Apprentissage pour les graphes – Partie 2

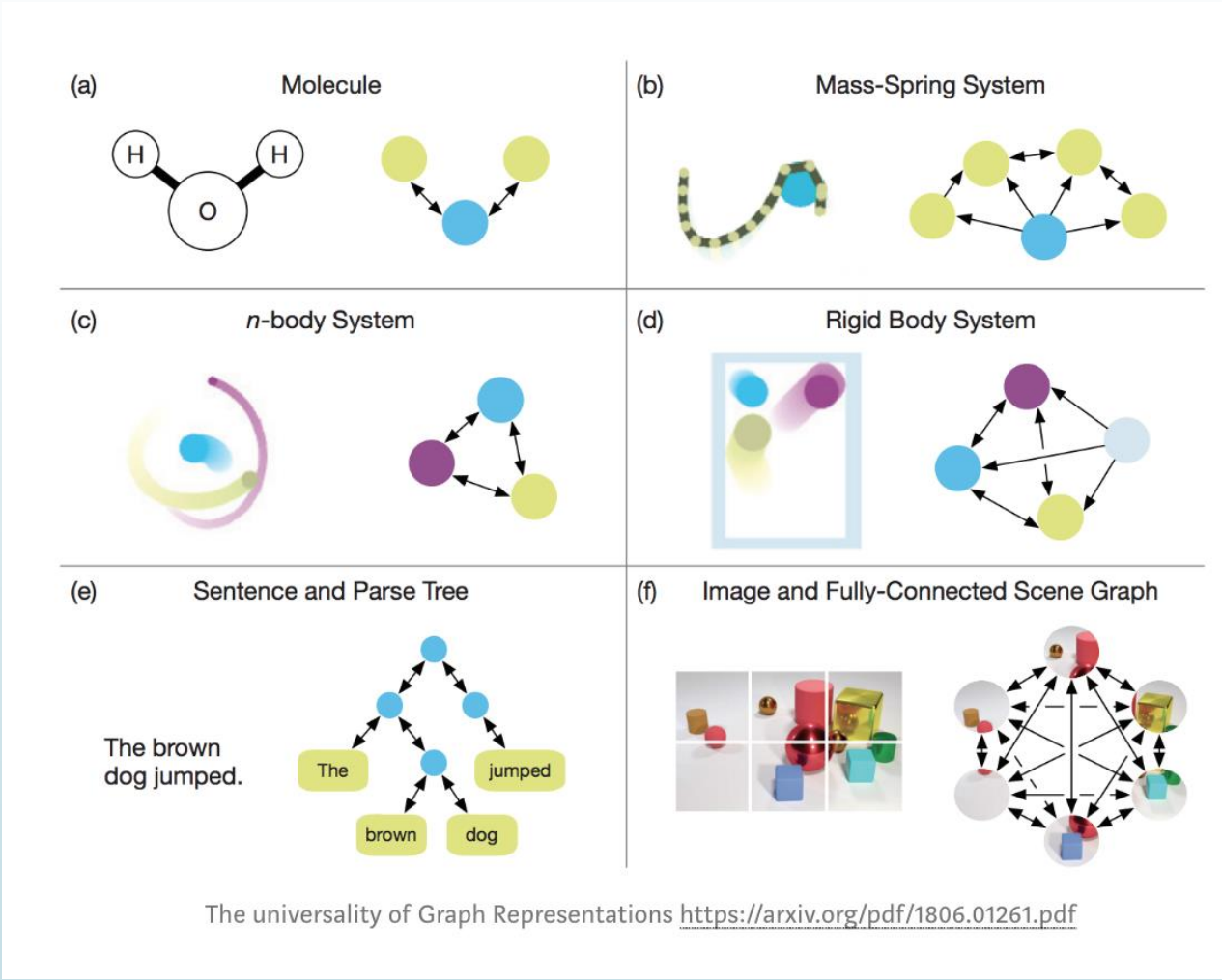
IFT 6758A – IFT 3700

Fall 2024

References

- Manuel sur les graphes et le ML : https://www.cs.mcgill.ca/~wlh/grl_book/
- Tutoriel sur les graphes et le ML: https://www.youtube.com/watch?v=fbRDfhNrCwo

Motivation: Universalité des graphes



The universality of Graph Representations https://arxiv.org/pdf/1806.01261.pdf

Encodage superficiel

- Approche d'encodage la plus simple : un plongement par noeud

$$\text{ENC}(v) = \mathbf{Z}\mathbf{v}$$

$\mathbf{Z} \in \mathbb{R}^{d \times |\mathcal{V}|}$ matrix, each column is node embedding [what we learn!]

$\mathbf{v} \in \mathbb{I}^{|\mathcal{V}|}$ indicator vector, all zeroes except a one in column indicating node *v*

From Shallow to Deep

- Nous allons maintenant discuter de méthodes « plus profondes » basées sur des réseaux de neurones pour les graphes.

$$\text{ENC}(v) = \text{complex function that depends on graph structure.}$$

- En général, tous les codeurs peuvent être combinés avec les fonctions de similarité qui dépendent de la structure du graphe.

Les bases: réseaux de neurones pour les graphes

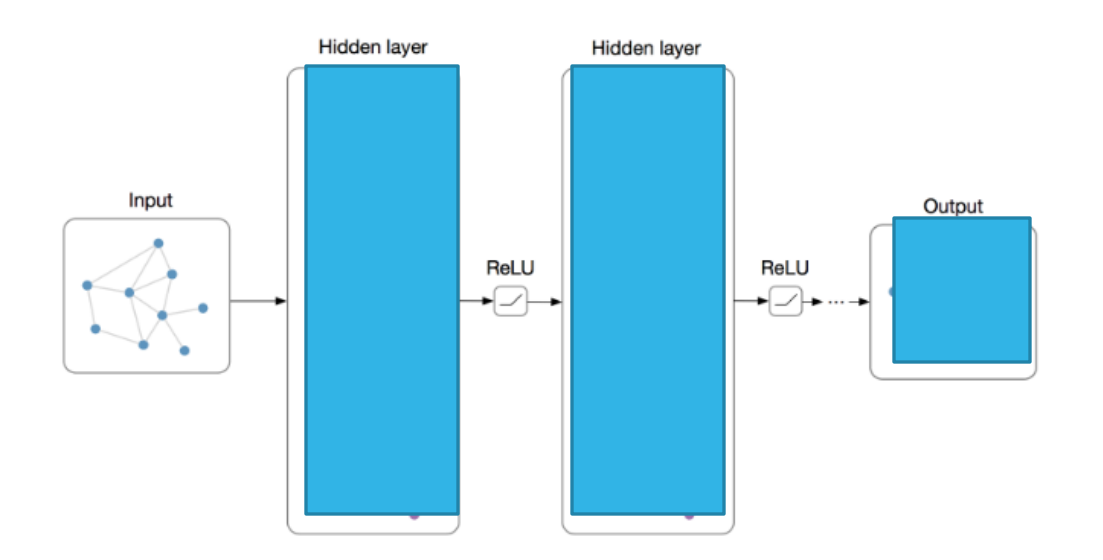
- Basé sur les papiers:
  - Hamilton et al. 2017. Representation Learning on Graphs: Methods and Applications
  - Scarselli et al. 2005. The Graph Neural Network Model

Notations

- Supposons que nous ayons un graphe G:
  - V est l'ensemble des sommets.
  - A est la matrice d'adjacence (supposons binaire).
  - X est une matrice de caractéristiques des nœuds.
- Attributs catégoriels, texte, données d'image
  - Par exemple, des informations de profil dans un réseau social.
  - Degrés de nœuds, coefficients de clustering, etc.
  - Vecteurs indicateurs (c.-à-d. codage à chaud de chaque nœud)

Une idée naïve : deep learning style

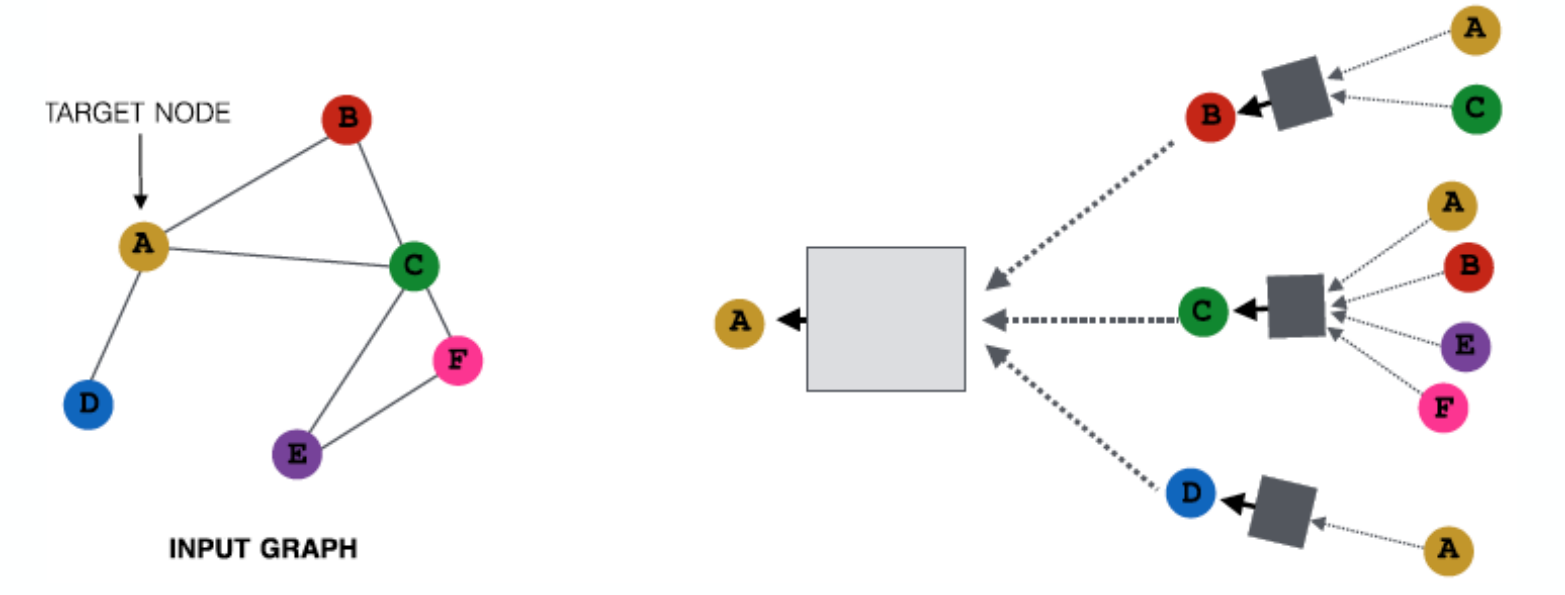
- Utilisez la matrice d'adjacence et transmettez-la à un MLP (perceptron multicouche)



- Problème : **dépend de l'indexation arbitraire des nœuds !!!**

Agrégation du voisinage

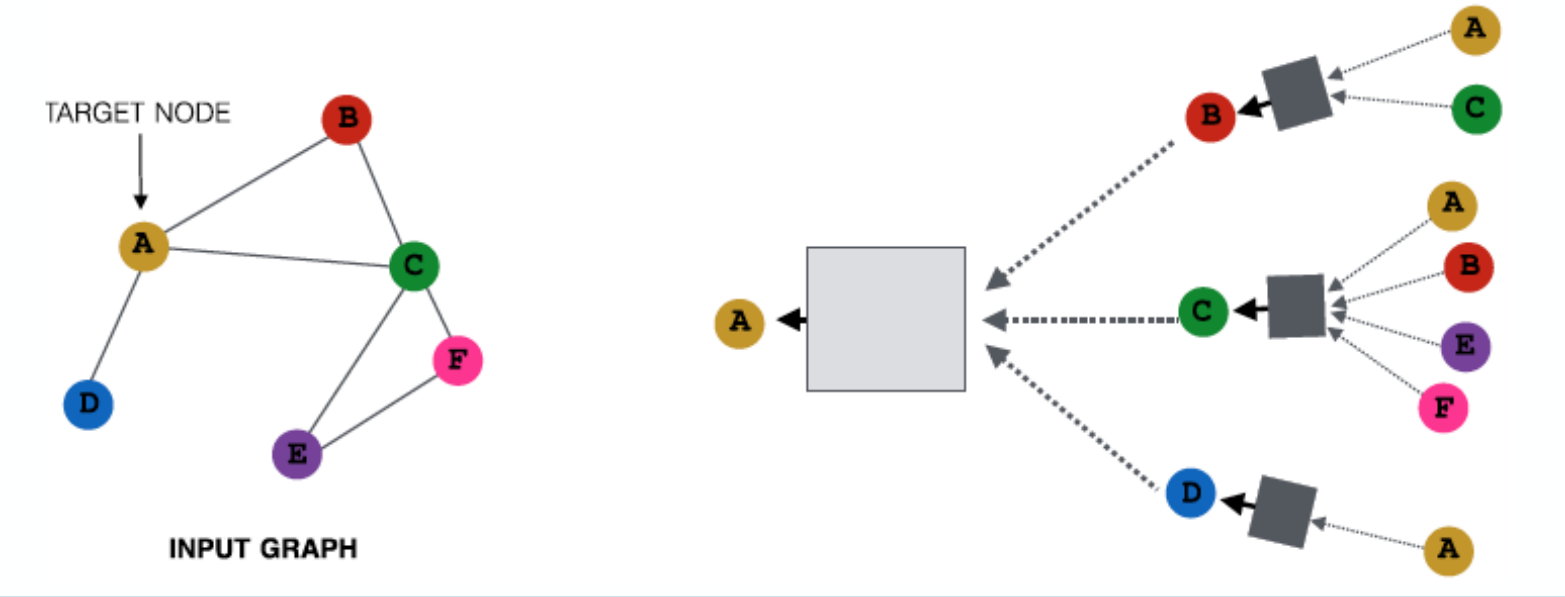
- **Idée clé :** générer des agrégation de nœuds en fonction du voisinage.



