IFT 4031/7031,
Machine Learning for Signal Processing
Week 10: Graphs in Signal Processing and Machine Learning

Cem Subakan





UNIVERSITÉ LAVAL



Admin

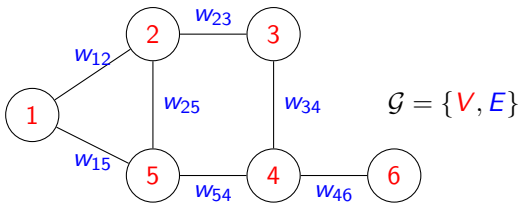
- Do not forget to fill in the project sign-up sheet! It's first come first serve.
  - N'oubliez pas le sign-up sheet pour les projets. C'est premier arrivé premier servi.
- There will be a last homework to be done in groups.
  - Il y aura un dernier devoir à faire en groupes!

Today's Lecture

- Graphs!
- Using Graphs instead of vectors / matrices
  - On va utiliser des graphs au lieu de vecteurs / matrices
- Graph Signal Processing Basics
  - Les bases du traitement du signal avec des graphs
- Graph Machine Learning
  - De l'apprentissage automatique avec des graphs

Graph

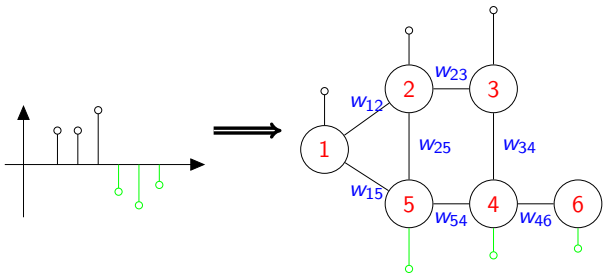
- A graph is a set of vertices (nodes) and edges.



- $\mathcal{G}$: Graph, V: Vertices (or Nodes), E : Edges
- Each node will have features / message. / Chaque node aura des features.
- Each edge has a weight / Chaque edge peut avoir un poids.
- Graphs might be directed or undirected. / Undirected: $1 \rightarrow 2 = 2 \rightarrow 1$
  - Les graph peut etre directionnel ou non.
- Graphs model **pairwise** relationships.
  - Les graphs modèle des relations **par paires**.

Representing a Signal as a Graph

- Associate each sample of the signal to a node. / On va associer chaque échantillon du signale avec un vertice.
  - Example: $x = [0.5, 0.5, 0.8, -0.4, -0.6, -0.3]$.



- Signal values become node features. Why? What's the advantage? / Les valueurs du signal devient les features du node. C'est quoi l'avantage?
- This enables us to explicitly model specific dependence structures, and it's a good representation if irregular signals.
  - Ça nous permet de modéliser des structures des dépendences spécifiques, et est une bonne manière de modéliser des signaux irregulaires.

How to represent a graph though?

- We understand that $\mathcal{G} = \{V, E\}$ defines a graphs.
  - On comprend que les ensembles V, E définissent un graph complétement. Mais comment est-ce qu'on répresente un graph numériquement?
- We will use an adjacency matrix A of size $N \times N$, for graph with N nodes.
  - On va utiliser une matrice d'adjacency du taille $N \times N$ pour un graphe de N nodes.
- The element a_{ij} gives the weight of the edge ij. / a_{ij} donne le poids de l'edge ij.
- Undirected graphs have symmetric adjacency matrices, such that $A^\top = A$.
  - Les graphes undirecteds ont des matrices adjacency symmetriques.
- We will pack the node values in a vector $\mathbf{v} \in \mathbb{R}^N$. / On mets les features pour chaque node dans un vecteur $\mathbf{v}$.
  - $\mathbf{v}_i$ gives the feature value of node i. $\mathbf{v}_i$ donne le valeur du feature du node i.

Examples

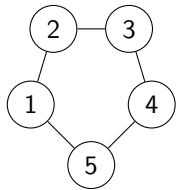
- Undirected Path Graph



$$A = \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \overset{to}{} \\ \end{matrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \end{mat

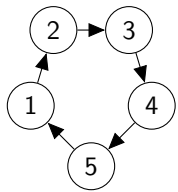
Examples

- Undirected Ring Graph



$$A = \begin{matrix} \\ \text{from} \end{matrix} \overset{\text{to}}{\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}}$$

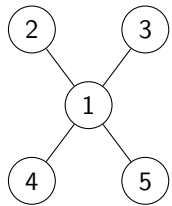
- Directed Ring Graph



$$A = \begin{matrix} \\ \text{from} \end{matrix} \overset{\text{to}}{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}}$$

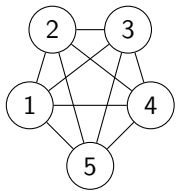
More Examples

- Undirected Star Graph



$$A = \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \overset{\text{to}}{\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}} \text{from}$$

- Undirected Fully Connected Graph



$$A = \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \overset{\text{to}}{\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}} \text{from}$$

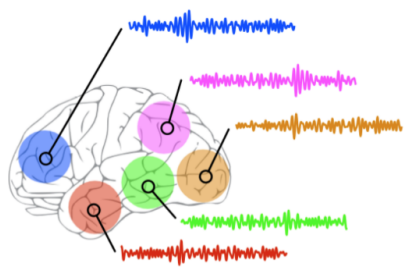
Real-World Examples

- Social Network Graph / Graphe d'un réseau sociale



Vertices: 1000 Twitter/X Users/Utilisateurs, **Edges:** Follower/not, **Node Features:** How many times did they tag IEEE MLSP 2023.

- Brain Measurements / Mésurements du cerveau



Vertices: Different regions , **Edges:** Structural Connectivity, **Node Features:** Brain activity measured as a 1d signal.

Graph Signal Processing

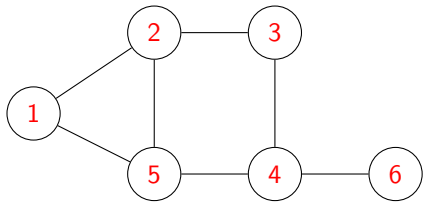
- How do we work with such representations? Comment est-ce qu'on travaille avec tel représentations?
  - Graph Signal Processing
- Classical DSP assumes 'regular' data (It's assuming a graph). / DSP classique suppose une régularité dans les données.
- So, GSP is a generalization of DSP. / GSP est la généralization de DSP.
  - This helps us to model more interesting covariance structures. / Ça nous permet de modéliser des covariances plus intéressentes.
- Also, GSP enables learning over graphs. / On peut aussi apprendre des modèles sur des graphes.

New things to worry about

- Classical Signal Processing
  - Time / Pixel Domain
  - We model relationships of samples laid on a regular grid / On modélise des relations sur un grille régulaire.
- Frequency Domain
  - Fourier Transform / DCT
- Graph Signal Processing
  - The graph domain.
  - Can model relationships described using an adjacency matrix. / On peut modéliser des rélations plus général défini par une matrice d'adjacency.
- Graph Spectral Domain
  - Graph Fourier Transform
- How do we move from spatial to graph domain? / Comment on va aller du domain spatial à la domaine de graphe?
  - We will use the Graph Laplacian L. / On va utiliser la graph laplacian L.

The Graph Laplacian

- Consider this graph, with all weights equal to 1 / Considérons ce graph avec les poids égaux à 1.



