2 Machine Learning on Graphs

Note takers: Bardou, Brandao, Even, Gonon, Mitarchuk, Natura, Le Q-T, Pic, Shilov, Weber

Instructor: Pierre Vandergheynst (EPFL)

2.1 Hour 1 (Monday 10-11 am)

2.1.1 Introduction

Graphs are natural structures to represent data in many applications, such as biology (e.g. proteins) or networks (e.g. transportation, social, energy). On such a structure, building, learning and processing meaningful features can be done at different scales:

- at node scale: to classify a node, to predict an edge,
- at a (sub)graph scale: to detect and classify structures (e.g. communities)

More generally, we want to be able to process a (sample of) a graph signal, to be able to classify information over a fixed network. These types of tasks require adapted tools and methods. The following sections describe some of them.

2.1.2 Some elements of Spectral Graph Theory

First of all, let us formally define what a graph is and some basic related notions.

Definition 1. *A graph $G = (V, E)$ is a pair, composed of $V = \{v_1, \cdots, v_N\}$ a set of vertices and $E = \{e_1, \cdots, e_M\}$ a set of edges.*

A graph has some basic descriptors, gathering information about its structure. The degree of a vertex v, denoted $d(v)$ or d_v is the number of nodes connected to it.

$$d(v) = |\{u \in V \text{ s.t. } (u, v) \in E \textbf{ or } (v, u) \in E\}|$$

The degrees of each node are gathered in the degree matrix $\mathbf{D}(G) = \mathrm{D}iag(d_1, \cdots, d_N)$.

Definition 2. *(Diameter, volume of a graph) The diameter $d(G)$ of a graph G is the longest shortest path between a pair of vertices. The volume $vol(G)$ of a graph G is the sum of the degree of its vertices, that is $vol(G) = \sum_{v \in V} d_v = tr(\mathbf{D})$.*

The incidence matrix $\mathbf{S}$ is a $N \times M$ matrix defined as:

$$\mathrm{S}_{i,j} = \begin{cases} +1 & \text{if } e_j = (v_i, v_k) \text{ for some } k \\ -1 & \text{if } e_j = (v_k, v_i) \text{ for some } k \\ 0 & \text{otherwise} \end{cases}$$

We can represent G with the adjacency matrix $\mathbf{A}$, a $N \times N$ matrix:

$$\mathbf{A}_{i,j} = \begin{cases} +1 & \text{if } (v_i, v_j) \in E \textbf{ or } (v_j, v_i) \in E \\ 0 & \text{otherwise} \end{cases}$$

We can extend the definition of the degree of a node to weighted graphs, for which the adjacency matrix becomes a weight matrix $\mathbf{W}$, with $\mathbf{W}_{i,j} \geq 0$: $d(v_i) = \sum_{j=1}^{N} \mathbf{W}_{i,j}$.

With these definitions, we can eventually define a slightly more complex matrix:

$$\mathrm{S}\mathrm{S}^T = \mathbf{D} - \mathbf{A}$$

Definition 3. *The (unnormalized) Laplacian of G, denoted $\mathbf{L}$ is defined as:*

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

$\mathbf{L}$ does not depend on the orientation of the edges, therefore it can be defined for undirected graph too. For weighted graphs, we have the extension $\mathbf{L} = \mathbf{D} - \mathbf{W}$. By definition, $\mathbf{L}$ is symmetric, but also positive semi-definite.

Proof. For any weight matrix $\mathbf{W}$, we can show through simple calculations that $\forall x \in \mathbb{R}^N$:

$$x^T \mathbf{L} x = \frac{1}{2} \sum_{j \sim i} \mathbf{W}_{i,j} (x[i] - x[j])^2 \geq 0$$

□

Considering a graph signal $f \in \mathbb{R}^N$ (node attribute), we can observe the differential nature of the incidence matrix $\mathbf{S}$ by noting that

$$(\mathbf{S}^T f)[j] = f[i] - f[k]$$

which can be considered as the derivative of f along edge j. Here, edge j connects vertex v_i to vertex v_k. Extending this observation to the whole signal f, we can think of $F = \mathbf{S}^T f \in \mathbb{R}^M$ as the gradient of f. Note that F is an *edge-based* signal. Similarly, for an edge-based signal G, $g = \mathbf{S}G \in \mathbb{R}^N$ is the divergence of G. Note that g is a *vertex-based* signal.

