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 Foundation

General Questions

- Difference Graph Learning and Graph Representation
- Eigenvalues
- First-order Chebyshev
- Connection to CryoEm
- Benchmarking and Dataset

1.1 Introduction to Graph Learning

Graph Representational Learning [1]

1.1.1 Spectral graph theory

Spectral graph theory [6] deals with learning properties and characteristics of graphs, in regard to the graphs eigenvalues and eigenvectors.

1.1.2 Graph deep Learning

TODO: write more

1.2 Noise

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

[1] https://towardsdatascience.com/introduction-to-graph-representation-learning-a51c963d8d11

1.2.1 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.

1.2.1.1 Non local means

Non local means is a state-of-the-art image denoising method [1]. 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:

$$NL[v](i) = \sum w(i,j)v(j) \tag{1.1}$$

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

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

Without going into detail, the weight can be seen as a similarity measure of the two pixels. Moreover, are these similarities calculated over square neighbourhoods of the two pixels. 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.

1.3 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.

1.3.1 Adjacency Matrix

The adjacency Matrix of G is then defined as follows:

$$A_{ij} = \begin{cases} 1 & \text{if } \langle i,j \rangle \in E \\ 0, & \text{otherwise} \end{cases} \tag{1.2}$$

1.3.1.1 k-hop neighbourhood

1.3.2 Degree Matrix

The degree Matrix of G is defined as follows:

$$D_{ij} = \begin{cases} deg(v_i) & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \tag{1.3}$$

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

1.3.2.1 Adjacency normalization

We 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.

$$A_{rownorm} = D^{-1}A \tag{1.4}$$

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

$$A_{colnorm} = AD^{-1} \tag{1.5}$$

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

$$A_{sym} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \tag{1.6}$$

TODO: Add some nice example

1.3.3 Graph Laplacian

The graph Laplacian is defined as follows:

$$L = D - A \tag{1.7}$$

1.3.3.1 Normalized Graph Laplacian

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

1.3.3.2 Normalized Graph Laplacian eigendecomposition

$$\begin{aligned} L_{sym} &= U\Lambda U^T \\ U &= [u_0, \iddots, u_{N-1}] \in R^{NxN} \\ \Lambda &= diag([\lambda_0, \iddots, \lambda_{N-1}]) \in R^{NxN} \end{aligned} \tag{1.8}$$

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.

1.3.4 Graph Properties

1.3.4.1 Directed vs. undirected vs. weighted

1.3.4.2 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.

1.3.5 Node Properties

1.3.6 Edge Properties

1.3.7 Graph Construction

TODO: KNN

1.4 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 [1] and graphs are a well suited data structure for this task.

1.4.1 Noisy Graph

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

$$\begin{aligned} G_{noisy} &= \langle V, E_{noisy} \rangle \\ \text{where } E_{noisy} &= E \setminus E^- \cup E^+ \\ \text{and } E^- &\subseteq E, E^+ \cap E = \emptyset \end{aligned} \tag{1.9}$$

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:

$$\bar{A}_{ij} = \begin{cases} 1 & \text{if } \langle i, j \rangle \in E_{noisy} \\ 0, & \text{otherwise} \end{cases} \tag{1.10}$$

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

$$\bar{A} \xrightarrow[method]{Graphdenoising} \tilde{A} \approx A \tag{1.11}$$

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

1.4.2 Graph 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:

$$M_p : E' \rightarrow [0, 1] \tag{1.12}$$

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.
Questions:
Link prediction: learning model M from subset of E. Graph denoising: learning model M from subset of U.

1.5 Graph Laplacian

1.6 Deep Learning on Graphs

1.6.1 Graph Convolutional Network

Graph Convolutional Networks (GCN) [4] 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:

$$\begin{gathered} H^{l+1} = f(H^l, A), \\ \text{with } H^0 = X \text{ and } H^L = Z, \end{gathered} \tag{1.13}$$

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:

$$f(H^l, A) = \sigma(AH^lW^l) \tag{1.14}$$

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 [4] could show during experiments, this choice of $f(\cdot, \cdot)$ is already very powerful and leads to state-of-the-art results.