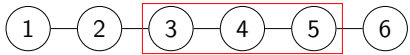
- $L := D - A$

$$\underbrace{\begin{bmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}}_{L} = \underbrace{\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{D = \mathrm{diag}(1^\top W)} - \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}_{A}$$

Graph Laplacian — Degree Matrix — Adjacency Matrix

But, what is this graph Laplacian?

- Undirected Path Graph



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

- The Laplacian

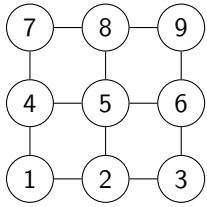
$$\underbrace{\begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}}_{L} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{D = \mathrm{diag}(1^\top W)} - \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}}_{A}$$

Graph Laplacian — Degree Matrix — Adjacency Matrix

This looks like the second derivative operator! (Discrete approximation to 2nd gradient) / C'est le noyau Laplacien, une approximation à la deuxième dérivée.

Grid

- Undirected Path Graph



- The Laplacian

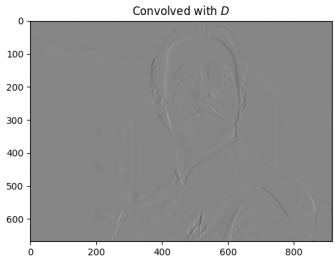
$$\underbrace{\begin{bmatrix} 2 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 3 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 3 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}}_{D = \mathrm{diag}(1^\top W)} \underbrace{\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}}_{A}$$

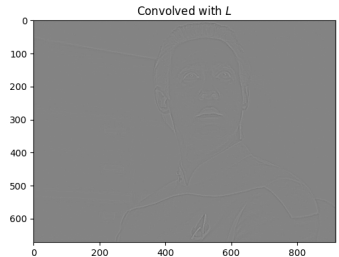
This looks like the second derivative operator! (Discrete approximation to 2nd gradient) / C'est le noyau Laplacien, une approximation à la deuxième dérivée.

Laplacian Operator on Images

Data is back!





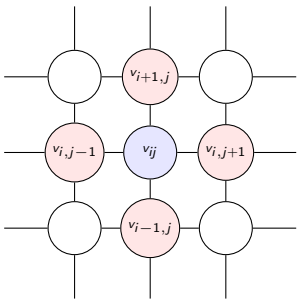


$$D = [1, -1] \quad L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

L is an edge detector here.

The Graph Laplacian Properties

- We approximate the Laplace operator on the graph with Graph Laplacian / On approxime l'opérateur Laplace avec le Graph Laplacian.



- For undirected graphs L is symmetric. / Pour des graphs undirected L est symmetrique.
- Off diagonals are non-positive. / Les non-diagonales sont non-positives.
- Rows add up to zero. / Les lignes s'ajoutent à zéro.
- There's also the normalized Laplacian, $L_{\text{norm}} = D^{-1/2}(D - A)D^{-1/2}$, but we'll stick to L for now. / Il existe aussi la Laplacienne normalizé.

The Graph Fourier Transform

- Given a Graph G containing a signal s, / Étant donné le graphe qui contient un signal s.
- Get Graph Laplacian L. / Obtiens la Laplancienne L.
- Do the eigenvalue decomposition on L. / Fait la décomposition éigen de L:

$$L = V \Lambda V^{\top}$$

- Transform the graph signal / Transforme le signal dans le graphe,

$$S = V^{\top} s$$

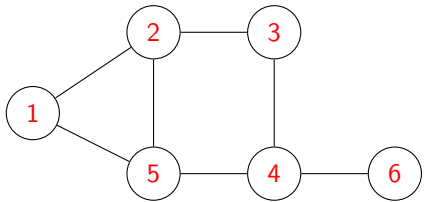
What does this mean?

- You can show that / On peut montrer que,

$$u^\top L u = \frac{1}{2} \sum_{i,j=1}^{N} \underbrace{w_{ij}}_{Adj.} \underbrace{(u_i - u_j)^2}_{\text{local variation}}$$

- What maximizes this measure of variation? / Que peut maximizer cette mésure de variation?
- $\max_u \; u^\top L u$ s.t. $u^\top u = 1$. What are these optimal u's? / Que sont ces u's optimales?
- Eigenvectors of u! / Vecteurs propres de u!
- Also, note that larger $u^\top L^\top u$ is, higher frequency we have in the signal u. / Notez que plus $u^\top L^\top u$ est large, plus le signale dans u est de haute fréquence.

Example



$$\begin{bmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

x	$x^\top Lx$
$[1,1,1,1,1,1]$	0
$[0,0,1,1,0,0]$	3
$[1,0,1,0,1,0]$	5

Low and High Frequency Graph Signals

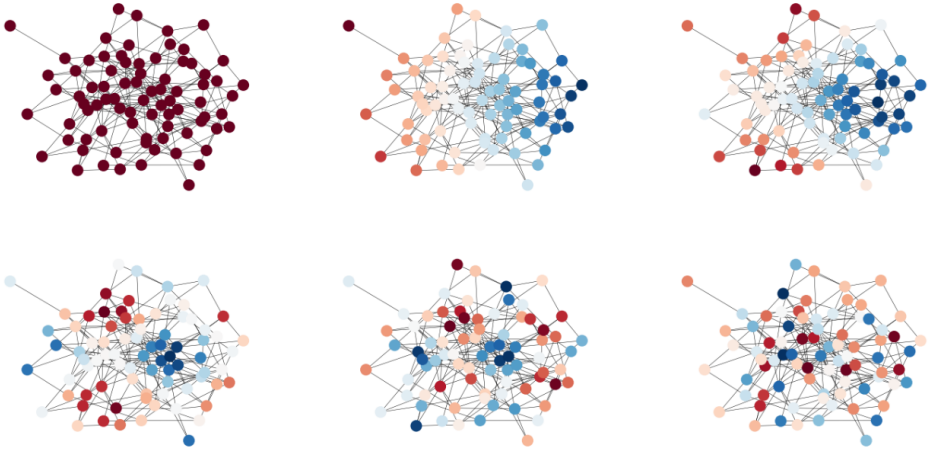
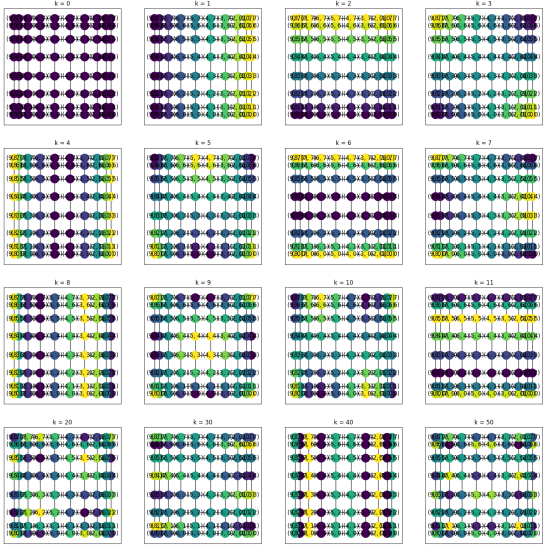
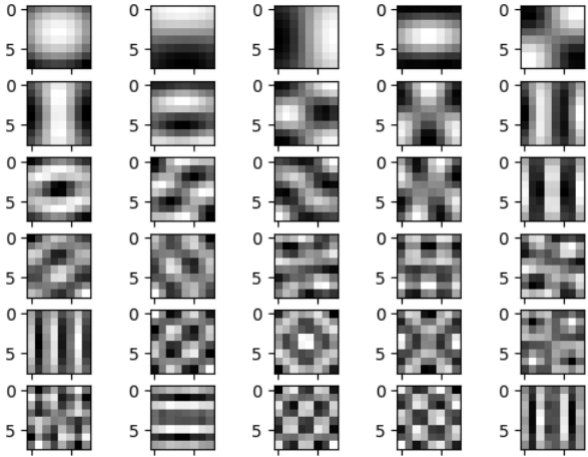


