Graph Convolutional Network

Convolution on graph structure and applications

Arthur Fetiveau, Baptiste Bouillon, Ismail Majbar, Mansour Tchenegnon

M2 AIDN /University of Bretagne Sud, Vannes

December 10, 2019

Table of content

Section

Fundamental concepts

Definition

A graph is a pair $G = (V, E)$ where:

- V is a set of elements called vertices or nodes,
- and E is a set of couples (x, y) called edges, $x, y \in V$.

Adjacency

- Two nodes x and y are adjacent if $(x, y) \in E$.
- Also, two edges a, b are adjacent if $\exists x, y, z \in V$ such as $(x, y), (y, z) \in E$ with $a = (x, y), b = (y, z)$.

Incidence

Let $x \in V$ and $a \in E$.
x and a are incident if x is a vertex of a.

Fundamental concepts

Degree of nodes

The degree of a node represents the number of edges it is incident to.

Types of graph

A graph can be directed or undirected.
In a directed graph, two types of degrees are considered:

- in-degree representing the number of edges ending by the vertex,
- out-degrees representing the number of edges starting with the vertex.

Fundamental concepts

Examples of graphs

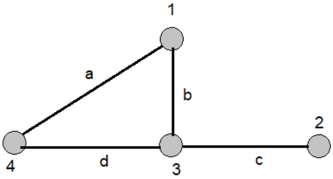


Figure 1: Undirected graph

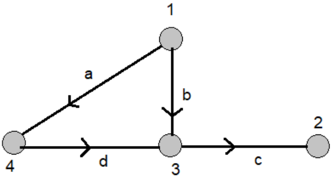


Figure 2: Directed graph

Adjacency matrix

Definition

Let a graph $G = (V, E)$ with $card(V) = N$. The order of the graph is then n. The adjacency matrix of G is define by $U = (u_{ij})$ of shape NxN such as:

$$u_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Example

Let's consider the previous undirected graph. The adjacency matrix is:

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

Degree matrix

Definition

The degree matrix of a graph is a diagonal matrix of nodes' degrees.
Given a graph $G = (V, E)$ with $card(V) = N$, the degree matrix of G is a NxN matrix defined by:

$$d_{ij} = \begin{cases} degree(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Example

Considering the previous undirected graph. The degree matrix is:

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

Laplacian matrix

Definition

Using the degree and the adjacent matrix of a graph, we can compute the Laplacian matrix.
The Laplacian matrix L_{NxN} is a the matrix defined by:
$L = D - A$
For an undirected graph, this matrix is symmetric.

Example

Considering the previous undirected graph. The degree matrix is:

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

Section

Why graph structure ?

Usefulness of graph

Connections are everywhere in the real world. Graphs are useful when representing real-world data where connections are important because, with graphs, the data and their relationships are of equal importance. The connections between things provide us with additional information, for example, a connection between two bank accounts.

Examples of graph applications

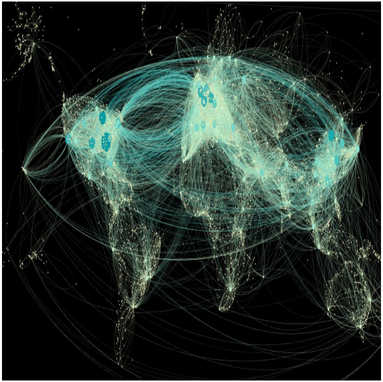


Figure 3: GlobeFlight

Source:
https://studentwork.prattsi.org/infovis/labs/visualizations-of-the-global-flights-network/

Examples of graph applications

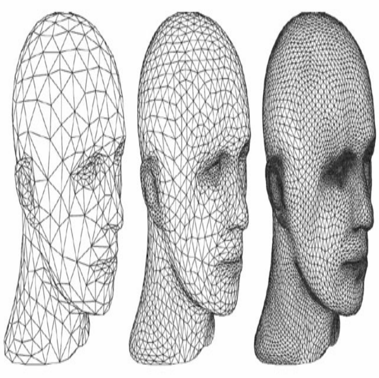


Figure 4: Computer-graphics

Source:
http://www-ljk.imag.fr/membres/Nicolas.Szafran/ENSEIGNEMENT/MASTER2/CS/cs-subdivision.pdf