1.6.1.1 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 [4] 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:

$$f(H^l, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^lW^l) \tag{1.15}$$

Questions:

- During feature propagation, only node features are considered.
- Spectral Analysis Chapter

1.6.1.2 Simple Graph Convolutional Network

Simple Graph Convolutional Network (SGC) [9] 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.

$$\begin{aligned} Z = S \cdot\cdot\, SXW^1W^2 \cdot\cdot\, W^L \\ Z = S^LXW^1W^2 \cdot\cdot\, W^L \\ Z = S^LXW \end{aligned} \tag{1.16}$$

where W is the matrix of all vector weights.

1.6.1.3 Link go Graph Laplacian:

[2]

not read currently: [3]

[2] https://towardsdatascience.com/spectral-graph-convolution-explained-and-implemented-step-by-step-2e495b57f801
[3] https://towardsdatascience.com/tutorial-on-graph-neural-networks-for-computer-vision-and-beyond-part-2-be6d71d70f49

1.7 Maths Foundation

1.7.1 Hilbert Space

1.7.2 SO(3), S,

1.7.3 LIE Group?

1.7.4 Principle component analysis - PCA

1.7.5 Local PCA

1.7.6 K-means

1.7.7 K-nearest neighboors Graph construction

1.7.8 Fourier domain

1.7.9 Graph Fourier transform

1.8 Graph Foundation

1.8.1 Curvature

1.8.2 Graphlets

1.8.3 Geodesic distance

1.8.4 Manifold Assumption

1.8.5 Point Cloud

1.8.6 Laplace

1.8.7 Walk Pooling

1.8.8 Link prediction

1.8.9 Katz Index

1.8.10 Eigenvector centrality

1.8.11 NN forward passing

$$H^{i+1} = \sigma(W^i H^i + b^i) \tag{1.17}$$

[4]

[4] https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-a2bfdb7aba7b

1.8.12 NN activation functions

1.8.13 Drops after layer

1.8.14 Hyper Graphs

1.8.15 Power Iterations

1.8.16 Manifold Learning

1.8.17 Signal Processing

1.8.18 Nonlinear dimensionality reduction

1.9 Diffusion Maps:

Coifman and Lafon [2] [2]
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 alignement
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 modeled as a graph, can be represented (embedded) as a cloud of points in a Euclidean space. two major advantages over classical dimensionality redution (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, chian 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

1.9.1 Vector Diffusion Maps (VDM)

[5] VDMis a mathematical and algorithmic generalization of diffusion maps and other nonlinear 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 tangets are mapped to the manifold. A way to globally connect Local PCAs.
SNR: signal-to-noise-ratio
LLE: ISOMAP: Laplacian eigenmaps:

1.9.2 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 $d << p$.

1.9.3 Multi-Frequency Vector Diffusion Maps (MFVDM)

[3] **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}.

1.10 Graph Laplacian Tomography From Unknown Random Projections

A reference that I already mentioned in the first mail: standard approach that we need to compare with. Maybe their setting (2D tomography with unknown angle) is a good setting to start with.

1.11 denoising

Recover original image from noisy observation. Is Achieved by averaging.
- classical local smoothing filters: - gaussian filters - anisotropic filters - Total Variation minimization - neighborhood filters
- neighborhood filters (review of image denoising) - non local means - functions adapted kernels (nonlinear independent component analysis)

1.12 Non-local means

Image denoising accurately done.
Better performance, when algorithms tries to correct noise rather than separate noise from original image.
Compares similar pixel neighborhoods and assign large weighted for similar pixels.

1.13 Learning to Drop

Graph denoising

1.14 WalkPooling

Image denoising

1.15 Point Clouds

1.15.1 Dynamic graph Cnn for learning on point clouds

One of the few reference related to graph neural network and learning of graph structure.

1.15.2 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.