Image taken from UIUC MLSP class

Grid Graph



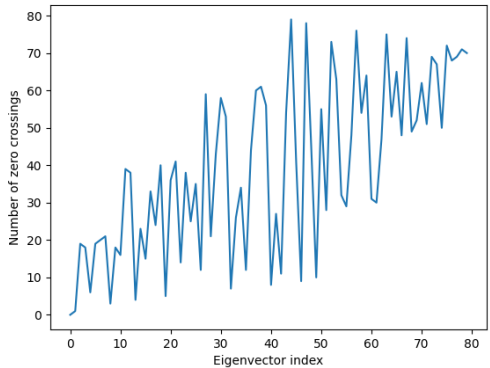
Do you remember this?



- It seems we were learning eigenvectors that correspond to a grid graph!
  - On apprenait des vecteurs propres qui correspondent à un graphe grille!

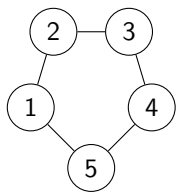
Eigenvalues as a measure of frequency

- The eigenvalues measure graph frequency
  - Les valeurs propres mesurent la fréquence de chaque base
- We can rank the eigenvectors by their eigenvalues / On peut mettre les vecteurs propres par leur valeurs propres
  - First eigenvectors are slow / change slowly. Les premiers vecteurs propres changent lentement
- The number of zero crossings for each eigenvector / le nombre de croisements zeros pour chaque vecteur propre:



Special Case: DFT

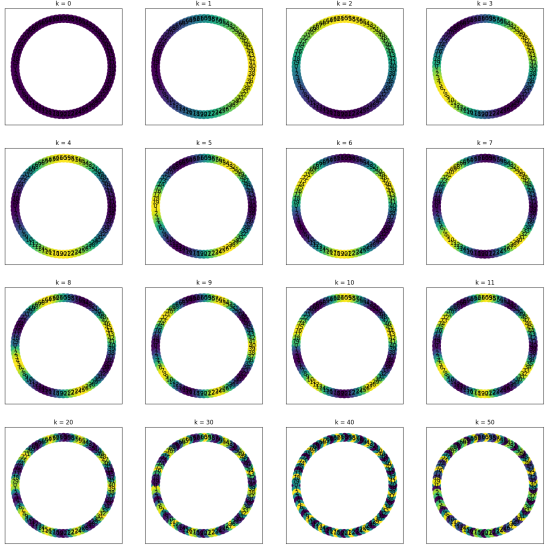
- Undirected Ring Graph



$$A = \begin{matrix} \\ from \end{matrix} \begin{matrix} \overset{to}{\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}} \end{matrix}$$

- Eigenvectors of L correspond to the Discrete Fourier Transform (DFT)! / Les vecteurs propres de L correspondent à DFT!

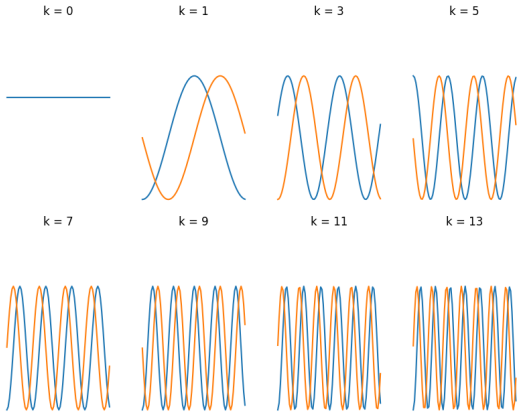
DFT



DFT Bases

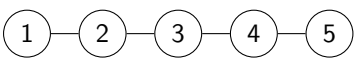
- Paired Sine / Cosine Bases,

$$u_k(n) = \exp(j2\pi kn/N) = \cos(2\pi kn/N) + j\sin(2\pi kn/N).$$



Special Case 2, Discrete Cosine Transform

- Undirected Path Graph

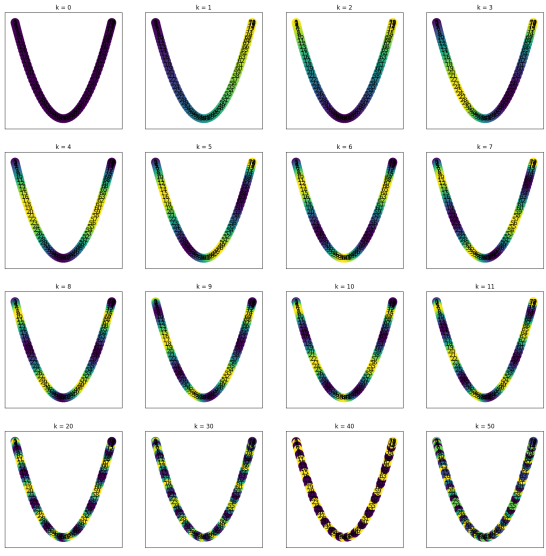


$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \textit{from} \end{matrix}$$

$$\textit{to}$$

- Eigenvectors of the Graph Laplacian will correspond to the DCT bases!
  - Les vecteurs propres du Laplacian correspondent aux bases DCT!

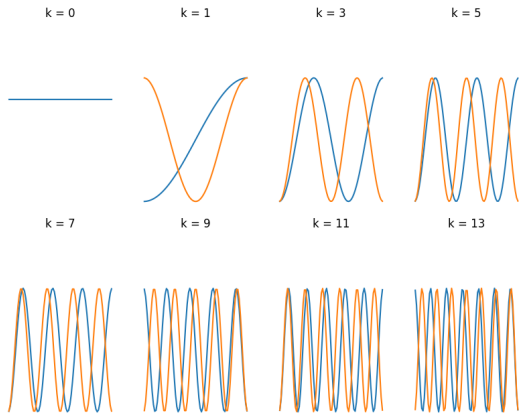
DCT



DCT-Type2 Bases

- Paired Sine / Cosine Bases,

$$u_{m,k}(n) \propto \cos((m + 0.5)k\pi/N).$$



Imposing a Graph

- Recap so far / Sommaire de ce qu'on a vu:
  - Graphs are collection of nodes and edges. / Un graphe est une collection des nodes et des edges.
  - A node is defined by an adjacency matrix A / Un graphe est défini par un adjacency matrix A.
  - Graph Laplacian $L = D - A$.
    - Is used to compute the Graph Fourier Transform. / Est utilisé pour calculer le Graph Fourier Transform.
    - This generalizes the classical Fourier Transform / Ça généralise le Fourier Transforme.
- We understand that standard SP assumes an underlying graph / Tout ça nous montre que le SP classique suppose un graphe.
  - Can we adjust the graph so that it's more suitable? / Peut-on adjuster le graph pour qu'il est plus ajusté?

Defining a Graph from a Signal

- We can define adjacency based on spatial location / On peut définir le voisinage en utilisant le localization spatiale, temporale

$$A_{ij}^t = \exp(-\alpha \| t_i - t_j \|)$$

- Or, we can define an adjacency based on sample values / Ou, on peut définir le voisinage en utilisant les valeurs
  - Do you remember this from somewhere else? / Souvenez-vous de ca d'ailleurs?