This insight can help us understand the nature of the Laplacian $\mathbf{L} = \mathbf{S}\mathbf{S}^T$. Considering a graph (vertex-based) signal, we have:

$$\begin{aligned} f^T \mathbf{L} f &= f^T \mathbf{S}\mathbf{S}^T f \\ &= ||\mathbf{S}^T f||_2^2 \\ &= \sum_{i \sim k} (f[i] - f[k])^2 \end{aligned}$$

Therefore, for a weighted graph, we can consider the quadratic form of a signal involving the Laplacian $\mathbf{L}$:

$$f^T \mathbf{L} f = \sum_{i \sim k} \mathbf{W}_{i,k} (f[i] - f[k])^2$$

as a measure of how *smooth* the signal is.

Since $\mathbf{L}$ is a real, symmetric and positive semi-definite matrix, it has an eigendecomposition into real eigenvalues and eigenvectors λ_i, u_i, with non-negative eigenvalues ($0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$). Given the properties of the Laplacian $\mathbf{L}$, its eigendecomposition may give us information about the structure of the graph.

Proposition. The number of connected components c of G is the dimension of the nullspace of $\mathbf{L}$. Furthermore, the null space of $\mathbf{L}$ has a basis of indicator vectors of the connected components of G. An indicator of a subset H of V is

$$x \in \mathbb{R}^N \text{ s.t. } \begin{cases} x[i] = 1 & \text{if } \in H \\ x[i] = 0 & \text{otherwise} \end{cases}$$

Definition 4. *(Algebraic connectivity, Fielder Vector) The algebraic connectivity of a graph is the second smallest eigenvalue of* $\mathbf{L}$*,* λ_2*, which is positive if and only if the graph is connected. Its associated eigenvector,* u_2 *is called the Fielder vector.*

The value of λ_2 gradually increases with the connectedness of the graph. In particular, we can show that $\lambda_2 \geq \frac{1}{vol(G)\mathrm{d}(G)}$.

Definition 5. *The Cheeger Constant of a graph is denoted* $h(G)$ *and is defined as:*

$$h(G) = \min_{A \subset V} \left\{ \frac{|\partial A|}{\min\left(vol(A), vol(\bar{A})\right)} \; s.t. \; 0 < |A| < \frac{1}{2}|V| \right\}$$

with $\bar{A}$ *the complementary of* A*,* $vol(A) = \sum_{v \in A} d(v)$ *and* $\partial A = \{(u, v) \in E \; s.t. \; u \in A, v \in \bar{A}\}$ *gathering the edges between* A *and* $\bar{A}$*.*

The Cheeger Constant is able to measure the presence of "bottlenecks" which are formed by strongly connected components loosely connected with each others.

We can relate the Cheeger Constant with algebraic connectivity by the Cheeger inequalities. A simple example is given below:

Theorem 2.1. *For a general graph* G*,*

$$2h(G) \geq \lambda_2 \geq \frac{h^2(G)}{2}$$

2.2 Hour 2 (Monday 11 am -12 pm)

2.2.1 Cut and Cluster

WIP

We can associate a cost to cutting a graph in subsets $A_1, \ldots, A_n$ inspired by the Cheeger constant, which is the minimal surface to volume ratio among graph cuts. To do so, begin by defining the total weight of the edges between $A, B \subset V$:

$$C(A, B) \coloneqq \sum_{i \in A, j \in B} \mathbf{W}\left[i, j\right], \tag{2.1}$$

which can be regarded as a proxy for surface between A and B and

$$\mathrm{cut}(A_1, \ldots, A_k) \coloneqq \frac{1}{2} \sum_{i=1}^{k} C(A_i, \bar{A}_i) \tag{2.2}$$

For two different definitions of subset volume we obtain

$$\mathrm{RatioCut}(A, \bar{A}) = \frac{\mathrm{cut}(A, \bar{A})}{|A|} + \frac{\mathrm{cut}(\bar{A}, A)}{|\bar{A}|} \tag{2.3}$$

$$\text{NormalizedCut}(A, \bar{A}) = \frac{\text{cut}(A, \bar{A})}{\text{vol}(A)} + \frac{\text{cut}(\bar{A}, A)}{\text{vol}(\bar{A})} \tag{2.4}$$

We shall see that there is a relationship between

$$\min_{A \subset V} \text{RatioCut}(A, \bar{A}) \tag{2.5}$$

and the second eigenvalue of the Laplacian, which will allow us to approximate RatioCut. To do so, first define the indicator function

$$f[i] = \begin{cases} \sqrt{|\bar{A}|/|A|} & \text{if} i \in A \\ -\sqrt{|A|/|\bar{A}|} & \text{if} i \in \bar{A} \end{cases} \tag{2.6}$$

