B := N$

$f(x) = \int_{\theta=0}^{\pi} p(\theta, s)|_{s = x \cdot (-\sin\theta, cos\theta)} d\theta$

6.2.3 3D example

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

Simple Case in

7

Related Work

In the following section, related work will be introduced.

7.1 Graph Deep Learning

Graph deep learning is a fast evolving field in research. With Graph Neural Networks (GNN) [8] [18]

7.1.1 Graph feature extraction with GCN

7.1.2 Graph Convolutional Network

Graph Convolutional Networks (GCN) [9] 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:

$$
\begin{aligned}
H^{l+1} &= f(H^l, A), \\
\text{with } H^0 &= X, \\
H^L &= Z,
\end{aligned}
\tag{7.1}
$$

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:

$$
f(H^l, A) = \sigma(AH^lW^l)
\tag{7.2}
$$

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 [9] could show during experiments,

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

7.1.2.1 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[9]. 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 $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$.

And finally, we can put all things together, and replace A in the original equation:

$$f(H^l, A) = \sigma(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^l W^l) \tag{7.3}$$

7.1.2.2 Simple Graph Convolutional Network

Basically Power method with normalization

Simple Graph Convolutional Network (SGC) [21] 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.

$$\begin{aligned} Z &= S \ldots SXW^1 W^2 \ldots W^L \\ Z &= S^L XW^1 W^2 \ldots W^L \\ Z &= S^L XW \end{aligned} \tag{7.4}$$

where W is the matrix of all vector weights.

7.1.2.3 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.

$$g \star x = U((U^T g)(U^T x)) = U\hat{G}U^T x, \tag{7.5}$$

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:

$$\approx \sum_{i=0}^{k} \Delta^i x = U \left(\sum_{i=0}^{k} \Theta_i \Lambda^i \right) U^T x, \tag{7.6}$$

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 [9] paper, the approximation is done with $k = 1$

$$g \star x = \Theta(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x, \tag{7.7}$$

, where Kipf and Welling [9] 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.

[9] [21] [20]

7.2 Manifold Learning

[16] [12] [13]

7.3 Random Walk approaches

[5] [14] [7]

7.4 Denoising

7.4.1 Image Denoising

7.4.1.1 Non local means

Non local means is a state-of-the-art image denoising method [2]. 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:

$$NL[v](i) = \sum w(i,j)\,v(j) \tag{7.8}$$

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

- $0 \le w(i,j) \le 1$

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

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

7.4.2 Graph Denoising

[2] [10]

7.4.3   cryo-EM calcuation

7.5   Graph Laplacian Tomography From Unknown Random Projections

Coifman et al. [6] 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. [6] 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.

7.5.1   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.

$$
\begin{aligned}
P_\Theta(t) &= \int_L \rho(x, y) ds \\
&= \int_{-\infty}^{\infty} \rho(x, y) \, \delta(x \cos \Theta + y \sin \Theta - t) dx \, dy
\end{aligned}
\tag{7.9}
$$

An algorithm for estimating angles from given projections have been introduced by [1]. 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 [6], they introduced a new way of ordering the angles, using Graph Laplacian.

7.5.2   Laplace-Beltrami operator

[6] 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.

7.5.3   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 [6] 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:

$$W_{ij} = k\left(\frac{\|x_i - x_j\|^2}{2\epsilon}\right),$$

$$i, j = 1, \ldots, N$$

(7.10)

where $\|\cdot\|$ is the euclidean-norm, k a semi-positive kernel and $\epsilon > 0$ the bandwidth of the kernel. As mentioned in [6], 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 $\tilde{L}$.

$$\tilde{W} = D^{-1}WD^{-1}$$
$$\tilde{D} = \text{Degree matrix corresponding to } \tilde{W}$$
$$\tilde{L} = \tilde{D}^{-1}\tilde{W} - I$$
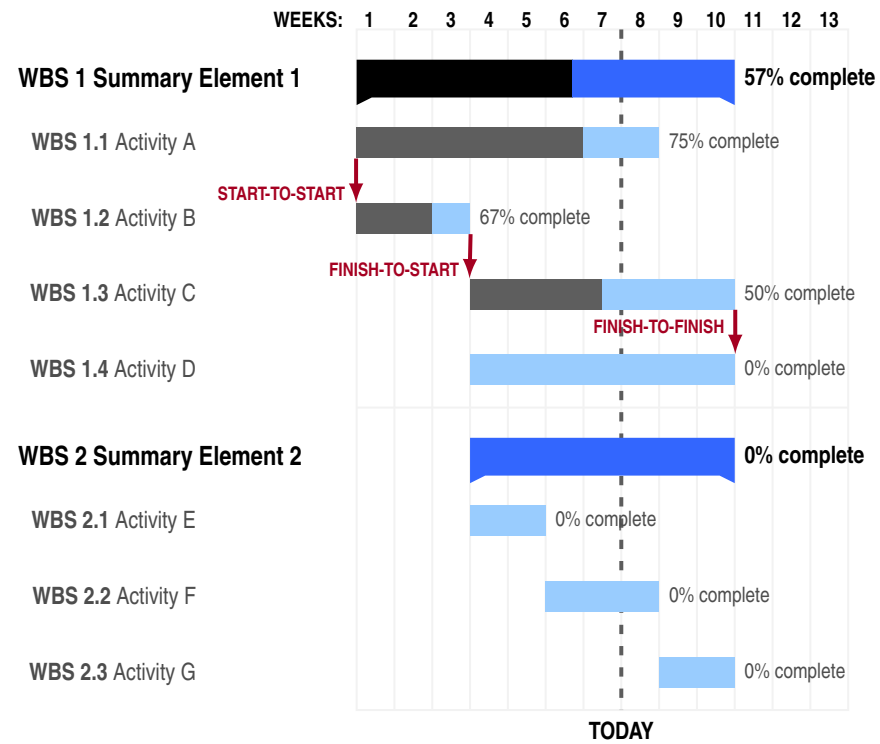
(7.11)

8

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.

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

[2] 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.

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

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

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

[6] 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.

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

[8] 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.

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

[10] 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.

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

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

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

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

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

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

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

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

[19] 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.

[20] 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.

[21] 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.