$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right)$$
$$= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right),$$

Agrégation du voisinage

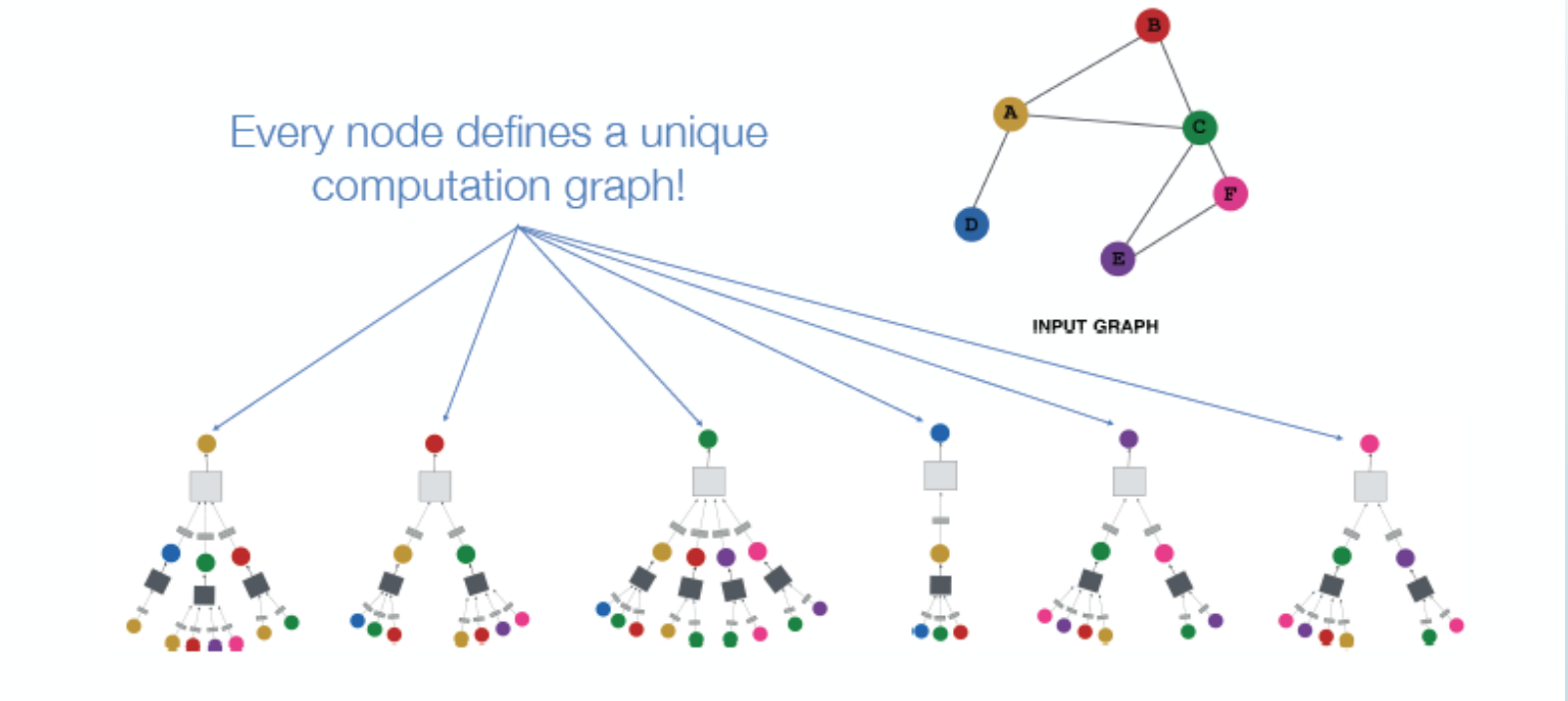
- Intuition : les nœuds agrègent les informations de leurs voisins à l'aide de réseaux neuronaux



$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right)$$
$$= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right),$$

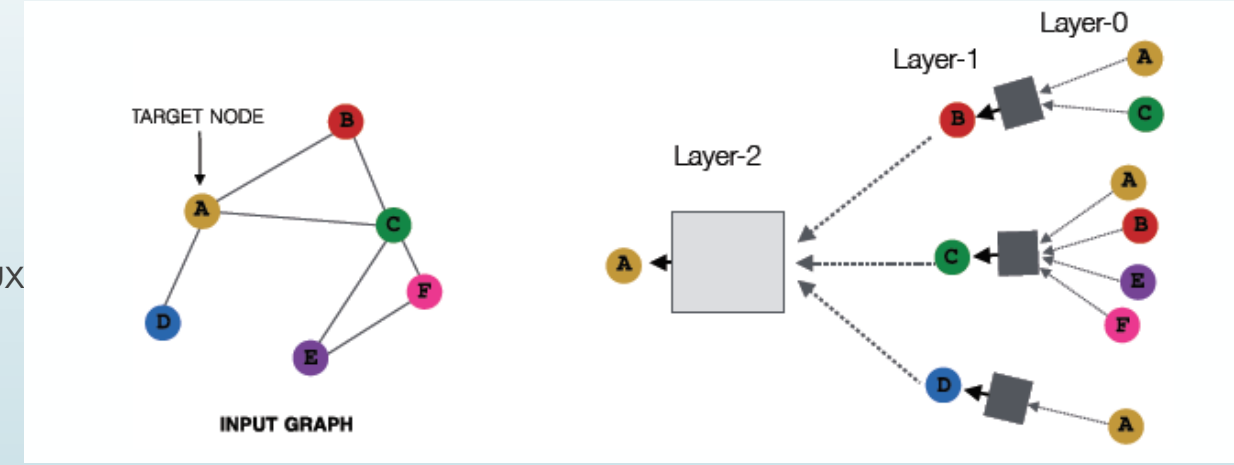
Agrégation du voisinage

- Intuition : Le voisinage dans le réseau définit un graphe de calcul!



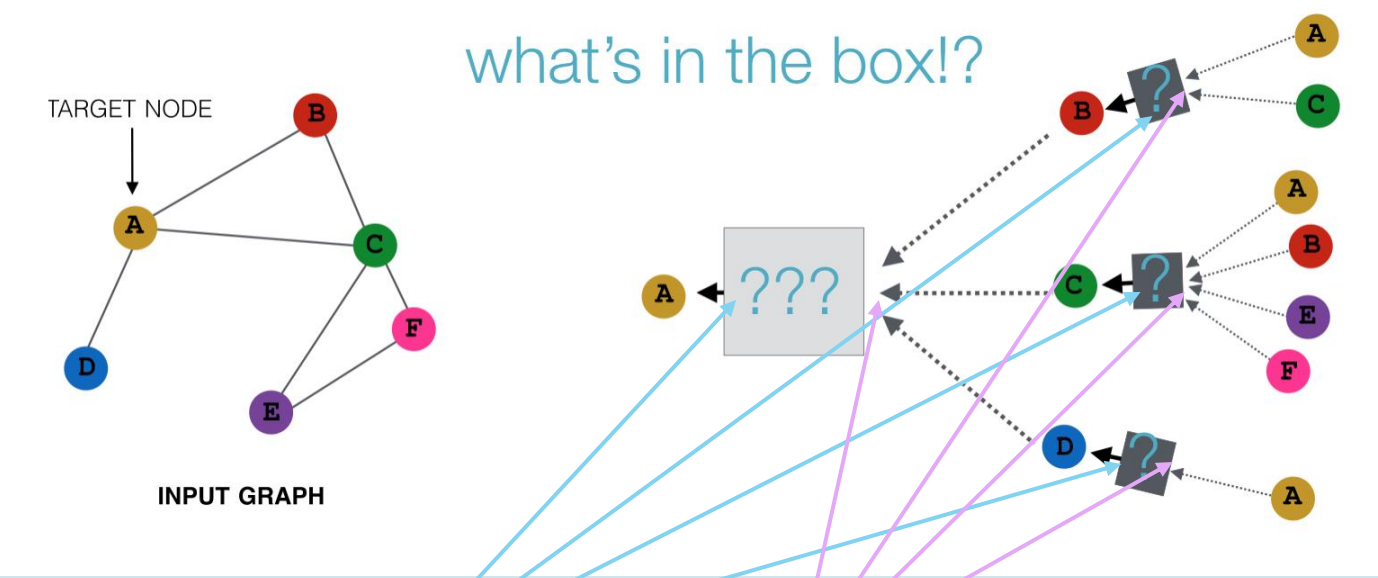
Agrégation du voisinage

- Les nœuds ont des **représentations à chaque couche.**
- Le modèle peut être d'une 'profondeur' arbitraire.
- La représentation initiale du nœud est ses caractéristiques.
- Contrairement aux méthodes superficielles, nous pouvons utiliser les caractéristiques des nœuds:
  - Caractéristiques supplémentaires des nœuds (caractéristiques d'expression génique dans les réseaux biologiques ou caractéristiques textuelles dans les réseaux sociaux)
  - En l'absence de caractéristiques de nœud supplémentaire:
    - Nous pouvons utiliser les statistiques de nœud (degré de nœud, centralité de nœud,... Voir : [livre])
    - Nous pouvons utiliser un encodage à chaud (ne peut pas généraliser aux nœuds invisibles ☹)



Agrégation du voisinage

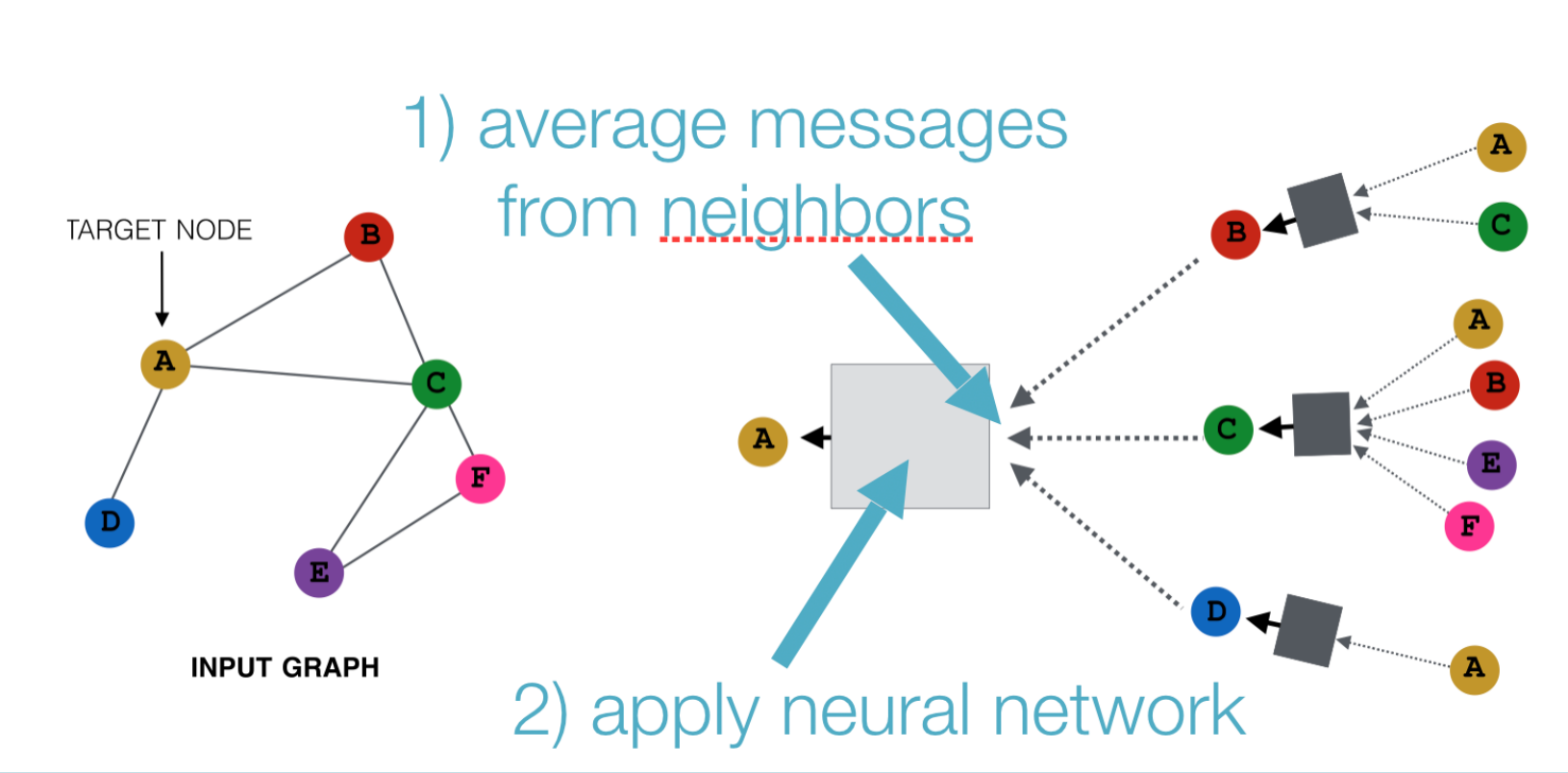
- Les principales distinctions résident dans la façon dont les différentes approches regroupent l'information à travers les couches.