Since we have

$$\begin{aligned} f^\top \mathbf{L} f &= f^\top (D) f - f^\top W f && (2.7)\\ &= \sum_i f_i \sum_j w_{ji} f_i - \sum_i f_i \sum_j w_{ij} f_j && (2.8)\\ &= \sum_{i,j} w_{ij} (f_i^2 - f_i f_j) && (2.9)\\ &= \frac{1}{2} \sum_{i,j} w_{ij} (f_i^2 - 2 f_i f_j + f_j^2) && (2.10)\\ &= \frac{1}{2} \sum_{i,j} w_{ij} (f_i - f_j)^2 && (2.11)\\ &= \sum_{i \sim j} \mathbf{W}(i,j) (f[i] - f[j])^2 && (2.12) \end{aligned}$$

replacing this above we can show that (assuming $A, \bar{A} \neq \emptyset$)

$$f^\top \mathbf{L} f = \frac{1}{2} \sum_{i,j} w_{ij}(f_i - f_j)^2 \tag{2.13}$$

$$= \frac{1}{2}\left(\sum_{i,j \in A} w_{ij}(f_i - f_j)^2 + \sum_{i,j \in \bar{A}} w_{ij}(f_i - f_j)^2 + \sum_{i \in A, j \in \bar{A}} w_{ij}(f_i - f_j)^2 + \sum_{i \in \bar{A}, j \in A} w_{ij}(f_i - f_j)^2 \right) \tag{2.14}$$

$$= \frac{1}{2}\left(\sum_{i \in A, j \in \bar{A}} w_{ij}(f_i - f_j)^2 + \sum_{i \in \bar{A}, j \in A} w_{ij}(f_i - f_j)^2 \right) \tag{2.15}$$

$$= \frac{1}{2}\left(\sum_{i \in A, j \in \bar{A}} w_{ij}\left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 + \sum_{i \in \bar{A}, j \in A} w_{ij}\left(-\sqrt{\frac{|A|}{|\bar{A}|}} - \sqrt{\frac{|\bar{A}|}{|A|}} \right)^2 \right) \tag{2.16}$$

$$= \frac{1}{2}\left(C(A, \bar{A})\left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 + C(\bar{A}, A)\left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 \right) \tag{2.17}$$

$$= C(A, \bar{A})\left(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2 \right) \tag{2.18}$$

$$= C(A, \bar{A})\left(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + \frac{|A|}{|\bar{A}|} + \frac{|\bar{A}|}{|A|} \right) \tag{2.19}$$

$$= C(A, \bar{A})\left(\frac{|V|}{|A|} + \frac{|V|}{|\bar{A}|} \right) \tag{2.20}$$

$$= |V|\left(\frac{\text{cut}(\bar{A}, A)}{|A|} + \frac{\text{cut}(A, \bar{A})}{|\bar{A}|} \right) \tag{2.21}$$

$$= |V| \,\text{RatioCut}(A, \bar{A}) \tag{2.22}$$

which implies

$$\frac{f^\top \mathbf{L} f}{|V|} = \text{RatioCut}(A, \bar{A})$$

Note that

$$\begin{aligned} f^\top f &= \sum_i f_i^2 \\ &= \sum_{i \in A} f_i^2 + \sum_{i \in \bar{A}} f_i^2 \\ &= \sum_{i \in A} \frac{|\bar{A}|}{|A|} + \sum_{i \in \bar{A}} \frac{|A|}{|\bar{A}|} \\ &= |\bar{A}| + |A| = |V| \end{aligned}$$

so we can replace the left-hand side of the expression above by the Rayleigh quotient of f and $\mathbf{L}$

$$\frac{f^\top \mathbf{L} f}{f^\top f} = \text{RatioCut}(A, \bar{A})$$

Solving

$$\min_{A \neq \emptyset \subset V, f \text{ indicator of } A} \frac{f^\top \mathbf{L} f}{f^\top f}$$

is thus equivalent to solving Problem (2.5). Note that the indicator function is a constant iff $A, \bar{A} \neq \emptyset$ and so

$$\min_{A \subset V, f \text{ indicator of } A, f^\top \mathbf{1} = 0} \frac{f^\top \mathbf{L} f}{f^\top f}$$

If we relax the requirement that f is the indicator above, by the Courant–Fischer–Weyl min-max principle, minimizing the Rayleigh quotient over the space orthogonal to the first eigenvalue yields the second eigenvalue. The f for which this happens is called the *Fiedler vector* u_f. From (2.1), we see that minimizing this quotient with these constraints corresponds to finding the *smoothest* vector (in the sense that the differences over adjacent nodes are small — and more so for strongly connected nodes), that is not constant. To recover the partition we choose clusters according to the sign of the eigenvector components, nodes i, j being in the same cluster if $\text{sign}(u_f[i]) = \text{sign}(u_f[j])$. Components with different signs are guaranteed to exist because the second eigenvalue is orthogonal to constants.