Examples of graph applications

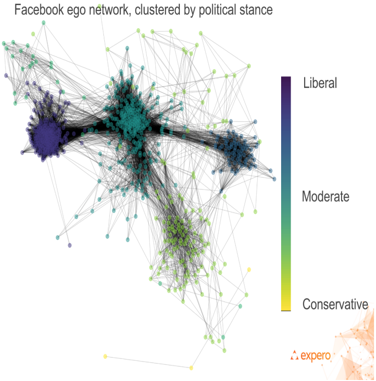


Figure 5: FacebookExemple2

Source: https://medium.com/expero/node-classification-by-graph-convolutional-network-5057faa10574

Examples of graph applications

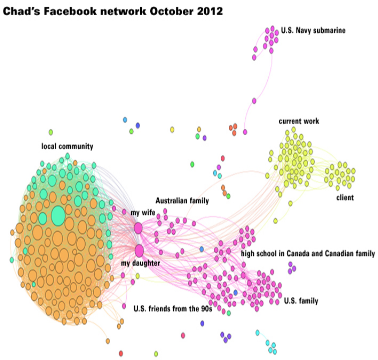


Figure 6: FacebookExemple

Source: https://www.sidewaysthoughts.com/blog/2012/10/facebook-social-network-analysis-with-gephi-mooc-musings-on-the-value-of-relationships/

Examples of graph applications

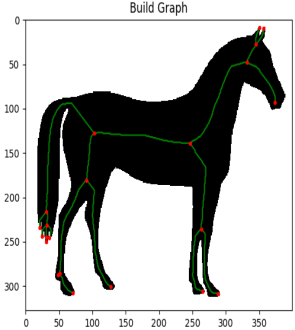
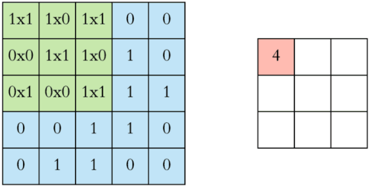


Figure 7: Graphic-Horse

Source: https://github.com/Image-Py/sknw

Section

Standard convolution



- 3x3 filter convolve the neighbourhood of a pixel
- each pixel is convolved

Computations

- each pixel has 9 neighbours
- structure in image

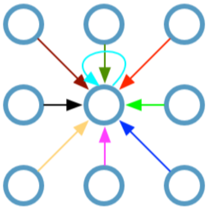


Figure 8: kernel in CNN

$$h_k^{(l+1)} = \sigma\left(\sum_{i=1}^{N} h_i^{(l)}.W_i^{(l)}\right)$$

(Kipf, CompBio Seminar, University of Cambridge, 2018)

Graph convolution

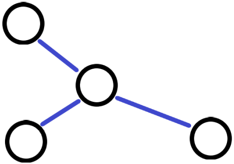


Figure 9: kernel in GNN

- each node has k neighbours
- weights are shared between nodes every iteration

$$h_k^{(l+1)} = \sigma\left(h_k^{(l)}.W^{(l)} + \sum_{j\in N_j} \frac{1}{c_{kj}}.h_j^{(l)}.W_1^{(l)}\right)$$

(Kipf, CompBio Seminar, University of Cambridge, 2018)

Graph convolution

Is it better or worst ?