$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right)$$
$$= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right),$$

Agrégation du voisinage

- Approche de base: Moyenne des informations de voisinage et appliquer un réseau neuronal.



Les Math pour le cas de base

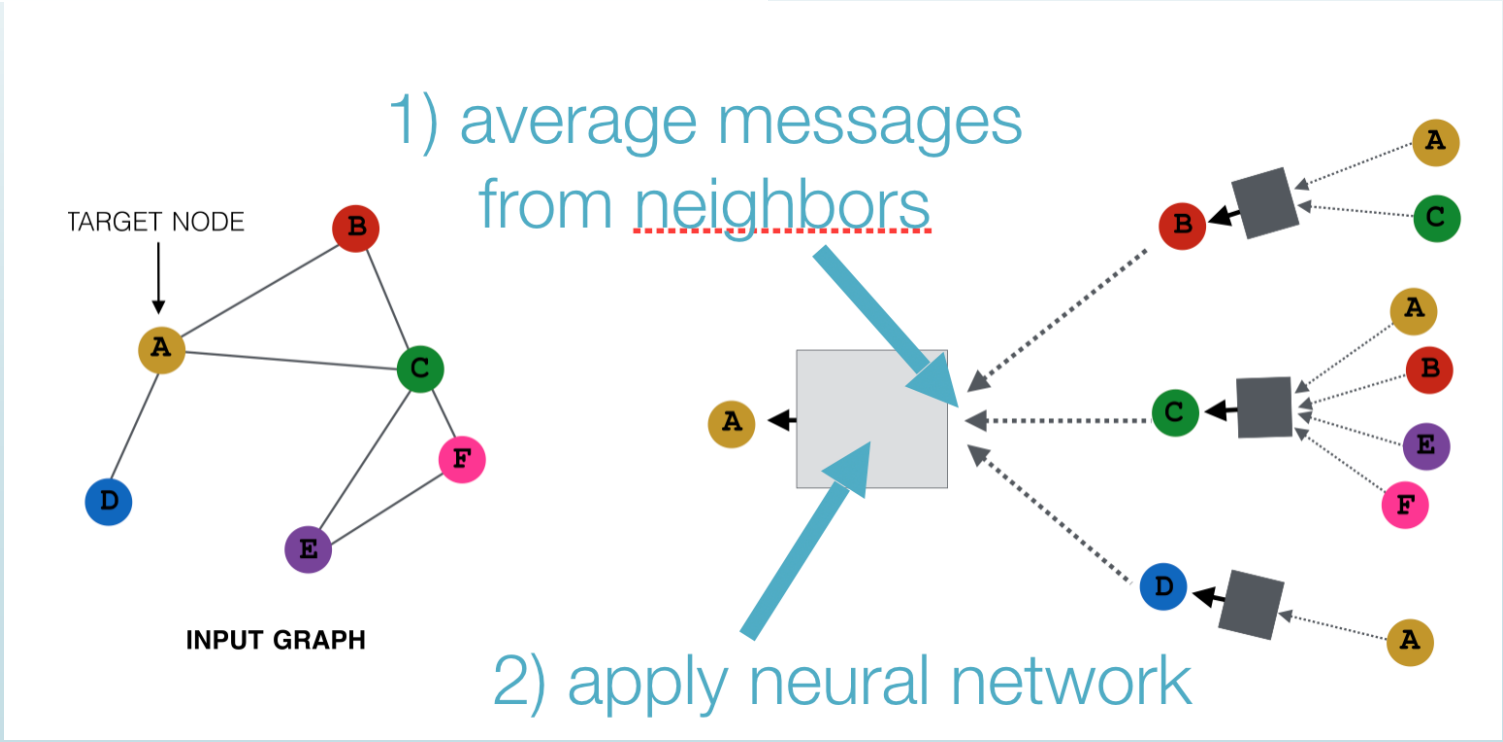
- Approche de base: Moyenne des messages voisins et application d'un réseau neuronal.

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

Initial "layer 0" embeddings are equal to node features

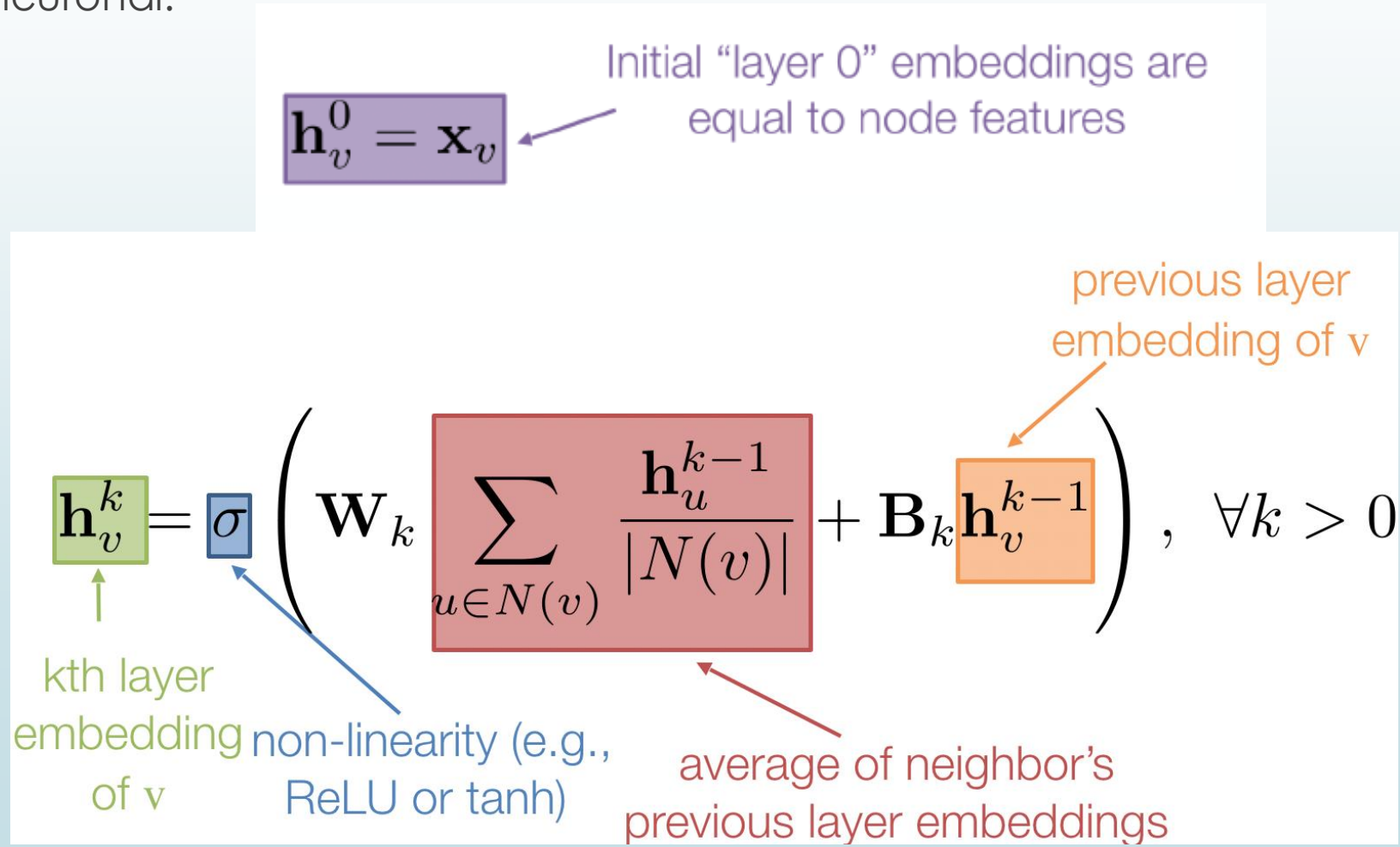
$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v,$$

$$\text{UPDATE}(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}) = \sigma\left(\mathbf{W}_{\text{self}}\mathbf{h}_u + \mathbf{W}_{\text{neigh}}\mathbf{m}_{\mathcal{N}(u)}\right),$$



Les Math pour le cas de base

- Approche de base: Moyenne des messages voisins et application d'un réseau neuronal.



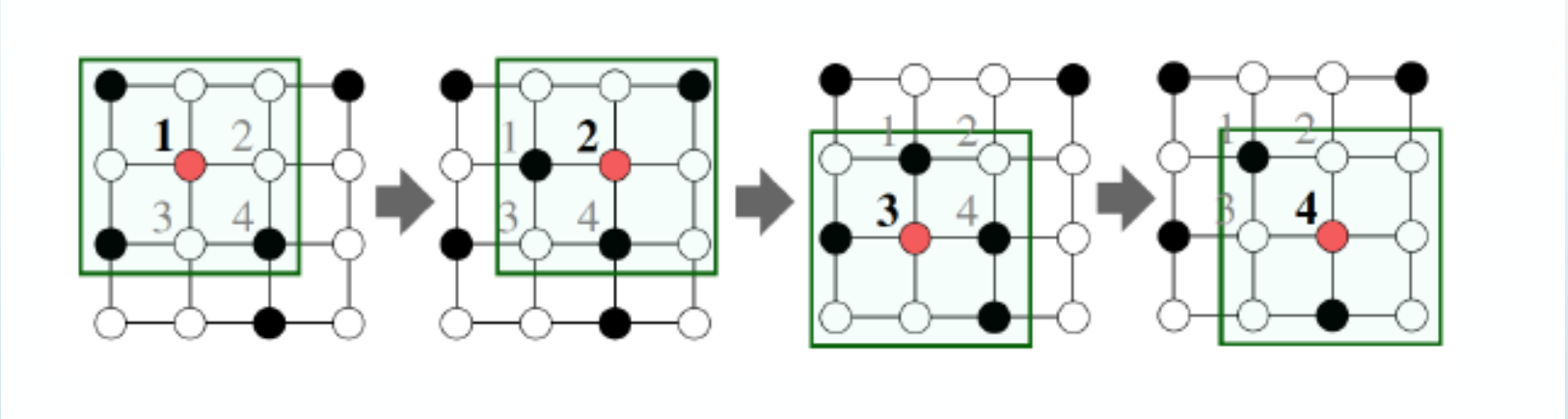
Motivations et intuitions

- **AGGREGATE et UPDATE** doivent conduire à des représentations **équivariantes par permutation** !