2.2.2 Generalizing to $k > 2$

We begin by generalizing the definition of RatioCut given above:

$$\text{RatioCut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_k, \bar{A}_k)}{|A_k|} \tag{2.23}$$

We define $F \in \mathbb{R}^{N \times k}$ as

$$F[i, j] = f_{ji} = \begin{cases} 1/\sqrt{|A_j|} \text{ if } v_i \in A_j \\ 0 \text{ otherwise} \end{cases} \tag{2.24}$$

We observe that

$$\begin{aligned}
\left(F^\top F\right)_{ij} &= \sum_k f_{ik}^\top f_{kj} \\
&= \sum_k f_{ki} f_{kj} \\
&= \sum_{k \in A_i \wedge k \in A_j} f_{ki} f_{kj} + \sum_{k \notin A_i \vee k \notin A_j} f_{ki} f_{kj} \\
&= \sum_{k \in A_i \wedge k \in A_j} \frac{1}{|A_j|} \\
&= \delta_{ij}
\end{aligned}$$

where the last step follows from noting that the same vertex cannot be in more than one element of the partition and that we are summing over all elements of that partition.

Observe that using Eq. (2.6) and (2.7)

$$\begin{aligned}
f_k^T \mathbf{L} f_k &= \frac{1}{2} \sum_{i,j} w_{ij} (f_{ki} - f_{kj})^2 \\
&= \frac{1}{2} \left(\sum_{i,j \in A_k} w_{ij} (f_{ki} - f_{kj})^2 + \sum_{i \notin A_k \vee j \notin A_k} w_{ij} (f_{ki} - f_{kj})^2 \right) \\
&= \frac{1}{2} \left(\sum_{i,j \in A_k} w_{ij} (f_{ki} - f_{kj})^2 + \sum_{j \in A_k, i \notin A_k} w_{ij} (f_{ki} - f_{kj})^2 + \sum_{i \in A_k, j \notin A_k} w_{ij} (f_{ki} - f_{kj})^2 + 0 \right) \\
&= \frac{1}{2} \left(\sum_{i \notin A_k, j \in A_k} w_{ij} \frac{1}{|A_k|} + \sum_{i \in A_k, j \notin A_k} w_{ij} \frac{1}{|A_k|} \right) \\
&= \frac{1}{2} \left(\frac{C(\bar{A_k}, A_k)}{|A_k|} + \frac{C(A_k, \bar{A_k})}{|A_k|} \right) \\
&= \frac{\mathrm{cut}(A_k, \bar{A_k})}{|A_k|}
\end{aligned}$$

Which implies that

$$\begin{aligned}
\mathrm{Tr}(F^\top \mathbf{L} F) &= \sum_k f_k^T \mathbf{L} f_k \\
&= \sum_k \frac{\mathrm{cut}(A_k, \bar{A_k})}{|A_k|}
\end{aligned}$$

which, as in the case $k = 2$ suggests solving a relaxed problem.

2.3 Hour 3 (Wednesday 11-12 am)

2.3.1 Signal Processing on Graphs

Goal 1: create models of smooth graph signals and operators to manipulate them (GSP)

For L a symmetric matrix ($L = u^\top \mathrm{D}iag(\lambda_1, \dots, \lambda_N)u$ for $u = (u_1, \dots, u_N)$ an orthonormal matrix), a Laplacian matrix in the sequel, and f a real-valued function, let $f(L)$ be defined as $L = u^\top \mathrm{D}iag(f(\lambda_1), \dots, f(\lambda_N))u$: f is applied on the eigenvalues. Typical example: diffusion on graphs, defined as

$$\partial_t f = -Lf\,,$$

where L is a graph Laplacian, yielding $f_t = e^{-tL} f_0$ and making appear the graph Fourier, defined as $\hat{f}(\ell) = \sum_j f(j) u_\ell(j)$, and the graph Fourier inverse.

Graph Fourier transform of f a signal: $\hat{f}(\ell) = \sum_j f(j) u_\ell(j)$, and its Fourier inverse $f(i) = \sum_k \hat{f}(k) u_k(i)$.

f is a M-smooth signal on a graph if ($\|S^\top M\|_2^2 \leq M$), or equivalently if $|\hat{f}(\ell)| \leq M\lambda_\ell$.