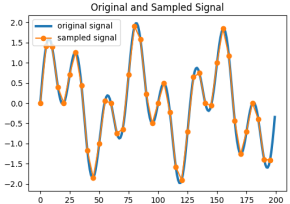
$$A_{ij}^x = \exp(-\beta \| x_i - x_j \|)$$

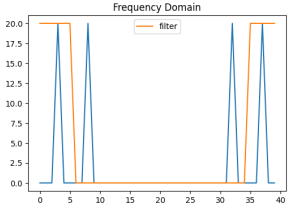
- We can define an overall adjacency matrix / On peut définir un matrice de voisinage ultime:

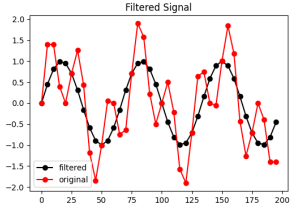
$$A = \gamma A^t + (1 - \gamma) A^x$$

Classical filtering

- $x_t = \cos(3\omega t) + \cos(8\omega t)$

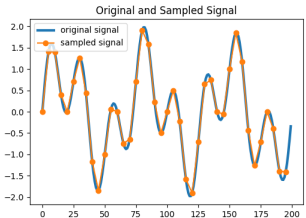


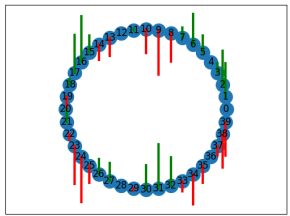




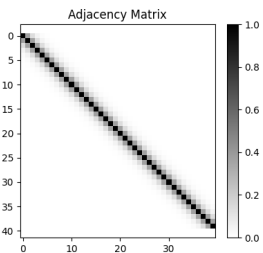
Let's do the same thing with Graphs

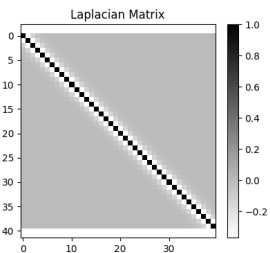
- Now we will define the same signal on a graph / On définit maintenant le meme signal sur un graphe





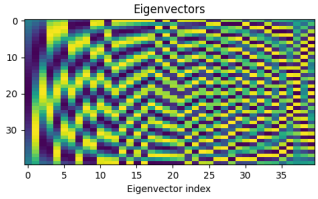
- Adjacency matrix, and the Laplacian (for a cycle graph)
  - $A_{ij} = \exp(-\alpha \|t_i - t_j\|)$

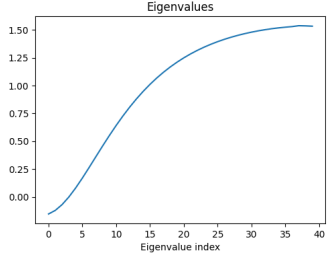


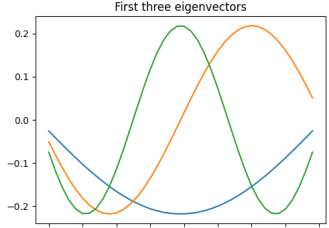


Step 1: Obtain the frequency bases of L

- $L = V \Lambda V^\top$







Graph Filtering Steps

- Transform onto the graph Laplacian eigenvectors.

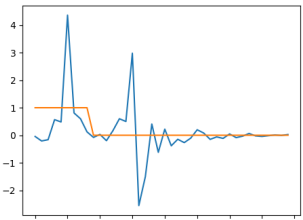
$$X = \underbrace{V^\top}_{\textit{Eigenvec.}} \underbrace{x}_{\textit{signal}}$$

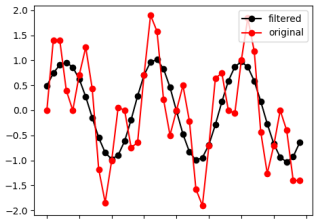
- Multiply with filter F

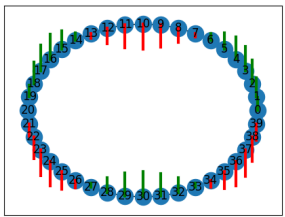
$$Z = F \odot X$$

- Reconstruct

$$\widehat{x} = VZ$$

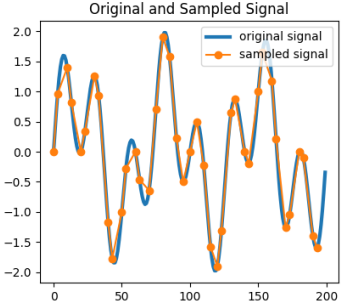


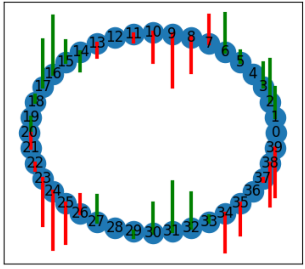




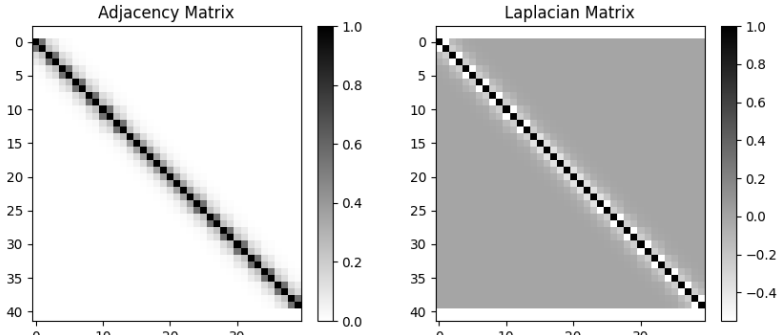
Nothing new with this, so why?

- What if the sample rate was not as regular / Et si le taux d'échantillonnage n'était pas si régulier?
- We can use a corresponding irregular graph maybe? / On peut utiliser la graph qui correspond?



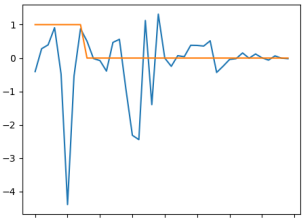


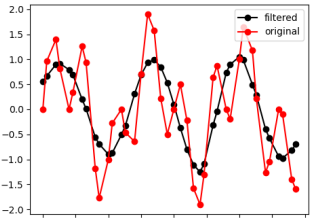
The new graph structure

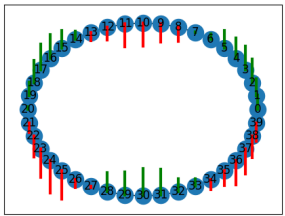


Non Uniform Graph Filtering

- We can still apply the same process as before / On peut encore appliquer le meme processus