- Profondeur du réseau <-> Profondeur de la représentation du graphe
  - Première couche (k = 1), les représentations contiennent des informations provenant de son voisinage à 1 saut
  - Les representations de deuxième couche (k = 2) contiennent des informations provenant de son voisinage à 2 sauts
  - …
- Deux types d'informations au niveau de la k-ème couche :
  - Informations structurelles sur le graphe: (structure du voisinage k-hop)
  - Informations sur les caractéristiques : informations sur toutes les caractéristiques de leur voisinage k-hop

Voisinage pour les « Convolutions »

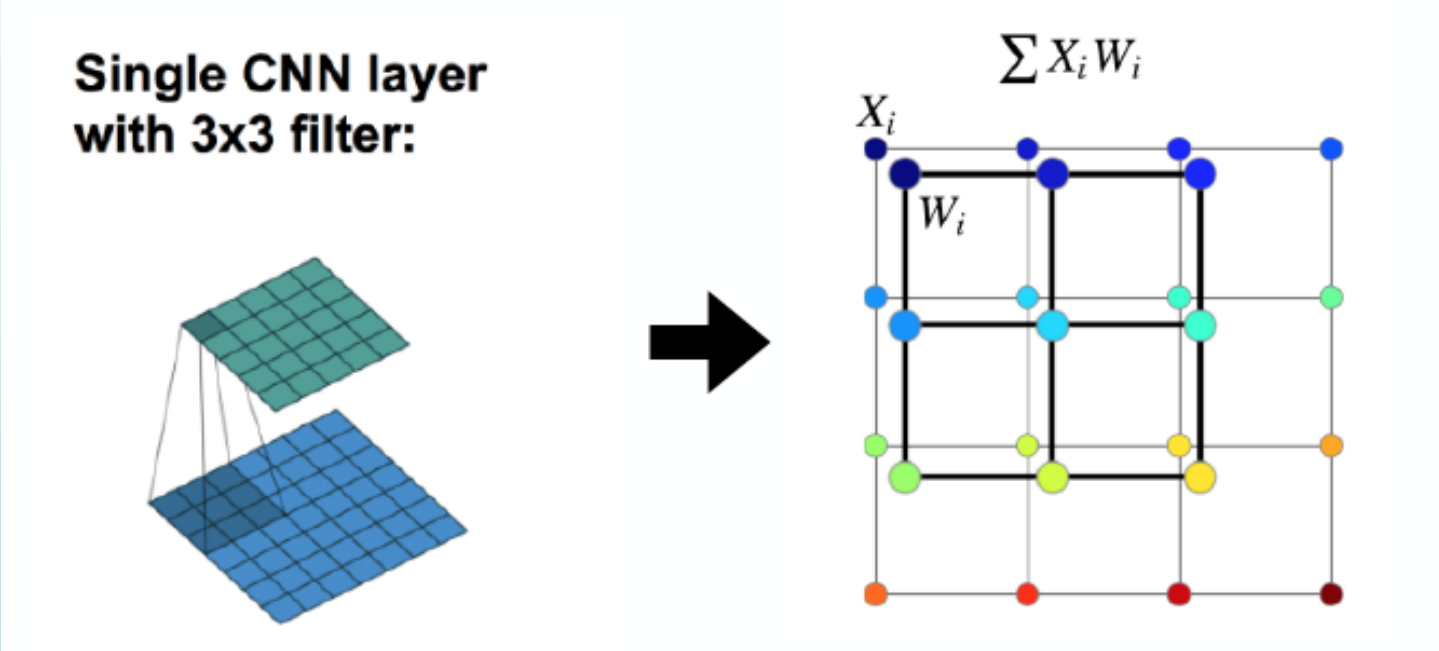
- L'agrégation de voisinage peut être considérée comme un filtre convolutionel.



- Mathématiquement lié aux convolutions spectrales pour les graphes (see Bronstein et al., 2017)

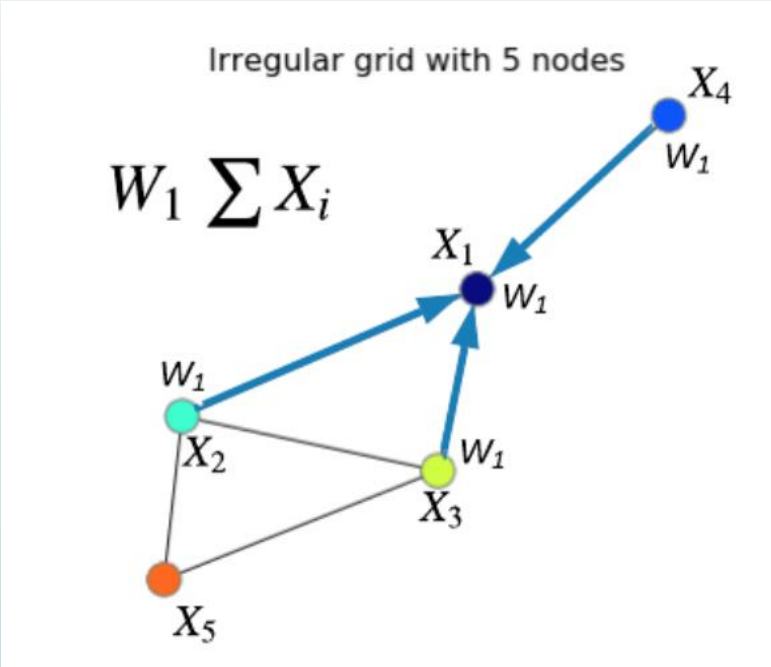
Convolution sur les images

- Convolution est un « opérateur agrégateur ». D'une manière générale, l'objectif d'un opérateur agrégateur est de résumer les données locales sous une forme réduite.



Convolution sur les graphes

- The most popular choices of convolution on graphs are averaging or summation of all neighbors, i.e. sum or mean pooling, followed by projection by a trainable vector W.



Motivations : invariance et équivariance

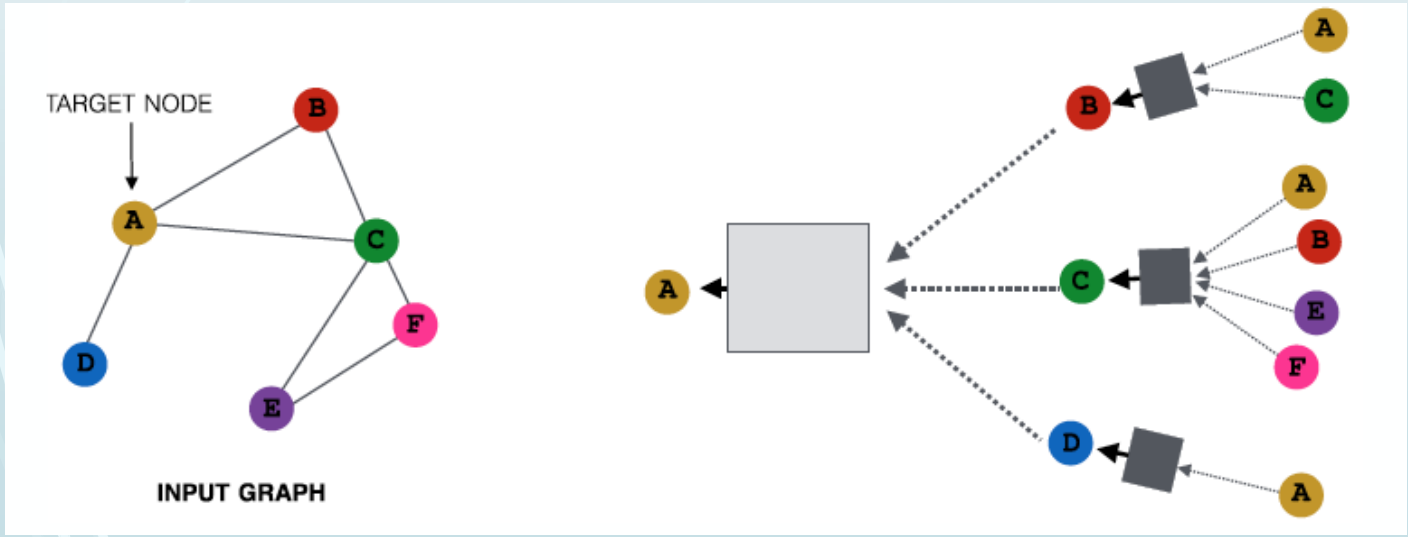
- A: Matrice d'adjacence
- P : matrice de permutation
- f : n'importe quelle fonction
- **Équivariant**
- **Invariant**

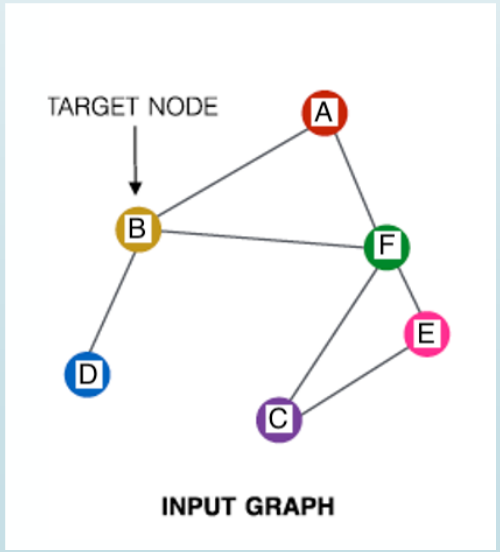
$$f(\mathbf{P}\mathbf{A}\mathbf{P}^\top) = f(\mathbf{A}) \qquad \text{(Permutation Invariance)}$$

$$f(\mathbf{P}\mathbf{A}\mathbf{P}^\top) = \mathbf{P}f(\mathbf{A}) \qquad \text{(Permutation Equivariance)}$$

Quand voulons-nous l'invariance et quand voulons-nous l'équivariance ?

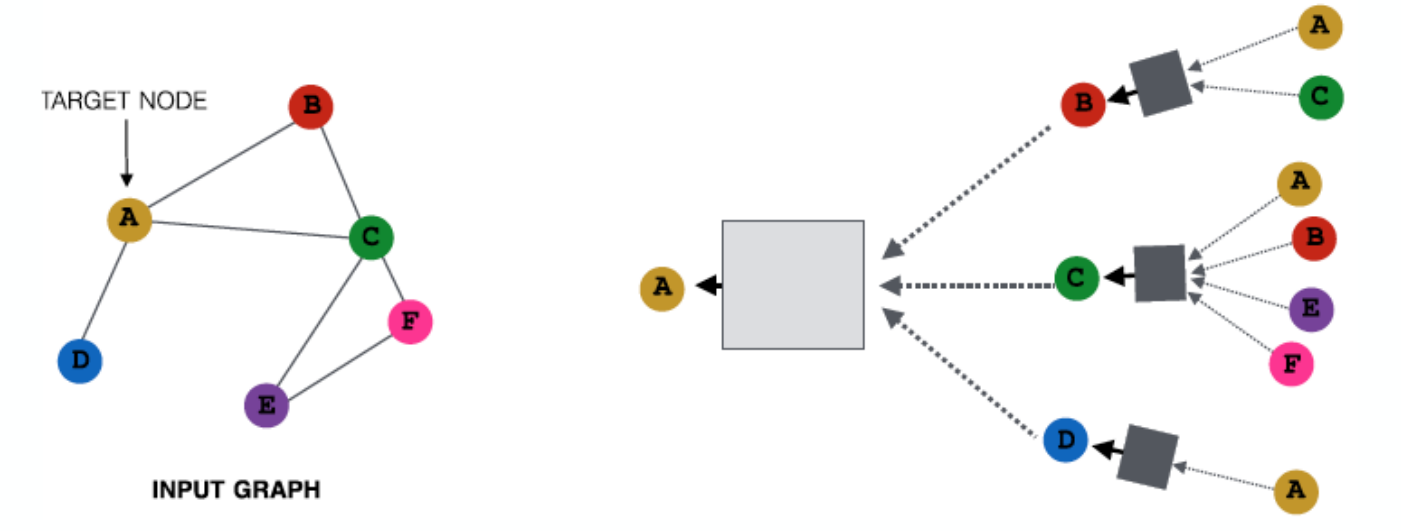
- Les deux sont étroitement liés.
- Exemple : supposons que la matrice d'adjacence est définie avec l'indexation (A,B,...,F)
  - La représentation du nœud jaune est invariante à la permutation de l'indexation du nœud
  - La représentations du nœud A est équivariante (permuter les indexes des nœuds permutera les représentations)