Definition 6. *A graph filter is an operator acting on graph signals f_{in}, represented as a function of the Laplacian: $g(L)$, leading to $f_{out} = g(L) f_{in}$.*

We have $\widehat{g(L)f}(\ell) = g(\lambda_\ell)\hat{f}(\ell)$.

Spectral kernels:

Localization: define $(T_i g)(n) = g(L)_{i,n}$ (is that it?)

Polynomial localization: g polynomial of degree less than $K \implies$ if the distance between i and n in the graph is more than K, then $(T_i g)(n) = 0$.

Efficient implementation of a graph filter Hammond et al. [2011] The image $g(L)f$ of a graph signal $f \in \mathbb{R}^N$ by a graph filter $g(L)$ may be approximately computed without having to compute spectral decomposition of the Laplacian matrix ($O(N^3)$ operations with SVD). Indeed, if g has some kind of smoothness, it can be approximated by a polynomial $g(L) \simeq \sum_{k=1}^{K-1} a_k L^k$. Computing Lf, then $L^2 f = L(Lf)$ and so on until $L^{K-1} f$ can be done in $O(KN^2)$ operations. If L is sparse, using a sparse matrix representation allows to actually use only $O(KM)$ operations (recall that M is the number of edges while N is the number of vertices)?

Summary of this section. We have a controlled way to define a graph filter g as localized as we want: the smoother it is, the better it can be approximated with polynomials and hence well localized. Moreover, we have an efficient way to evaluate the filter on a signal.

2.3.2 Designing and Processing Graph Features with Graph Signal Processing

Graph wavelets. Let us now first show how to design an analog of wavelets on graphs Hammond et al. [2011], wavelets being a classical powerful signal processing tool on Euclidean

domains. The main motivation for wavelets is that it allows to define a filter localized both in time/space and frequency. Recall that the Fourier transform of a graph signal $f \in \mathbb{R}^N$ is defined by $\hat{f}(\ell) = \langle u_\ell, f \rangle$ and a graph filter $g(L)$ operates as $g(L)f(n) := \sum_{\ell=1}^{N} g(\lambda_\ell)\hat{f}(\ell)u_\ell(n)$. As for classical wavelets, one can translate and dilate a "mother wavelet" g in order to define wavelets with different localizations in time/spatial and frequency domains. In order to control the localization in the frequency domain, we scale g with a scale parameter $s \in \mathbb{R}$:

$$D_s g := g(sL).$$

For instance, when $s < 1$, it dilates g in the frequency domain ($\lambda \mapsto g(\lambda)$ is replaced by $\lambda \mapsto g(s\lambda)$) likely resulting in a smoother function than g in the sense that it has smaller variations than g within "communities" of the graphs hence it is less spread on the nodes of the graph. Decreasing the scale reduces the spread of the wavelet, see Figure 2.1.

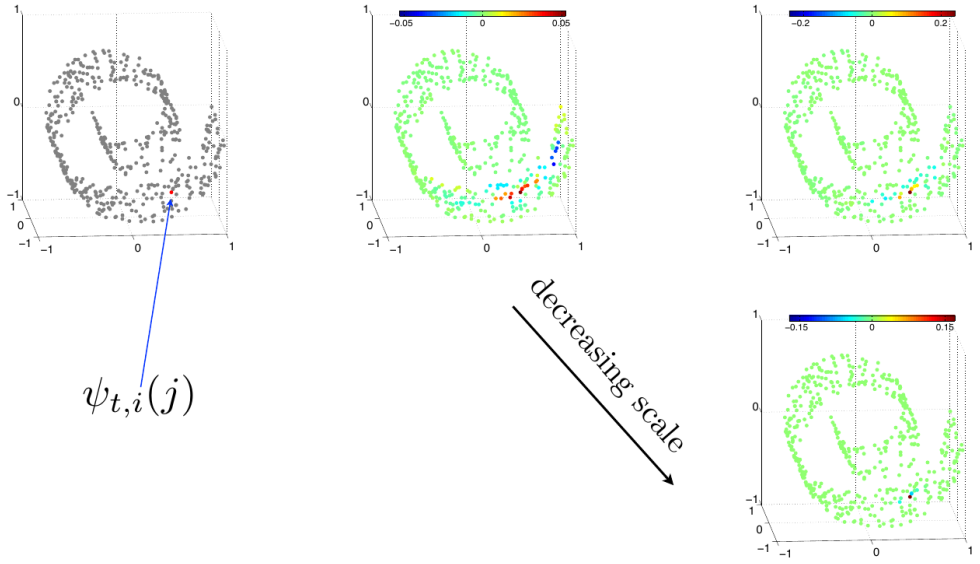

Figure 2.1: Illustration of the phenomenon of decreasing scale on a toy spiral dataset.