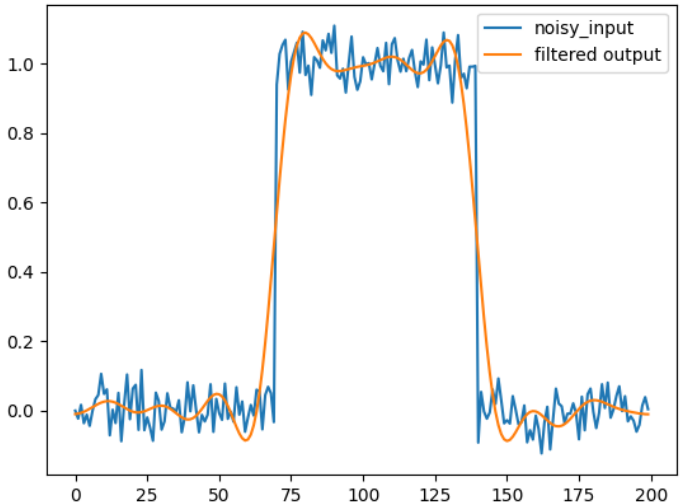






Encoding context from a signal

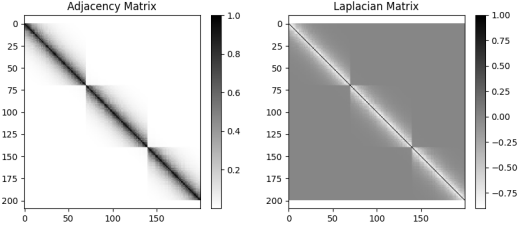
- Consider this noisy signal with steep edges / Considerons ce signal avec des sautes

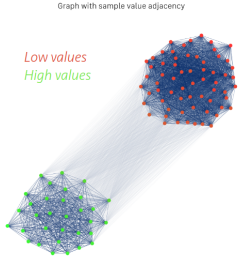


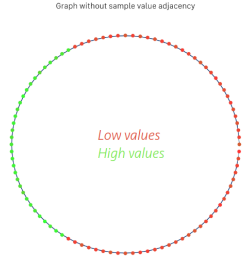
Using the signal in the adjacency

- Can we somehow encode that closer values are similar? / Peut-on encoder que les valeurs qui sont proches sont plus similaires?
- Adjacency matrix:

$$A_{ij} = \exp(-\alpha \|t_i - t_j\| - \beta \|x_i - x_j\|)$$

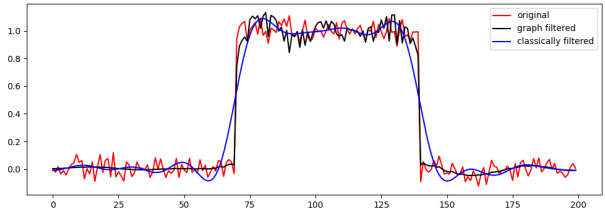






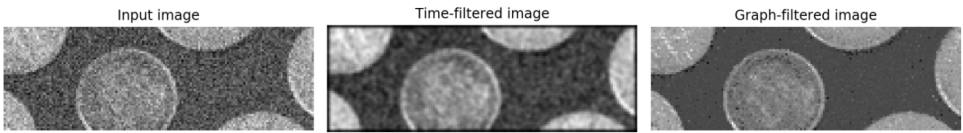
Edge preserving filter

- We can see that the graph filtering better preserves the edges / On voit que le filtre de graph mieux protège les sautes.



Edge preserving 2D example

- We can also apply the same thing on 2D signals / On peut appliquer la meme affaire sur des signaux 2D aussi.

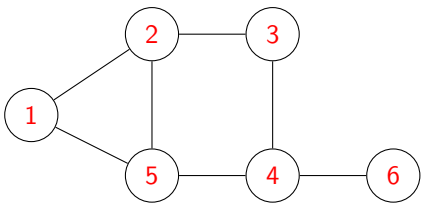


Defining Graphs from Data

- We already did this! / On a déjà fait ça
  - Eigendecomposition on an affinity matrix = KPCA/MDS/ISOMAP.
  - Clustering the eigenvectors = Spectral Clustering.

Filtering in the graph spatial domain

- In regular signals we have the frequency domain / time domain filtering duality. / Dans les signaux réguliers on a une dualité entre filtrage dans le domaine de temps / domaine de fréquence.
- Filtering in the time domain is defined by convolution which uses shifts. Filtrage dans le domaine de temps utilise des shifts.
- We can also define a shift operator on graphs. / On define a shift operator over graphs



$$S = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} = I + A$$

Graph Filtering / Convolution

- Graph filtering definition: $h * x = \sum_k h_k S^k x$
- For instance for $h = [1, 1, 0.5]^\top$, $x = [-1, 2, 0, 0, 0, 0]^\top$

$$S = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} = I + A$$

$$\begin{aligned} Sx &= [1, 1, 2, 0, 1, 0]^\top, \\ S^2x &= [3, 5, 3, 3, 3, 0]^\top, \\ y &= h_0 x + h_1 Sx + h_2 S^2 x \\ &= [1, 5, 2.5, 1.5, 2, 0]^\top \end{aligned}$$

- Notice that S defines a kernel with which we convolve! / Notez que S implique une sorte de noyau de convolution!

To summarize

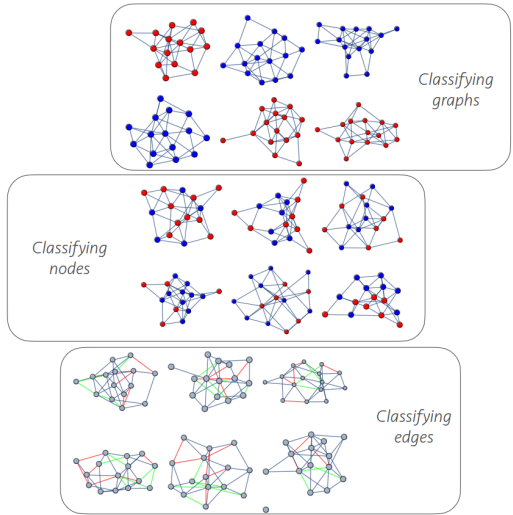
- Signals can be defined as graphs / Signaux peut-etre défini comme des graphs.
  - It enables us to represent more structured sample relations / Ça nous permet d'établir des relations plus structurés entre les échantillons.
- We can define analogs to classical DSP / On peut définir des analogues aux DSP.
- Now, let's briefly mention that we can do ML also! / On peut faire du ML aussi!

Graph Neural Networks

- Neural Nets that operate on graphs. / Neural Nets qui fonctionne sur des graphs.
  - Using the graph shift matrix, we can define CNNs, RNNs.. / On peut définir des CNNs / RNNs en utilisant l'opérateur graph shift.
- Useful for things like / Utile pour des choses comme:
  - Molécules, social graphs, mesh representations, LIDAR, ...
- Hot topic / Je ne suis pas expert du tout!