Pros

- notion of distance between nodes
- each node is linked to his true neighbours
- learn from a structure

Cons

- must have the adjacency matrix
- in practice, small graphs

Graph convolution

Loss function

The loss of a graph:

$$L = L_0 + \lambda L_{reg}$$

$$L_0 = - \sum_{l \in \upsilon_L} \sum_{f=1}^{F} Y_{lf} . ln(Z_{lf})$$

$$L_{reg} = \sum_{i,j} \left(A_{ij} . \left\| f(X_i) - f(X_j) \right\|^2 \right)$$

- υ_L, a set of labelled nodes
- Z, the output of GCN
- $f(X_i)$, the prediction of GCN
- $A_{ij} = 1$ if i and j are connected

(Kipf and Willing, 2017)

Graph convolution

Update step

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right) = \sigma\left(\hat{A}H^{(l)}W^{(l)}\right)$$

- $\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$
- $\tilde{A} = A + I_N$, the adjacency matrix
- $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, the number of connections
- $W^{(l)}$, the weights at step l
- $H^{(0)} = X$, the dataset

(Kipf and Willing, 2017)

Spectral graph convolution

In our example, we have a graph with some features over each node. Those features can be defined as a signal x. Considering this, we want to do a spectral graph convolution. We defined spectral convolution on graph by a multiplication of a signal $x \in \mathbb{R}^N$ and a filter $g_\theta = diag(\theta)$ with $\theta \in \mathbb{R}^N$.

$$g_\theta * x = U g_\theta U^T x \tag{1}$$

Where U is the matrix of eigenvector of the normalized Laplacian $L = I_N - D^{-1/2} A D^{-1/2} = U \Lambda U^T$ (Λ is the diagonal matrix of eigenvalues), and $U^T x$ is the graph Fourier transform of x.

Chebychev polynomials approximation

Calculate the previous expression can be computationally expensive, so we can approximate this expression with the Chebyshev polynomials. We can now define the new filter g'_θ by a function of Chebyshev polynomials as follow :

$$g'_\theta(\Lambda) = \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda}) \tag{2}$$

Where $\tilde{\Lambda} = \frac{1}{\lambda_{max}}\Lambda - I_N$, λ_{max} is the largest element of Λ. And where $\theta' \in \mathbb{R}^K$ is the vector of Chebyshev coefficients.

Chebychev polynomials

The Chebyshev polynomials are a sequence of orthogonal polynomials. There are defined by the following recursive equation :

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \tag{3}$$

With $T_0(x) = 1$ and $T_1(x) = x$.

Spectral graph convolution approximation

Using the Chebyshev polynomials approximation, we can define the spectral convolution with the following expression :

$$g'_\theta * x = \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda})x \tag{4}$$

This approximation represent the K-th order convolution of the graph. To compute the convolution, we use up to the K-th step neihgbor to compute the convolution of a node.

Examples of graph convolutionnal network

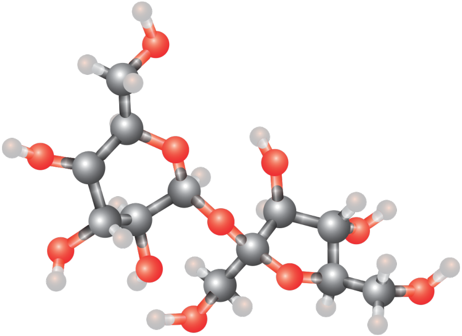
We've seen the theory for GCN on a semi-supervised example but it can also be used to recognize patterns into the graph to categorize it entirely. In an image for example. Graph convolutionnal network are used in a lot of applications using graph.

Graph convolutionnal network real applications :

- text classification
- semantic role labelling
- neural machine translation
- image classification

Examples of graph convolutional network

- molecular fingerprints
- protein interface prediction
- side effect prediction
- desease classification



Practical exercises presentation

In the practical exercises, you will work on the CORA dataset. CORA is composed of 2708 documents to classify. The graph is composed of 5429 edges representing the citations and each node has 1433 features corresponding in the presence or the absence of 1433 words.
Due to the data structure and the fact that it can be computationnally extensive to compute a full matrix filled by many 0, you may use scipy sparse matrix to complete the exercises.
The aim is to preprocess data first with local pooling and then with Chebyshev polynomials. You will also have to create a graph convolutional layer in Keras in order to do a semi-supervised classification on this dataset.

Section

Sources

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.

Thomas N. Kipf, Max Welling,SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS, September 2016

Jie Zhou, et al. Graph Neural Networks: A Review of Methods and Applications