In order to control the localization in the time/spatial domain, we apply $g(L)$ on the impulse δ_n localized on a single vertex $n \in V$. This defines an analog of the translation operator but for graph:

$$T_n g(i) := \sum_{\ell=1}^{N} \hat{g}(\lambda_\ell)u_\ell(n)u_\ell(i).$$

Definition 7. *(Spectral Graph Wavelets) Let $g(L)$ be a graph filter. The associated wavelet operator scaled by s and centered at vertex n is defined as:*

$$\psi_{s,n}(i) := (T_n D_s g)(i) = \sum_{\ell=1}^{N} \hat{g}(s\lambda_\ell)u_\ell^*(n)u_\ell(i).$$

Neighbourhoods-dependent features based on wavelets. Let $g(L)$ be a graph filter. For every vertex n, the feature $\|\psi_{1,n}\|_2^2 = \|T_n g\|_2^2 = \sum_{\ell=1}^N (g(\lambda_\ell))^2 |u_\ell(n)|^2$ depends on the structure of the neighbourhood. For instance, if g is K-localized, then by definition $T_n g$ only depends on the k-nearest neighbours of vertex n. In order to change the spread of the filter g, one can more generally consider a scale $s \in \mathbb{R}$ and associate to every vertex n the feature $\|\psi_{s,n}\|_2^2$. Note that if the scale is too small, these features will not discriminate the vertices since $\psi_{s,n}$ is then localized at the vertex n for every n. Similarly, when s is too large, $\psi_{s,n}$ is localized on the whole graph and the features are non-discriminant.

A linear interpolation of features Shuman et al. [2016]. Let us now describe a simple interpolation example where we recover missing features corresponding to vertices in $V_1^c := V \setminus V_1$ based on the known features f_{V_1} associated with the vertices in V_1. What follows is a simplified version of Pesenson's variational spline interpolation. We consider $\varepsilon > 0$, a regularized version of the Laplacian $\bar{L} := L + \varepsilon I$ and we define $\varphi_j := T_j g$ with g that acts on the spectrum of $\bar{L}$ as $g(\lambda) = \frac{1}{\lambda}$. Note that the functions φ_j are regularized Green's functions since they satisfy $\bar{L}\varphi_j = \delta_j$. Define α_* as:

$$\alpha_* := [\bar{L}_{V_1,V_1} - \bar{L}_{V_1,V_1^c}(\bar{L}_{V_1^c,V_1^c})^{-1}\bar{L}_{V_1^c,V_1}]f_{V_1}.$$

Upsample α_* to define a signal on the whole graph:

$$f_{\text{upsample}} = \sum_{n \in V_1} \alpha_*(n)\delta_n.$$

We now interpolate the original signal f as follows:

$$f_{\text{interp}} := g(\bar{L}) f_{\text{upsample}}.$$

One can check that this coincides with f on V_1 where the signal was known. Moreover, this choice of interpolation minimizes, over all $\hat{f}$ that coincides with f on V_1, some kind of energy (typically $\hat{f}^T \bar{L} \hat{f}$).

2.4 Hour 4 (Thursday 2-3 pm)

In the previous lectures on Graph Representation Learning we:

- learned that the eigendecomposition of the Laplacian of graph reveals a lot about its structure
- leveraged the smoothness of Eigenvectors for partition vectors / signals
- did dimensionality reduction unsupervised
- came up with the idea of operators that can be applied to graph filters
- set up an algorithm that allowed us to apply these operators in a computational efficient way to signals, which can be used to craft specific features for specific applications.

This can be summarized as **LOVE** - "**L**aplacian **O**rthogonal eigen**VE**ctors" (see Figure 2.2).

Another topic we discussed and that we will now explore further is Spectral Clustering.

Figure 2.2: Beatles (1967) and Vandergheynst (2022) - "All You Need Is LOVE"

2.4.1 More Spectral Clustering

Let G be a large graph, which is guaranteed to have many communities. The purpose of this section is to show tools that allow to find the communities of G with a low computational budget (in this regime even performing spectral clustering would be too expensive). The idea is to sample (in a good way) m nodes on the graph with the hope that this sub-graph will be a good representation of G, in the sense that it reveals the k clusters of the graph.

The immediate challenge is to estimate k. This quantity is usually not given as input, in particular in unsupervised learning.