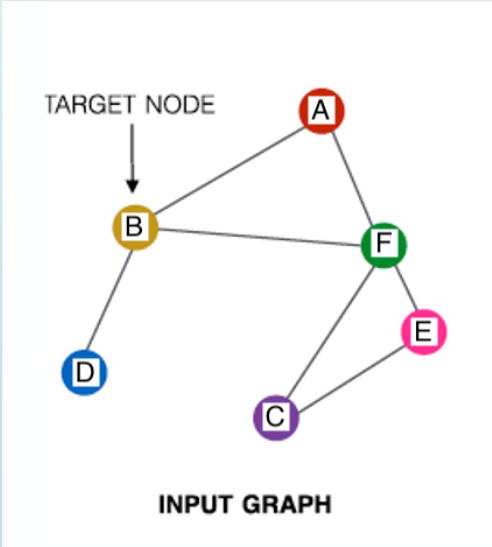




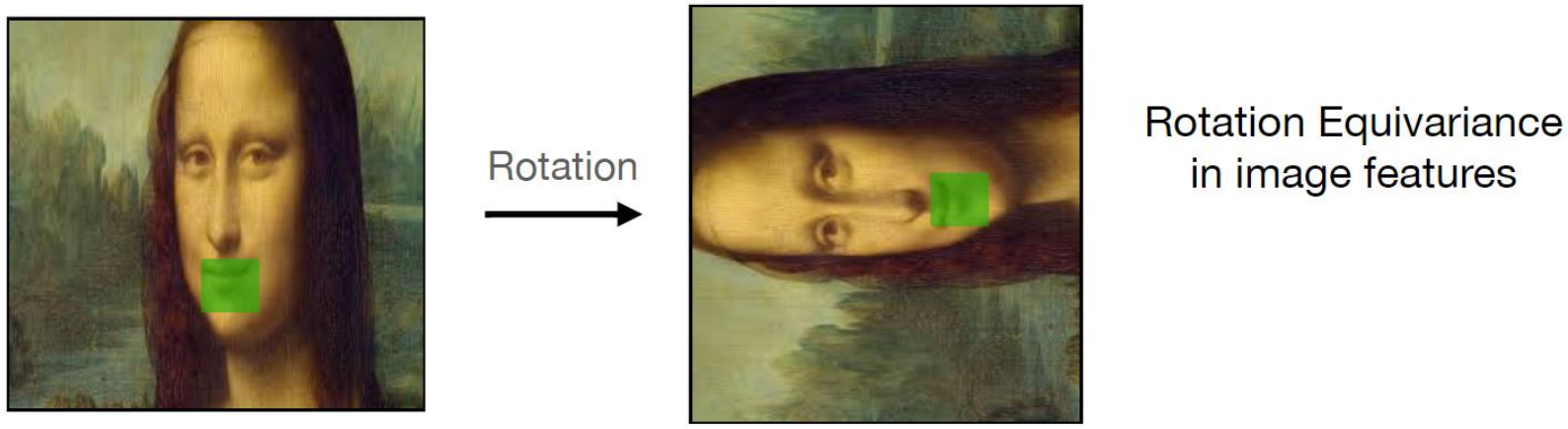
Quand voulons-nous l'invariance et quand voulons-nous l'équivariance ?

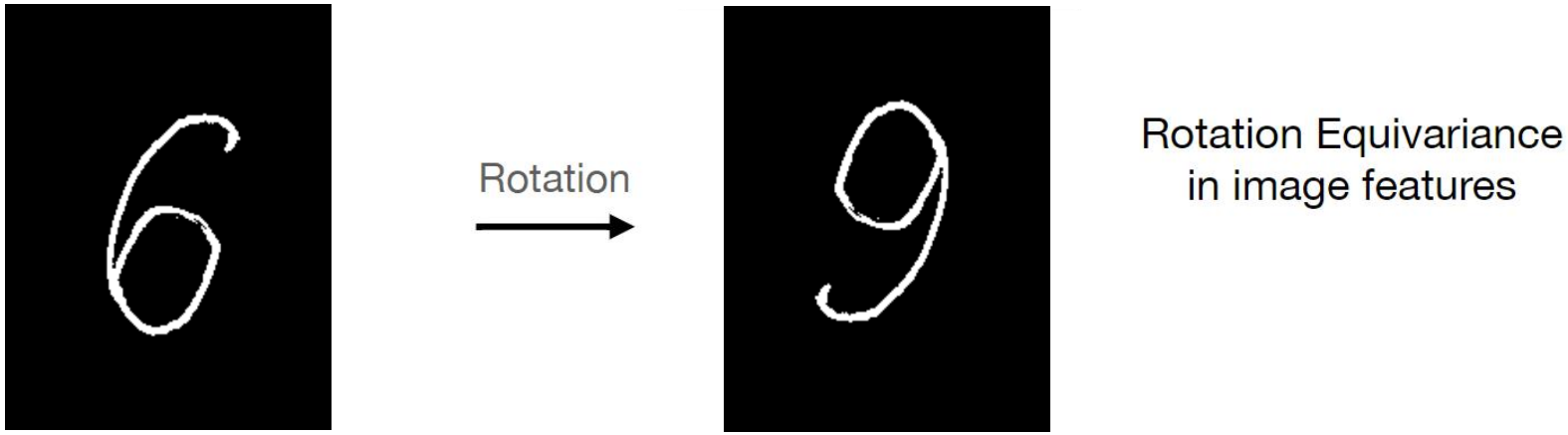
- Les deux sont étroitement liés.
- Habituellement:
  - Nous voulons des **représentations équivariantes.**
  - Nous voulons des **prédictions invariantes.**





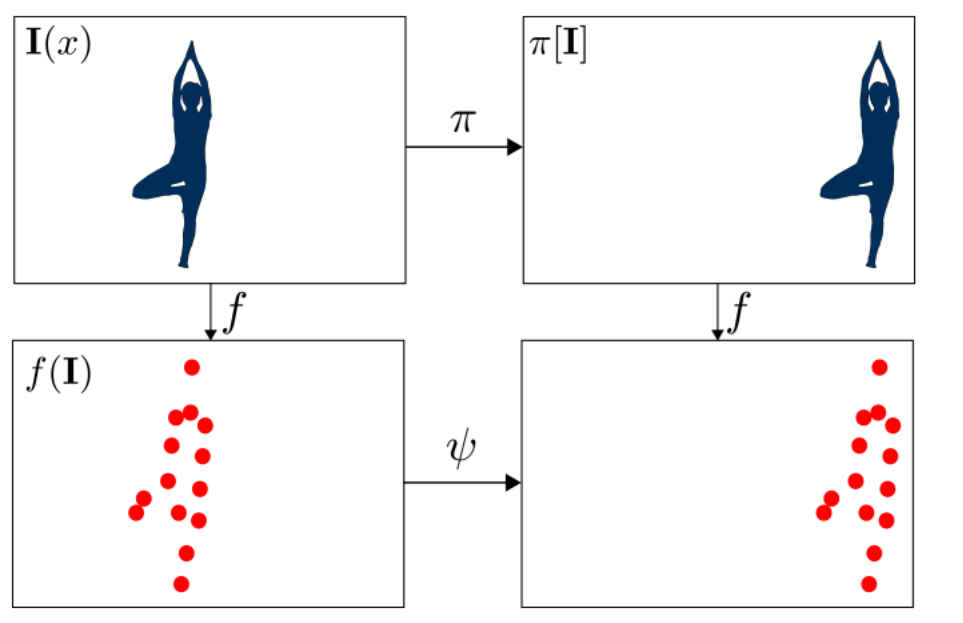
Exemple:


Rotation
Rotation Equivariance
in image features

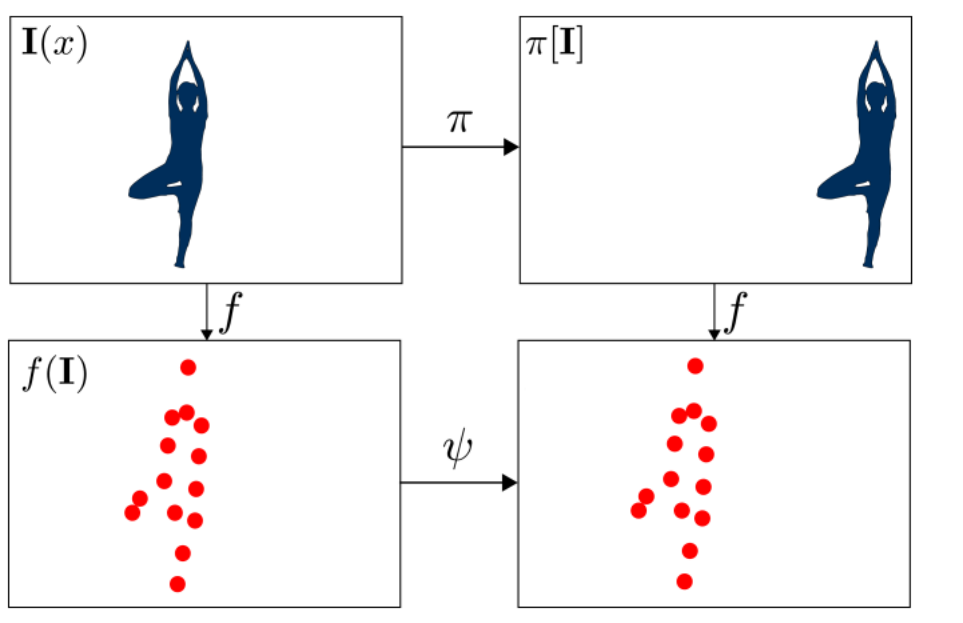

Rotation
Rotation Equivariance
in image features

Exercice:

A)

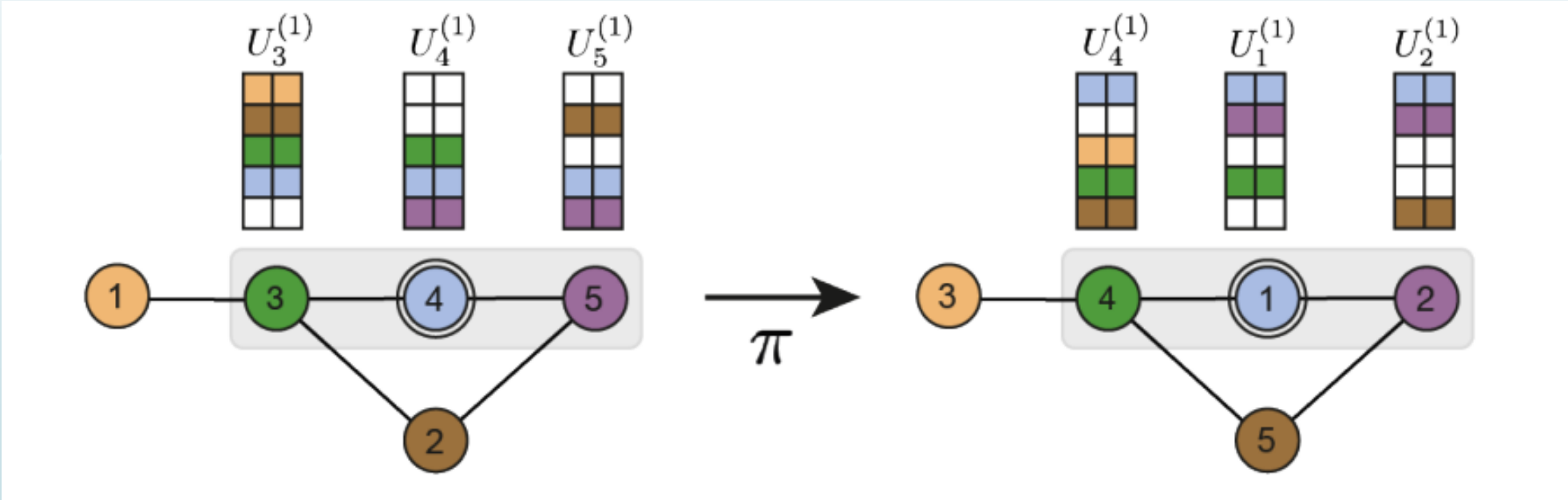

$\mathbf{I}(x)$
π
$\pi[\mathbf{I}]$
f
f
$f(\mathbf{I})$
ψ

B)


$\mathbf{I}(x)$
π
$\pi[\mathbf{I}]$
f
f
$f(\mathbf{I})$
ψ

Exercice

- Équivariant, invariant ou ni l'un ni l'autre?