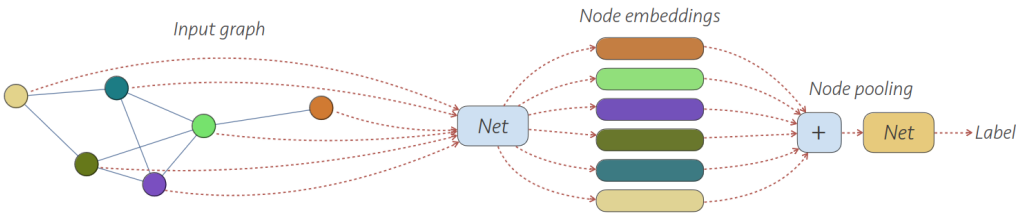
Types of Graph Level Processing

- Graph Level Processing, e.g. classifying a molecule
- Node Level Processing, e.g. LIDAR pixel classification
- Edge Level Processing e.g. Relationship classification in a social network



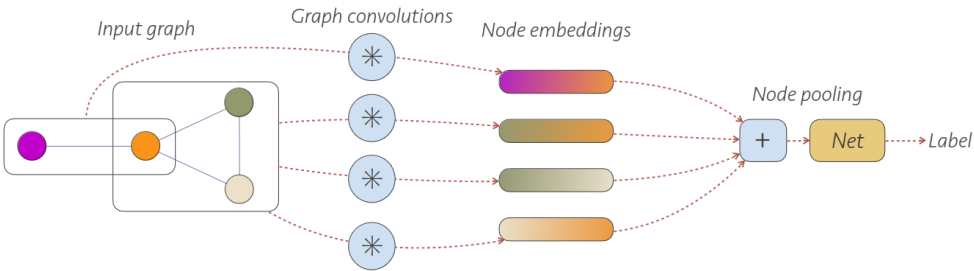
A very naive Graph Neural Network

- We can embed the nodes only / On embed les nodes seulement
  - Like you can do on your spectra in HW2-Q2 :)



Adding the graph structure

- We can also use graph convolutions to combine neighboring nodes when computing the node embeddings / On peut aussi inclure des convolutions graphes quand on calcule les node embeddings.
  - This enables us to model the graph structure / Ça nous permet de modéliser la structure de la graphe.



- Instead of using CNNs, it's possible to use RNNs / Attention Models, all that / Au lieu d'utiliser des CNNs, c'est possible d'utiliser des RNNs, modèles d'attention.

Graph Convolutional Network

- The graph convolutions (σ: a nonlinearity, e.g. ReLU)

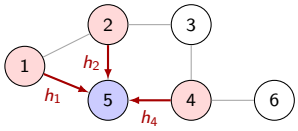
$$h_v^k = \sigma\Big(W^k \frac{\sum_{u \in \mathcal{N}(v)} h_u^{k-1}}{|\mathcal{N}(v)|} + B^k h_v^{k-1}\Big)$$

- This is simple to interpret. Incoming messages are averaged and transformed. And what is the k index? / C'est simple a interpreter, on moyonne les messages arrivants.
- The second term is kind of an MLP term. / La deuxieme terme est un genre de MLP.
- Note that if we want to express the transformation above with matrix multiplications, we can do the following: / On peut faire la meme chose avec des multiplications de matrices.

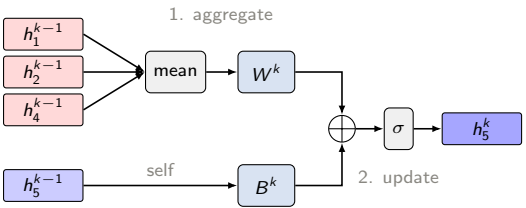
$$h^k = \sigma\big(\widetilde{W^k}\bar{A}h^{k-1}/(1^\top A)\big),$$

where $\bar{A} = A + I$, A being the adjacency matrix of the graph (size node by node), I is the identity matrix, and $\widetilde{W}^k$ combines W^k and B^k.

A GCN layer, as a diagram

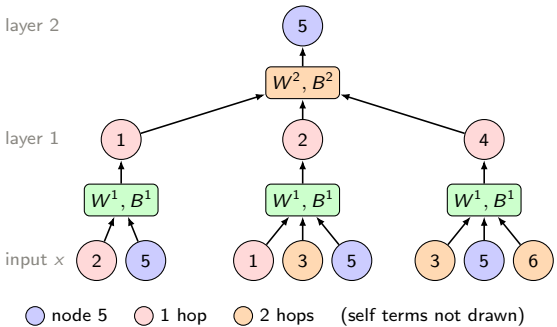


Update node $v = 5$,
neighbors $\mathcal{N}(5) = \{1, 2, 4\}$



- **Aggregate**: average the neighbors. / **Agréger**: moyenne des voisins.
- **Update**: add yourself, apply σ. / **Mettre à jour**: s'ajouter soi-même, appliquer σ.
- Every node at once, with the **same** W^k, B^k, like a CNN kernel. / Tous les nœuds, mêmes poids, comme un noyau de CNN.

Stacking layers: the computation tree



- 2 layers: node 5 hears from nodes 3 and 6, two hops away. / 2 couches = 2 sauts.
- Same color = same weights. / Même couleur = mêmes poids.
- Other graph, other tree, same weights: works on any graph. / Autre graphe, mêmes poids.

Why "convolution"? A GCN layer is a graph filter

Spectral filter (earlier)

$\widehat{x} = VFV^\top x$

× needs all eigenvectors
× not local

use S^k

Polynomial filter (earlier)

$y = \sum_k h_k S^k x$

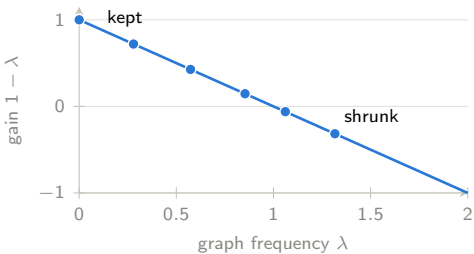
✓ no eigenvectors
✓ local: K hops

$K = 1$

GCN layer (Kipf & Welling, 2017)

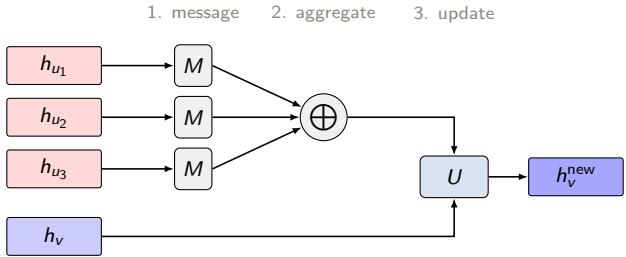
$H^k = \sigma(\hat{A}\, H^{k-1} W^k)$

$K = 1$, learned W^k, σ stack layers for more hops



- $\hat{A}$: $A + I$, normalized by degrees. It multiplies graph frequency λ by $1 - \lambda$. / $\hat{A}$ multiplie la fréquence λ par $1 - \lambda$.
- Dots: the 6 frequencies of our graph. / Points: les 6 fréquences de notre graphe.
- A GCN layer is a learnable **low-pass** graph filter. / Un filtre **passe-bas** appris.

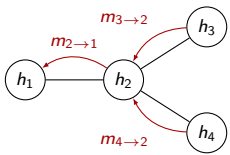
The general recipe: message passing



GCN: h_u mean $\sigma(Wm + Bh_v)$

- Neighbors have no order, so $\bigoplus$ must ignore it: sum, mean or max. / Pas d'ordre: somme, moyenne ou max.
- Every GNN is a choice of M, $\bigoplus$ and U. / Chaque GNN est un choix de M, $\bigoplus$, U.

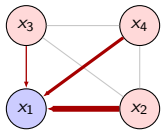
You have seen message passing before



Belief propagation (week 9)

messages = probability tables, fixed rules

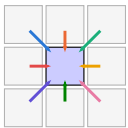
messages = tables de probabilités



Self-attention (week 7)

every token is a neighbor, weights = attention

chaque token est un voisin



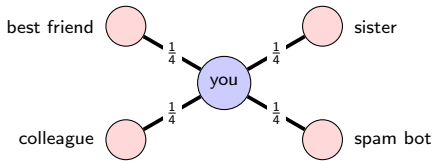
CNN: grid graph

neighbors have an order, so each position gets its own weight

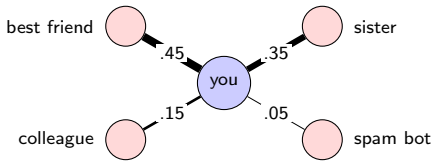
un poids par position

GNN: learned messages, on any graph. / Messages appris, sur n'importe quel graphe.