To address this problem, the main idea is to heuristically perceive a gap in the eigenvalues of the Laplacian L of G. The integer k is the number of the greatest eigenvalues of L before the gap. These k will help us to determine m. The search for k can be done in a stochastic approach. More details on these techniques can be found in Napoli et al. [2016].

From now on assume that a reasonable estimate of k has been computed and let us get back to the sampling of m vertices. A trivial lower bound on m to recover all k clusters of G would be $m \geq k$. We will not be able to get away with only $m = O(k)$ samples. But we will show that up to some parameters, with high probability $m = O(k \log(k))$ many samples suffice.

Let $L = U\Lambda U^\top$ be the eigendecomposition of L such that diagonal entries of Λ are ordered in increasing order: $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

Then, we sample m nodes independently, where node i is sampled with probability

$$p_i^* := \frac{\|U_k^\top \delta_i\|_2^2}{k}, \tag{2.25}$$

where U_k is the submatrix consisting of the first k columns of U and δ_i is the i-th vector of the standard basis in $\mathbb{R}^n$.

If we represent the action of sampling $m \leq n$ nodes by a matrix $M \in \{0,1\}^{m \times n}$, and grouping the the probability of a node to be selected in a diagonal matrix P we can derive guaranties on the compression-decompression of the graph if M is chosen wisely. More formally:

Lemma 2.2. *There exists a distribution p^* over the nodes of G, such that if $M \sim p^*$ then for all $0 < \varepsilon, \delta < 1, \forall x \in \text{span}(U_k)$*

$$\mathbb{P}\left[(1-\delta)\|x\|_2^2 \leq \frac{1}{m}\|MP^{-\frac{1}{2}}x\|_2^2 \leq (1+\delta)\|x\|_2^2\right] \geq 1-\varepsilon$$

for $m = \Theta\left(\frac{1}{\delta^2}k\log\left(\frac{k}{\varepsilon}\right)\right)$.

However, recall that we mentioned at the beginning of this section that computing such a decomposition would computationally be too expensive. Fortunately, approximating the p_i is sufficient for our purposes and such an approximation can be obtained efficiently with high probability:

What is interesting is that we can estimate the optimal distribution of the above assertion by applying random filters and taking the mean.

The recovering procedure of the graph is given by:

$$\min_{z \in \mathbb{R}^n} \|P_\omega^{-\frac{1}{2}}(Mz - y)\|_2^2 + \gamma \underbrace{z^T g(L) z}_{A}$$

Where $y = Mx$ and A is soft constraint of frequencies.

The experiments conducted show the difference in reconstruction performance depending on the distribution chosen on the graph. In particular, the uniform distribution is generally a bad idea this task. Another experiment carried out on the graph of proximity of pixels of an image (for a given image we associate a graph weighted by the proximity between two pixels) shows that we can reconstruct the image quite faithfully by sampling 7% of the nodes.

2.5 Hour 5 (Thursday 3-4 pm)

2.5.1 Graph Neural Networks

Good recipe for Neural Networks : Convolutional NN shared weights: parallelization, shift invariance.

Adapting it to graphs

Idea 1 : work at the node level to compute local aggregator *Idea 2 :* make computation scalable by weight sharing ("convolutional net")

Look at neighbors to aggregate and compute the features.

Generalization of CNN to Graph : see article ChebNet

- ChebNet : At each layer use a graph filter and learn parameters of the filter : polynomial filter for computations
- **GCN - Simplified architecture :** Only use linear filters because using multiple layers has "similar" effect Use same weights for nodes and their neighbors Rescale for stability and have a more uniform effect across the graph

Some examples of the matrix weights of GNNs:

1. $W = UD_\theta(\Lambda)U^\top$.
2. $W = UG_\theta(\Lambda)U^\top$.
3. (Chebnet?) $W = P_\theta(L)$.
4. Simplification: $W = \theta_1\mathbf{I} - \theta_2 A$ (can be simplified even more by setting $\theta = \theta_1 = -\theta_2$).

where the next features (neurons) are computed as: $F_i(x) = \sigma([W]_i^\top x + b)$.

Message Passing GNNs in GNN : Compute local features, transfer message to the neighbors, re-scale the neighbors, receive message from the neighbors to agglomerate/update your state. Repeating the operation is similar to receiving signals from further and further neighbors. This is what ChebNet does: it tries to learn how to transmit the message through the polynomial filters.

This approach can be generalized to non-polynomial filters. Three main steps :

- Compose message: $m^{t+1}(i \to j) = \mathcal{M}(f_t(x_i), f_t(x_j), a_{ij})$.
- Aggregate messages: $\mathcal{A}(m^{t+1}(j \to i), j \in \mathcal{N})$.
- Update states: $f_{t+1}(x_i) = \mathcal{U}(f_t(x_i), \mathcal{A}(m^{t+1}(j \to i), j \in \mathcal{N}))$.