Entraînement du modèle

trainable matrices
(i.e., what we learn)

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

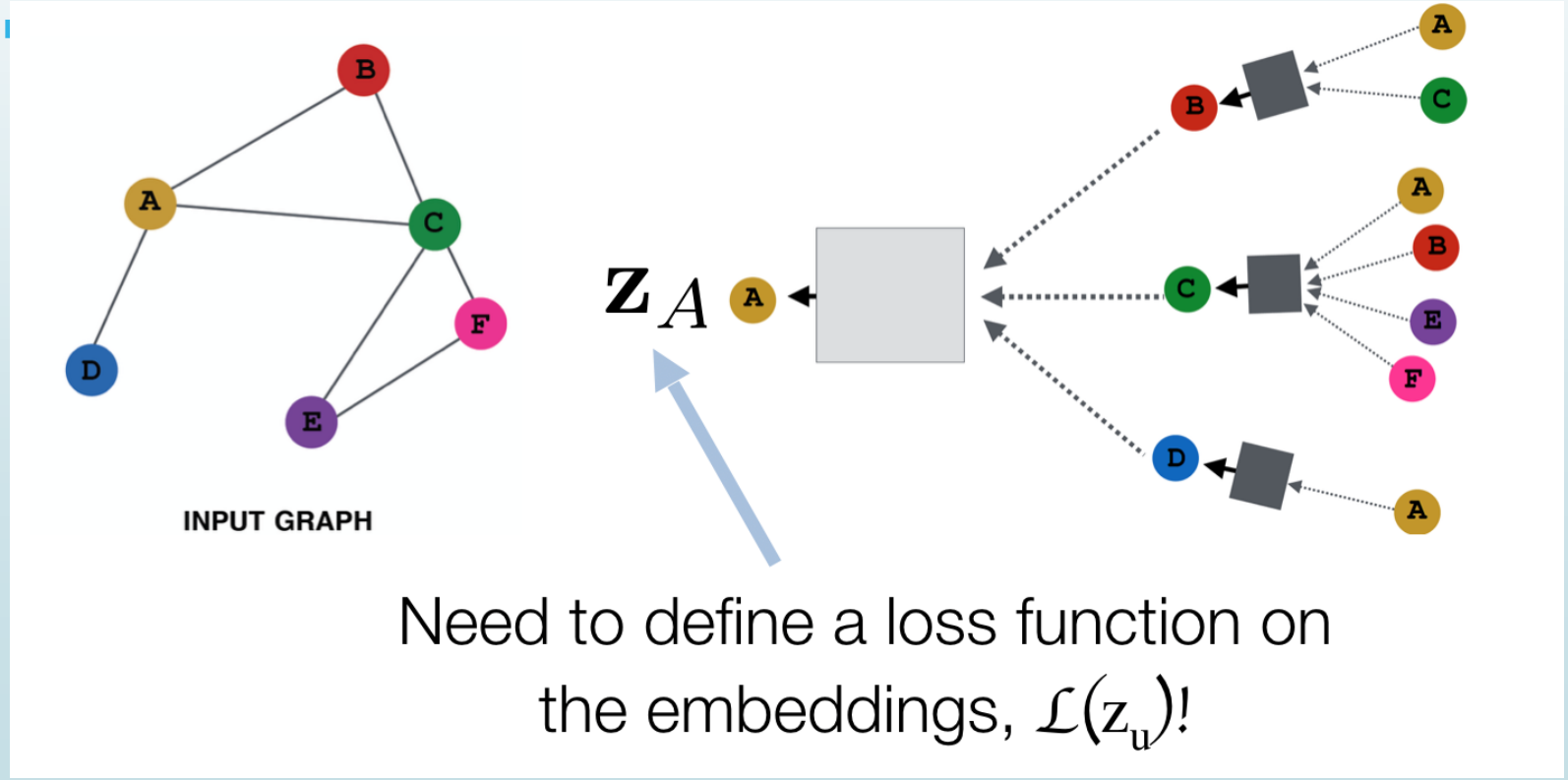
$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k \in \{1, ..., K\}$$

$$\mathbf{z}_v = \mathbf{h}_v^K$$

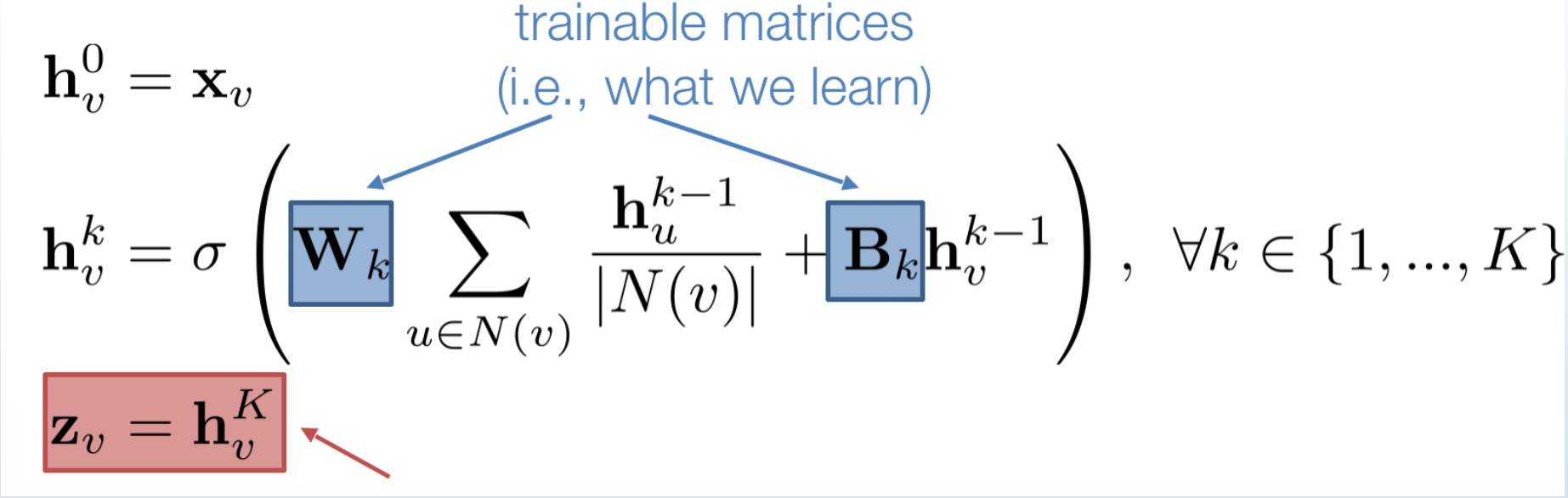
Après les couches K de l'agrégation de voisinage, nous obtenons des représentations de sortie pour chaque nœud.

Entraînement du modèle

- Comment entraînons-nous le modèle à générer des représentations de « haute qualité » ?



Entraînement du modèle



- Après les couches K de l'agrégation de voisinage, nous obtenons des plongements pour chaque nœud.
- Nous pouvons utiliser ces représentations dans n'importe quelle fonction de perte et exécuter une descente de gradient stochastique pour entraîner les paramètres d'agrégation.

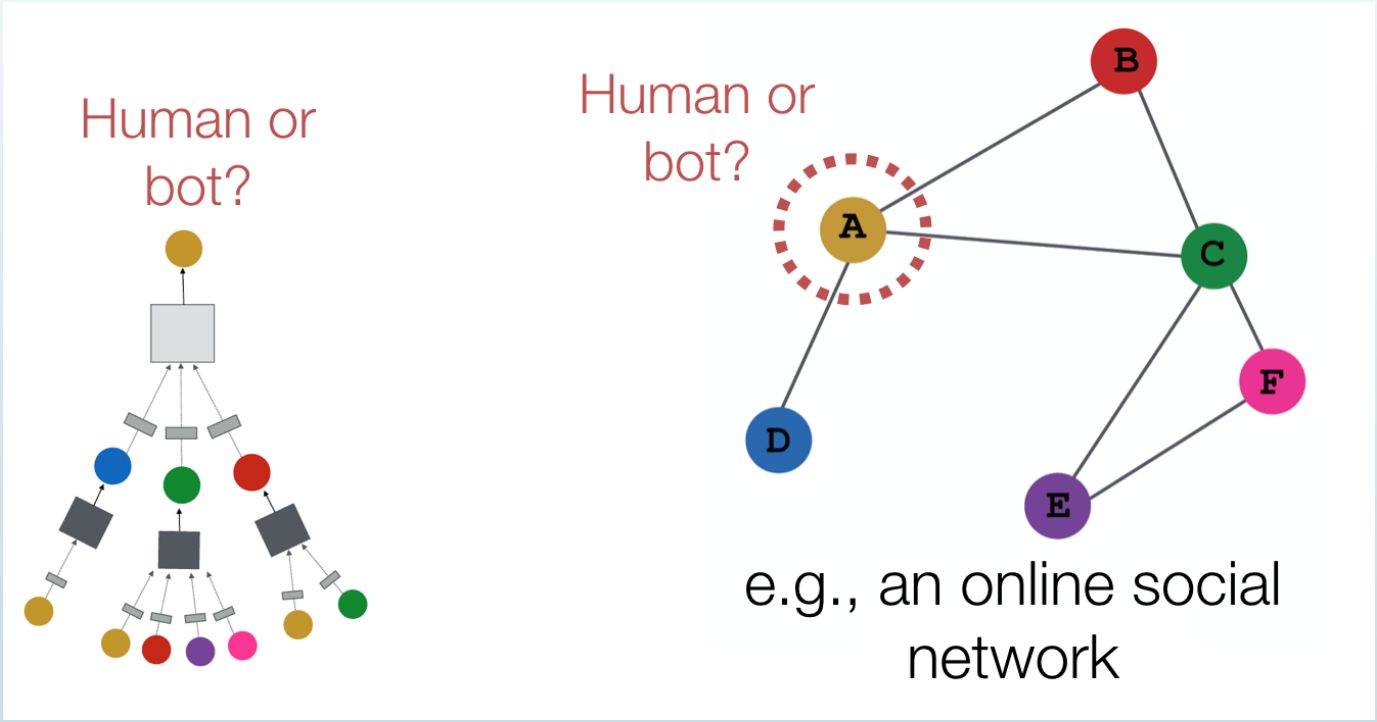
Entraînement du modèle

- Entraîner de manière **non supervisée** en utilisant uniquement la structure du graphe et les similarités.

- La fonction de perte non supervisée peut être n'importe quoi, par exemple, basée sur
  - Marches aléatoires (node2vec, DeepWalk)
  - Factorisation de graphes
    - c'est-à-dire entraîner le modèle de sorte que les nœuds « similaires » aient des intégrations similaires.

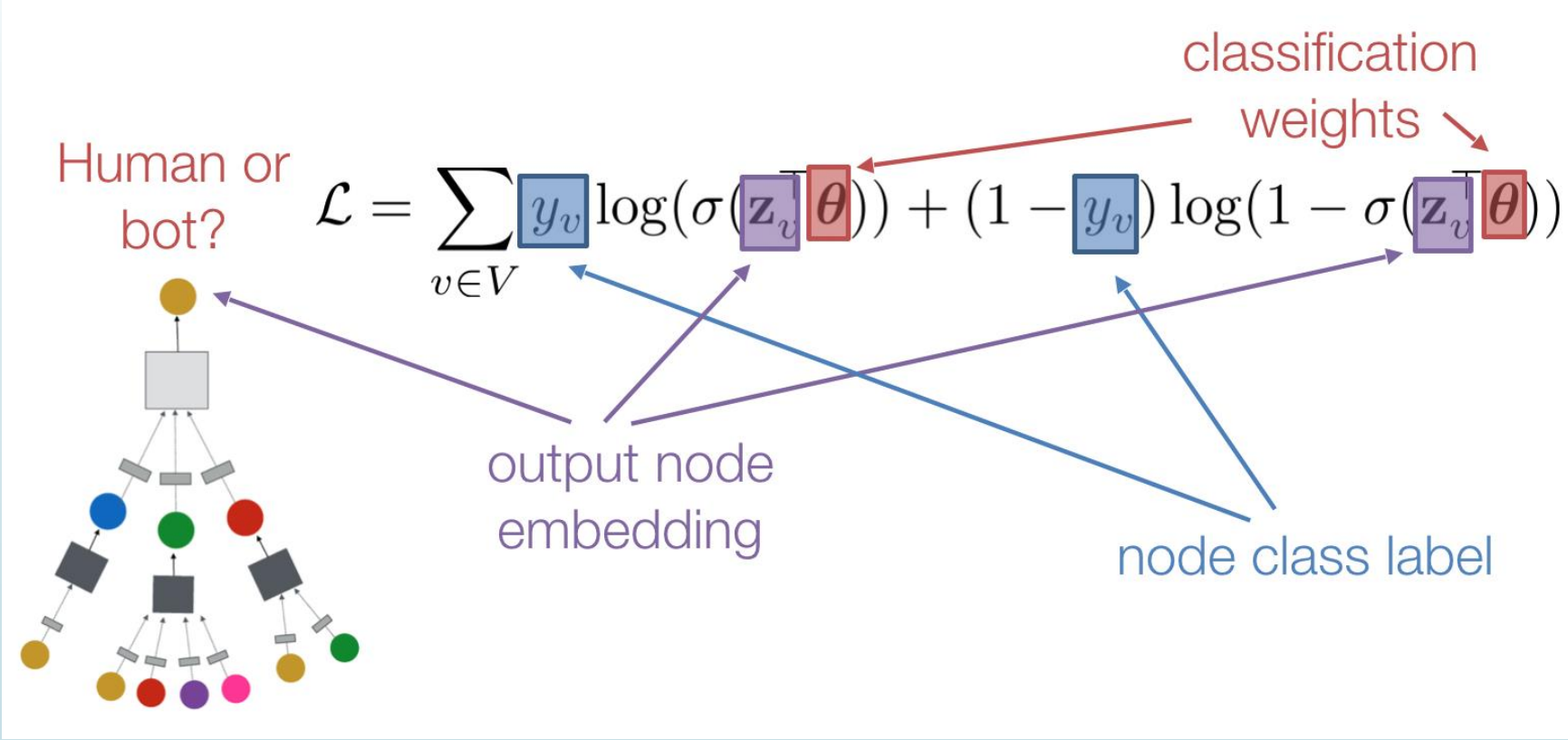
Entraînement du modèle

- Alternative : Entraîner directement le modèle pour une **tâche supervisée** (p. ex., classification des nœuds) :