Not all neighbors matter equally: attention (GAT)



GCN: weights fixed by the graph



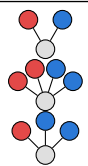
GAT: weights learned from the features

$$\alpha_{vu} = \underset{u \in \mathcal{N}(v)}{\text{softmax}}\big(\text{score}(Wh_v, Wh_u)\big), \qquad h_v^k = \sigma\Big(\textstyle\sum_u \alpha_{vu}\, Wh_u^{k-1}\Big)$$

- Score: a small learned layer (Veličković et al., 2018). / Score: une petite couche apprise.
- With a dot product, this is self-attention (week 7), on the neighbors only. A transformer is a GAT where every node is a neighbor. / C'est l'attention de la semaine 7, limitée aux voisins.

Mean and max can't count: use the sum (GIN)

- A node only sees the bag of its neighbors' vectors. / Un nœud ne voit que le sac des vecteurs de ses voisins.

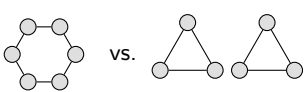
neighbors	mean	max	sum
	$(.5, .5)$	$(1, 1)$	$(1, 1)$
	$(.5, .5)$	$(1, 1)$	$(2, 2)$
	$(.33, .67)$	$(1, 1)$	$(1, 2)$

red $= (1, 0)$, blue $= (0, 1)$; shaded: different neighbors, same result

GIN (Xu et al., 2019): sum, then an MLP

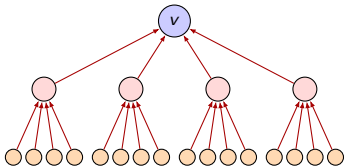
$$h_v^k = \mathrm{MLP}^k\left((1+\epsilon)\, h_v^{k-1} + \sum_{u \in \mathcal{N}(v)} h_u^{k-1}\right)$$

only the sum keeps the counts / seule la somme garde les comptes

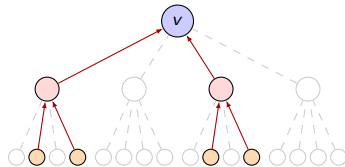


The limit: every node always sees two neighbors, so no message passing GNN can tell these apart. / Aucun GNN ne distingue ces deux graphes.

Too many neighbors: sample them (GraphSAGE)



all neighbors: $4^2 = 16$ inputs



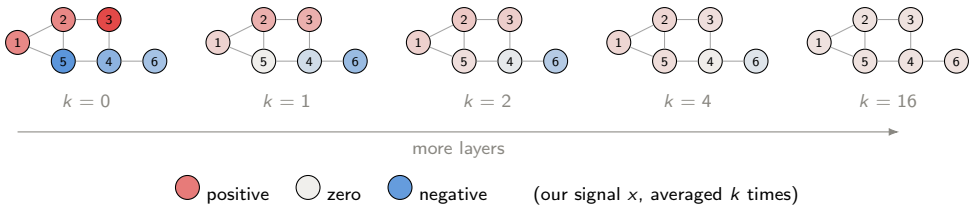
sample 2 per node: $2^2 = 4$ inputs

- Real graphs: 100 neighbors, 3 layers $\rightarrow 10^6$ inputs for one node. / 10^6 entrées pour un seul nœud.
- GraphSAGE (Hamilton et al., 2017): our GCN formula on a random sample of neighbors. / Notre formule GCN, sur un échantillon de voisins:

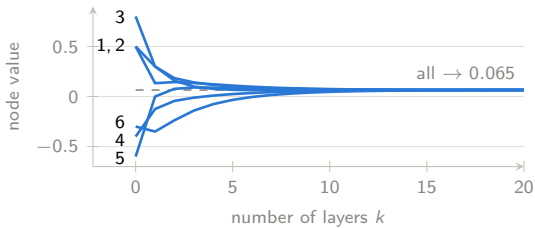
$$h_v^k = \sigma\Big(W^k \left[\, h_v^{k-1} \,\Big\|\, \text{mean}_{u \in \text{sample}}\, h_u^{k-1} \,\right]\Big)$$

- New node? Run the same function, no retraining. PinSage (Pinterest, 2018): 3 billion nodes. / Nouveau nœud: même fonction.

Going deeper: over-smoothing



- Each layer averages a node with its neighbors: a low-pass filter. / Chaque couche: une moyenne, un passe-bas.
- Deep ⇒ all nodes look the same. / Profond ⇒ tous les nœuds pareils.
- Fix: 2–4 layers, skip connections (week 7). / Solution: 2–4 couches, connexions résiduelles.

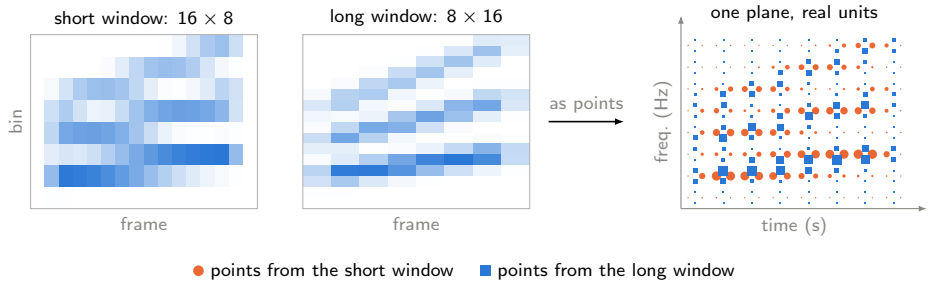


One recipe, different choices

	1. message	2. aggregate	3. update	fixes
GCN	h_u	mean	$\sigma(Wm + Bh_v)$	baseline
GAT	$\alpha_{vu} W h_u$	sum	$\sigma(m)$	unequal neighbors
GIN	h_u	sum	$\mathrm{MLP}((1+\epsilon)h_v + m)$	counting
GraphSAGE	h_u, u sampled	mean / max	$\sigma(W[h_v \,\|\, m])$	big graphs
Transformer	$\alpha_{vu} W^V h_u$, all u	sum	skip + FFN	no graph given

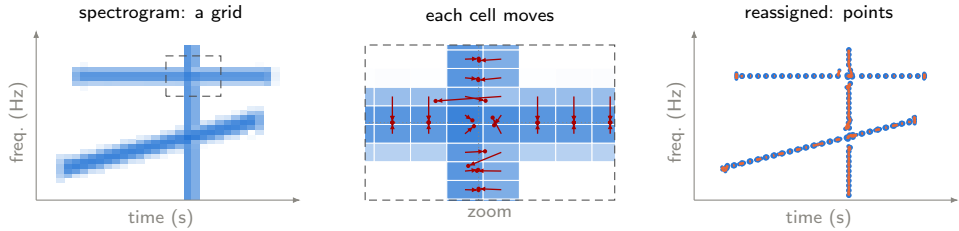
- Stack 2–4 layers, then read out: per node, sum over all nodes (whole graph), or $h_u^\top h_v$ (edge). / Puis une sortie par nœud, par graphe ou par arête.