To finalize this approach, depending on the task at hand : node or graph.

Attention-based GNNs : Graph Attention Networks (GATs), Non-Local Neural Networks (NLNNs) Use a self-attention mechanism t weight node features The attention can be generalized : there are weights to decide if at a certain level of the network we use the information at any graph even if they are not connected directly.

Unifying view of GNNs : Message Passing General properties arise from this general idea : symmetries see slides

Possibility to add constraints justified by the problem setting (eg: node equivariant message passing functions)

Applications :

- E-commerce
- Anonymize Networks
- Biology : Classify sites on proteins with their respective nodes (Pocket classification, interface prediction, Ultra-fast PPI search : learn proteins interactions and compatibility)

2.5.2 Analyzing GNNs

Check how expressive GNNs are : how good they can distinguish non-isomorphic graphs. Weisfeiler-Leman test : assign a color to nodes of each graph (same color for every one at beginning), update the color via an injective hash based on the previous color and the color of the neighbors of each node. Finally, compare color histograms.

2.6 Hour 6 (Friday 2-3 pm)

2.6.1 Understanding GNNs

WL Failure There are non-isomorphic graphs that cannot be distinguished by 1-WL (e.g. Decalin & Bencyclopentyl).

How does GNN compare to WL? Both are equivalent. If a GNN can distinguish two graphs, 1-WL can distinguish them as well. Conversely, if 1-WL can distinguish two graphs, with sufficient depth a GNN can distinguish them as well.

Takeaway The paper *How Powerful are Graphs Neural Networks* characterizes the kind of injective multisite functions.

2.6.2 Beyond GNNs: exploiting higher order structures

To distinguish two graphs, we can look for motives like triangles. This is what the k-WL test does. Question: what is the neighbourhood of a set of nodes? (k+1)-WL is more expressive than k-WL.

Simplicial Neural Nets We want to exploit higher-order structure in a more simplified way. The idea is to create a simplex and then keep increasing the structure. But the subsets of the higher structure must be contained by the lower level.

Definitions:

1. boundary (e.g. the boundary of a triangle are its edges)
2. boundary incidence
3. signed boundary matrices
4. k-th Hodge Laplacian ... We can prove that L_k is positive semi-definite.

Simplicial Weisfeiler-Leman We exploit richer structures of **adjacencies in simplicial complexes**:

- boundary adjacency
- co-boundary adjacency
- lower adjacency
- upper adjacency

We use aggregation over adjacent structures in WL fashion. It is not less powerful than 3-WL

2.6.3 Conclusion

There are limitations of GNNs that we do not understand yet. First problem: we can make it more expressive by adding more layers. Since we average the layers, the last layers, that are the most important to take meaningful decisions, get inexpressive.

Second problem: over-squashing.

Bibliography

Shipra Agrawal and Nikhil R Devanur. Bandits with concave rewards and convex knapsacks. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 989–1006, 2014.

Shipra Agrawal and Randy Jia. Optimistic posterior sampling for reinforcement learning: worst-case regret bounds. *Advances in Neural Information Processing Systems*, 30, 2017.

Shipra Agrawal, Vashist Avadhanula, Vineet Goyal, and Assaf Zeevi. Mnl-bandit: A dynamic learning approach to assortment selection. *Operations Research*, 67(5):1453–1485, 2019.

Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

Francis R. Bach and Eric Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate o(1/n). *CoRR*, abs/1306.2119, 2013. URL http://arxiv.org/abs/1306.2119.

Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 207–216. IEEE, 2013.

Peter L Bartlett and Ambuj Tewari. Regal: A regularization based algorithm for reinforcement learning in weakly communicating mdps. *arXiv preprint arXiv:1205.2661*, 2012.

David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2010.04.005. URL https://www.sciencedirect.com/science/article/pii/S1063520310000552.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Edoardo Di Napoli, Eric Polizzi, and Yousef Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Linear Algebra with Applications*, 23(4):674–692, March 2016. doi: 10.1002/nla.2048. URL https://doi.org/10.1002/nla.2048.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *arXiv preprint arXiv:1209.1873*, 2012.

David I. Shuman, Mohammad Javad Faraji, and Pierre Vandergheynst. A multiscale pyramid transform for graph signals. *IEEE Trans. Signal Process.*, 64(8):2119–2134, 2016. doi: 10.1109/TSP.2015.2512529. URL https://doi.org/10.1109/TSP.2015.2512529.