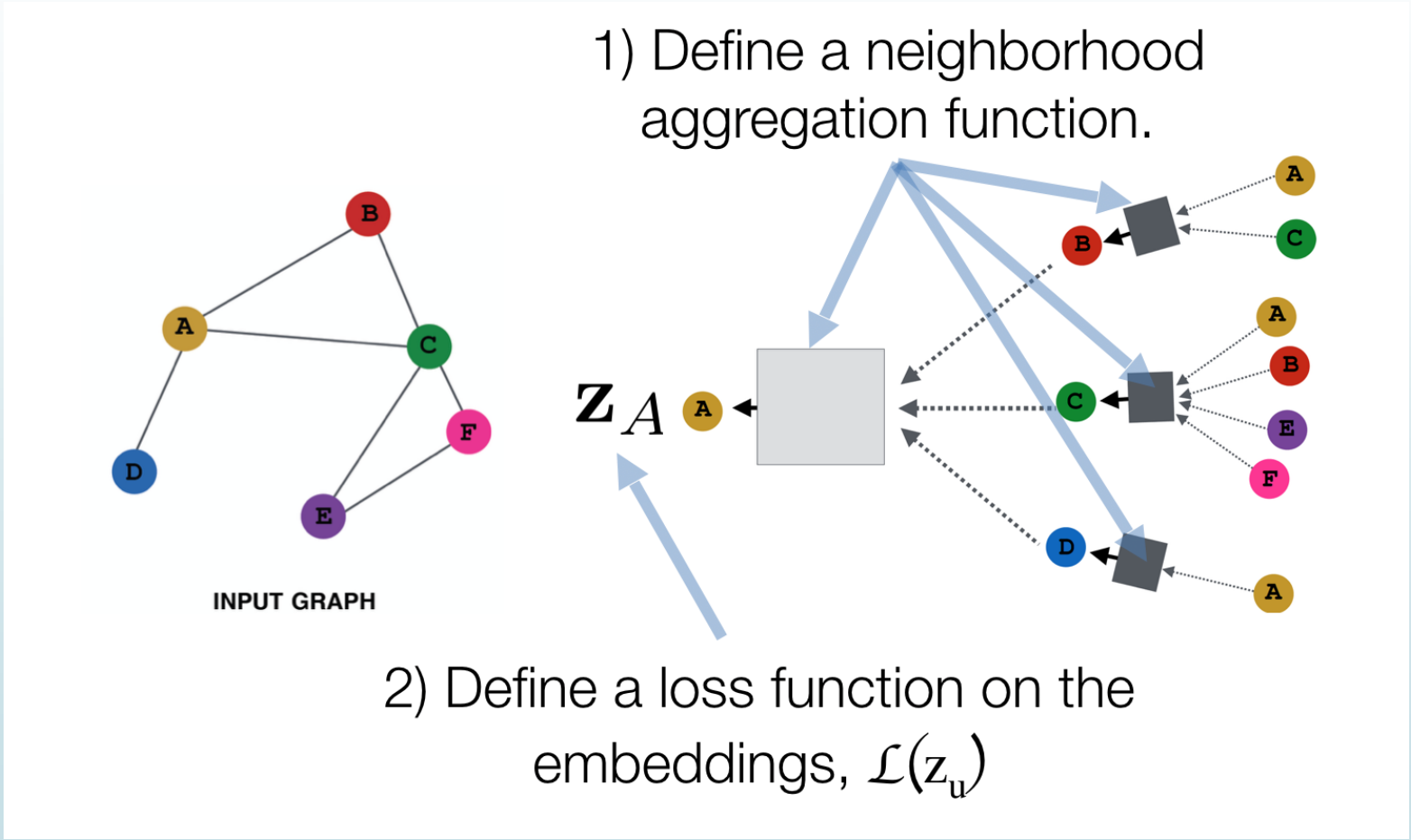


Entraînement du modèle

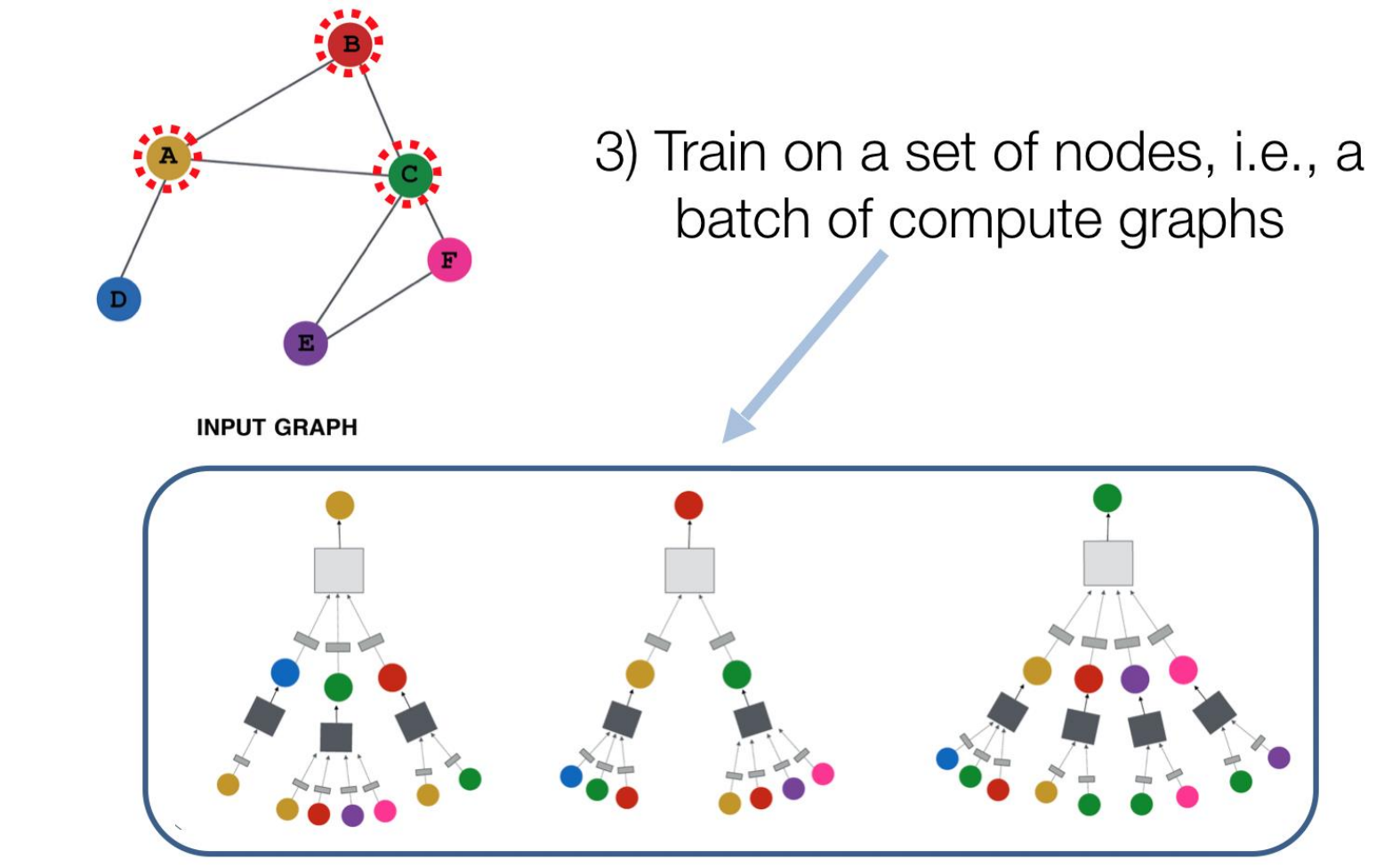
- Alternative : Entraîner directement le modèle pour une **tâche supervisée (**p. ex., classification des nœuds) :