Audio example: a spectrogram is a grid... or a set of points



- The same sound gives two images of different shapes. A CNN trained on one cannot read the other, and the same happens when the sample rate changes. / Même son, deux images de tailles différentes.
- As points with real units (seconds, Hz), both fit in one plane: just more or fewer points. / En points (secondes, Hz), tout tient dans le même plan.

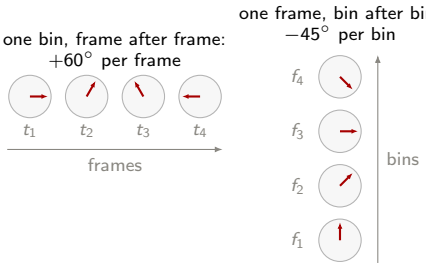
Reassigned spectrograms: from cells to points



synthetic sound: a chirp (80→280 Hz), a 400 Hz tone, a click at 0.62 s

- A spectrogram smears each sound over a block of cells. / Le spectrogramme étale chaque son sur plusieurs cases.
- Reassignment moves each cell's energy to its center of gravity (Kodera et al., 1976; Auger & Flandrin, 1995). / Chaque case va au centre de gravité de son énergie.
- Sharp points, off the grid: a point cloud $(\hat{t}, \hat{f}, m)$. / Des points nets, hors grille.
- Link each point to its 2 nearest neighbors: the edges follow each sound. A graph, ready for a GNN. / Les arêtes suivent chaque son: un graphe.

How does a cell know where to go? Its phase



$$\hat{f} = \frac{1}{2\pi}\frac{\partial\varphi}{\partial t} \qquad \text{instantaneous frequency}$$

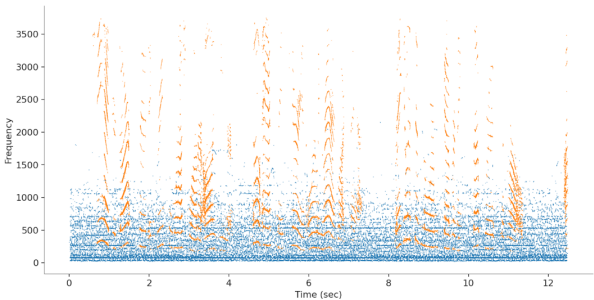
$$\hat{t} = t - \frac{1}{2\pi}\frac{\partial\varphi}{\partial f} \qquad \text{group delay}$$

φ: phase of the STFT cell (window-centered)

- The spectrogram keeps $|X|$ and throws the phase away. Reassignment reads it back: how fast it turns gives the frequency, how it shifts across bins gives the time. / La réassignation utilise la phase.
- In practice, no derivatives: two extra STFTs with windows $t\,h(t)$ and $h'(t)$. / En pratique: deux STFT de plus.
- Tones and clicks land exactly on their line; two sounds in the same cell blur together. / Exact pour un son pur ou un clic.

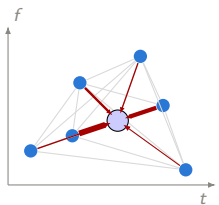
A spectrogram as a point cloud

- Using point clouds for a spectrogram, e.g. a re-assigned spectrogram / Nuage de points pour un spectrogram, p.ex. réassigné

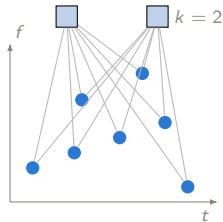


- Instead of having series of vectors, we have a point cloud / on a un nuage de points au lieu de vecteurs: $C = [(f_1, t_1), (f_2, t_2), \ldots, (f_N, t_N)]$.
- Each point also carries its magnitude: (t, f, m), with f in Hz, not a bin index. / Chaque point: (t, f, m), avec f en Hz.
- The order of the points means nothing, so the model must not depend on it, like the GNN aggregation. / L'ordre des points ne veut rien dire.

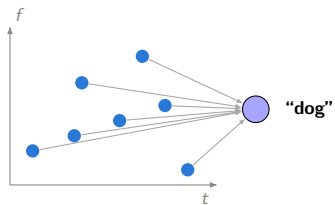
From points to a graph



every point ↔ every point
weights = attention, n^2 edges



through k learned hubs
$k \cdot n$ edges

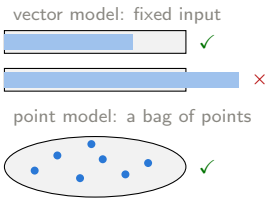


pool → one vector
then classify

- Subramani & Smaragdis (WASPAA 2021): nodes = points (t, f, m), edges = attention, a transformer with no positional encoding (a Set Transformer). It is the "GAT on the complete graph" from before. / Nœuds: les points; arêtes: l'attention.
- Another option (not in the paper): connect each point to its k nearest neighbors in (t, f), then use a GCN or a GAT. / Autre option: les k plus proches voisins.

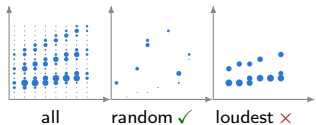
What points buy you, and what they cost

Any window, any sample rate



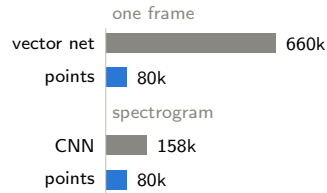
Trained at one setting, tested at others: the point model keeps working, the vector model cannot even take a longer window.

Keep 10% of the points



10% of the points, chosen at random: very small drop in accuracy. Random beats loudest: the model also needs to know where there is *no* energy.

Smaller models



Trainable parameters. Accuracy: better or similar.

- The price: attention between points costs 100–1000× more operations per iteration, so training is slower. Training on a random 1/16 of the points halves the training time and costs about 1% accuracy. / Le prix: 100 à 1000× plus d'opérations par itération.

Recap

- Signals are graphs! It's better to be aware of that to be able to generalize. / Les signaux sont des graphs en tout cas! Si on voit ça, on peut generaliser.
- Signal processing with Graph Perspective / Traitement du Signal avec un Perspective de Graphe
  - The Graph Fourier Transform
- Graph Neural Networks
  - Useful in many structured data domains (e.g. drug design) / Utile dans plusieurs données avec structure
  - One recipe, message passing: GCN (a learnable low-pass graph filter), GAT (attention), GIN (sum), GraphSAGE (sampling). Keep them shallow (over-smoothing). / Une recette, le passage de messages; peu de couches.

Reading Material

- Graph Signal Processing: https://arxiv.org/pdf/1712.00468.pdf
- GNN tutorial: https://distill.pub/2021/understanding-gnns/
- Point Cloud Audio Processing, Subramani & Smaragdis (WASPAA 2021): https://arxiv.org/abs/2105.02469
- Reassigned spectrograms: Auger & Flandrin, "Improving the readability of time-frequency and time-scale representations by the reassignment method", IEEE Trans. Signal Processing, 1995
- GCN, Kipf & Welling (ICLR 2017): https://arxiv.org/abs/1609.02907
- GraphSAGE, Hamilton et al. (NeurIPS 2017): https://arxiv.org/abs/1706.02216
- GAT, Veličković et al. (ICLR 2018): https://arxiv.org/abs/1710.10903
- GIN, Xu et al. (ICLR 2019): https://arxiv.org/abs/1810.00826
- Message passing (MPNN), Gilmer et al. (ICML 2017): https://arxiv.org/abs/1704.01212
- Over-smoothing, Li et al. (AAAI 2018): https://arxiv.org/abs/1801.07606

Next and last week

- I will talk about speech/audio topics. Je vais parler sur speech/audio.