2

Introduction

This is the introduction to the thesis template. The goal is to give students a starting point on how to format and style their Bachelor or Master thesis[5].
Please make sure to always use the most current version of this template, by downloading it always from the original git repository:

http://www.github.com/ivangiangreco/unibas-latex

We will use throughout this tutorial some references to Turing's imitation game [8] and the Turing machine [7]. You may be interested in reading these papers.

The package comes with an option regarding the bibliography style. You can include the package with

```
\usepackage[citeauthor]{basilea}
```

to be able to cite authors directly with

```
\citet{turing:1950}
```

If the option is enabled, then the following reference should print Turing [2]: Turing [8]

[5] This document also shows how to use the template.

3

Body of the Thesis

This is the body of the thesis.

3.1 Structure

3.1.1 Sub-Section

3.1.1.1 Sub-Sub-Section

Paragraph

Even Sub-Paragraph This is the body text. Make sure that when you reference anything you use labels and references. When you refer to anything, you normally capitalise the type of object you reference to, e.g. Section 3.1 instead of section 3.1. You may also just use the `cref` command and it will generate the label, e.g., for Section 3.1, we did not specify the word "Section".
Hint: Try to structure your labels as it is done with `sec:my-label` and `fig:machine`, etc.

3.2 Equations

A Turing Machine is a 7-Tuple:

$$M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle \tag{3.1}$$

A Turing Machine is a 7-Tuple even if defined in the text, as in $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$.

3.3 Tables

Some tables can also be used as shown in Table 3.1[6]. Remember that tables might be positioned elsewhere in the document. You can force positioning by putting a `ht!` in the definition.

[6] Table captions are normally above the table.

Table 3.1: Frequency of Paper Citations. By the way: Make sure to put the label always after the caption, otherwise LaTeX might reference wrongly!

Title	f	Comments
The chemical basis of morphogenesis	7327	
On computable numbers, with an application to the ...	6347	Turing Machine
Computing machinery and intelligence	6130	

3.4 Figures

Figures are nice to show concepts visually. For organising well your thesis, put all figures in the Figures folder. Figure 3.1 shows how to insert an image into your document. Figure 3.2 references a figure with multiple sub-figures, whereas the sub-figures are referenced by Fig. 3.2(a), etc.

Missing: Description figure.

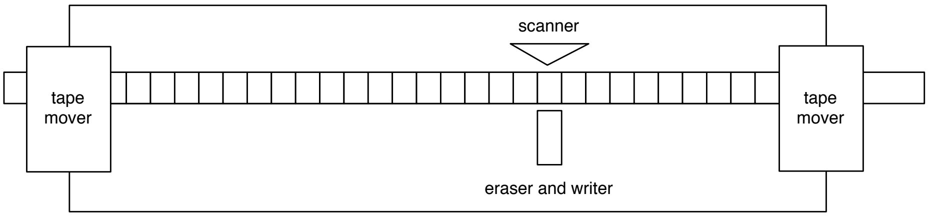
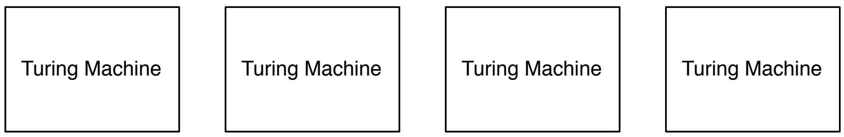


Figure 3.1: A Turing machine.



(a) Turing Machine 1 (b) Turing Machine 2 (c) Turing Machine 3 (d) Turing Machine 4

Figure 3.2: Plots of four Turing machines

3.5 Packages

These packages might be helpful for writing your thesis:

`caption` to adjust the look of your captions

`glossaries` for creating glossaries (also list of symbols)

`makeidx` for indexes and the back of your document

`algorithm`, `algorithmicx`, `algpseudocode` for adding algorithms to your document

4 Conclusion

This is a short conclusion on the thesis template documentation. If you have any comments or suggestions for improving the template, if you find any bugs or problems, please contact me.

Good luck with your thesis!

Bibliography

[1] 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.

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

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

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

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

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

[7] Alan M Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society*, 42(2):230–265, 1936.

[8] Alan M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[9] 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.