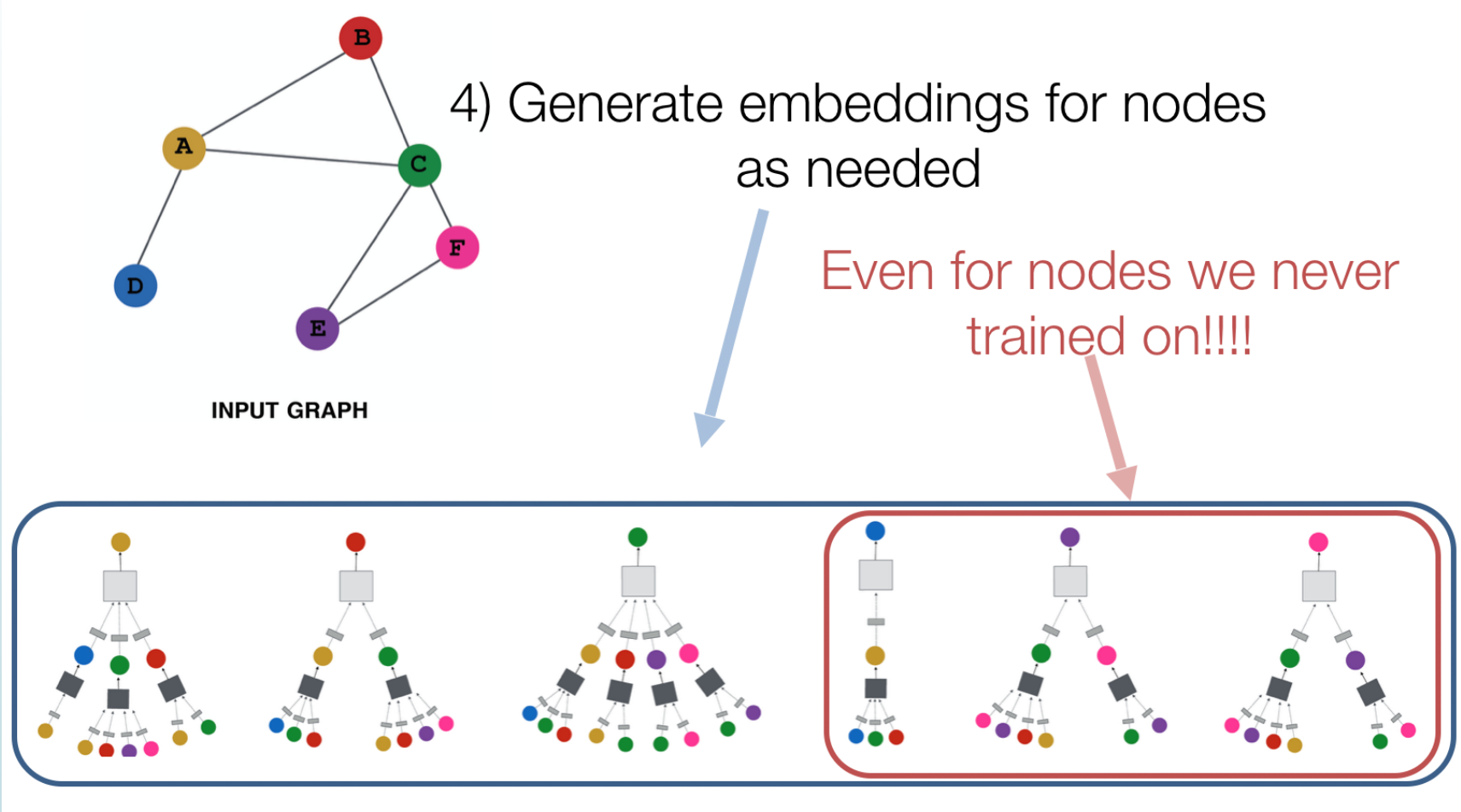
Vue d'ensemble du modèle



Vue d'ensemble du modèle

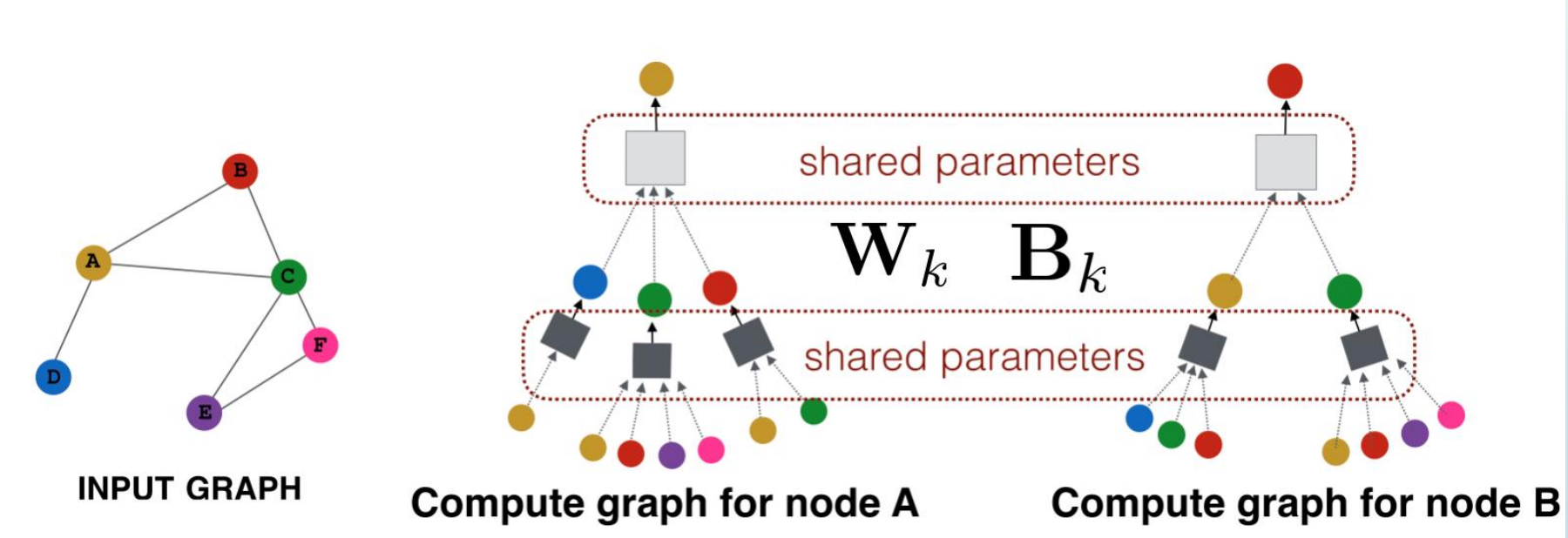


Vue d'ensemble

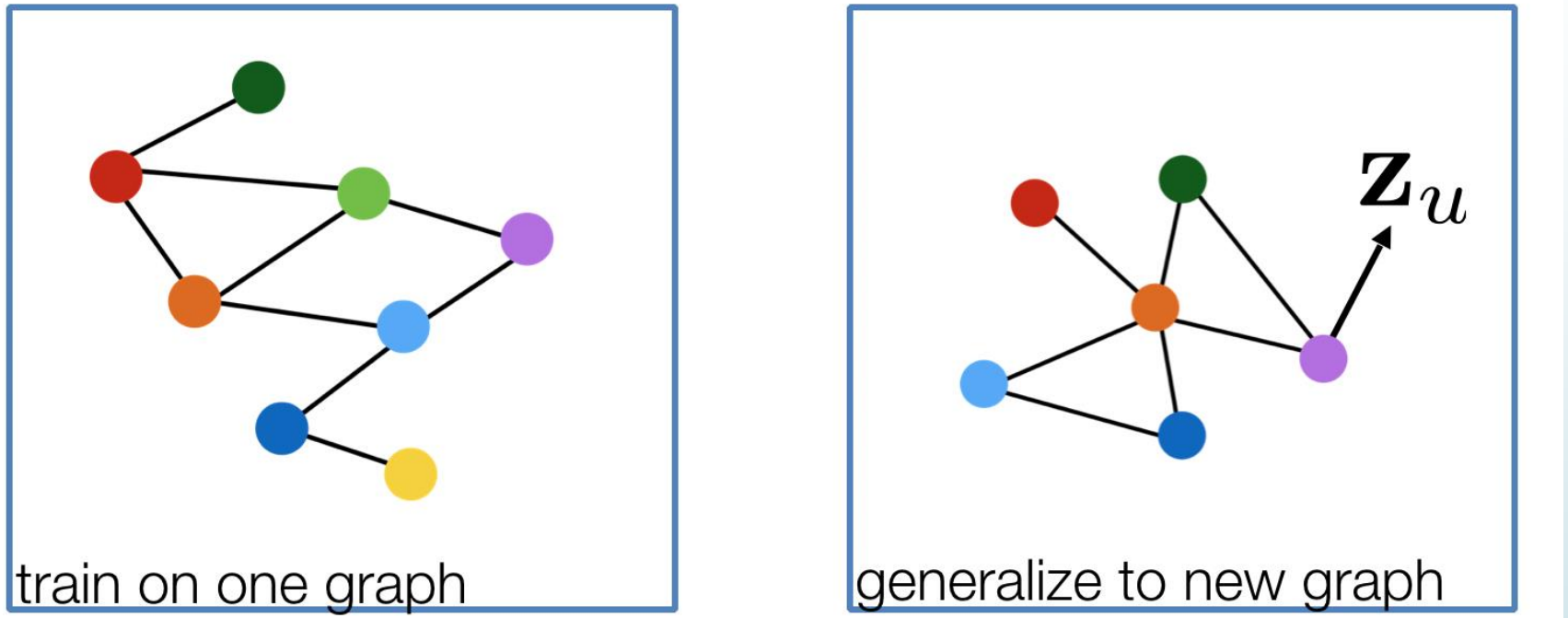


Capacité de généralisation

- Les mêmes paramètres d'agrégation sont partagés pour tous les nœuds.
- Le nombre de paramètres du modèle est sous-blinéaire en |V| Et nous pouvons généraliser aux nœuds invisibles/nouveaux!



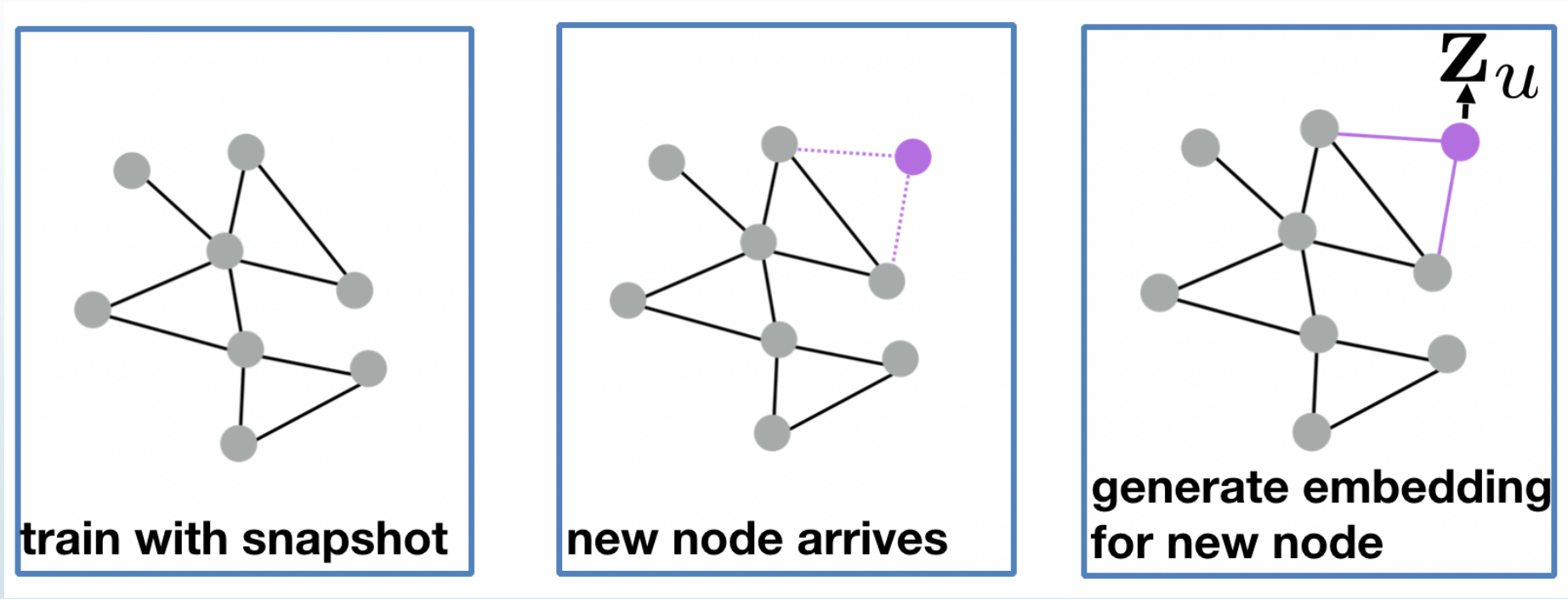
Capacité de généralisation



Inductive node embedding → generalize to entirely unseen graphs

- p. ex., s'entraîner sur le graphe d'interaction des protéines de l'organisme modèle A et générer des intégrations sur les données nouvellement recueillies sur l'organisme B

Capacité de généralisation



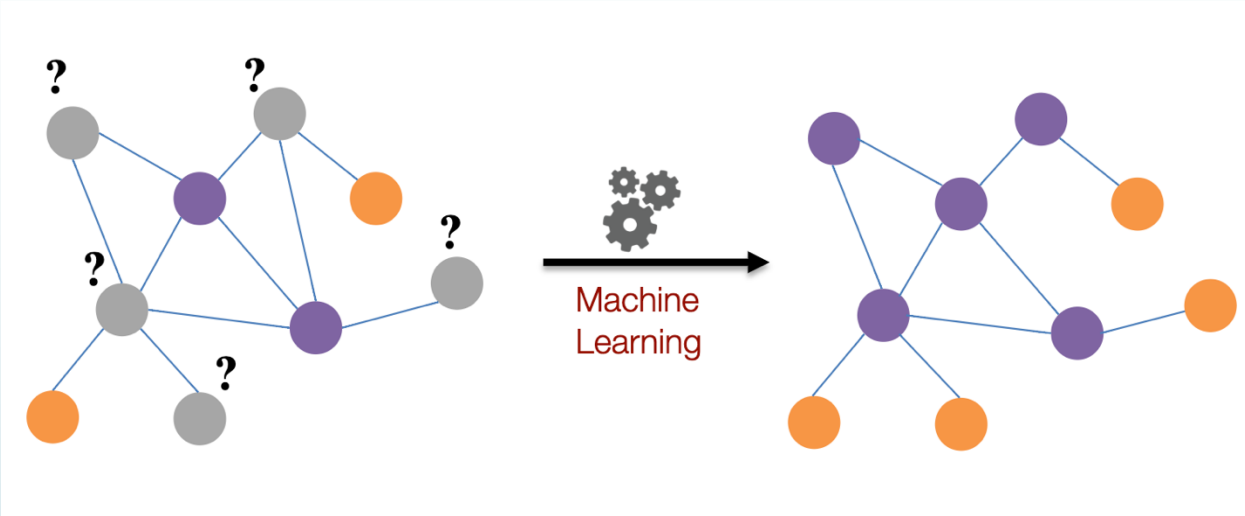
- De nombreux applications rencontrent constamment des nœuds inédits.
  - Par exemple, Reddit, YouTube, GoogleScholar, ...
- Besoin de générer de nouvelles représentations « à la volée »

Les réseaux de neurones pour graphes dans la pratique

- Prédiction de nœud
- Prédiction de lien
- Classification de graphes

Classification des nœuds

- y_u : étiquettes pour les nœuds
- z_u : représentations de nœuds
- Appris pour la tâche



$$\mathcal{L} = \sum_{u \in \mathcal{V}_{\text{train}}} -\log(\text{softmax}(\mathbf{z}_u, \mathbf{y}_u)).$$

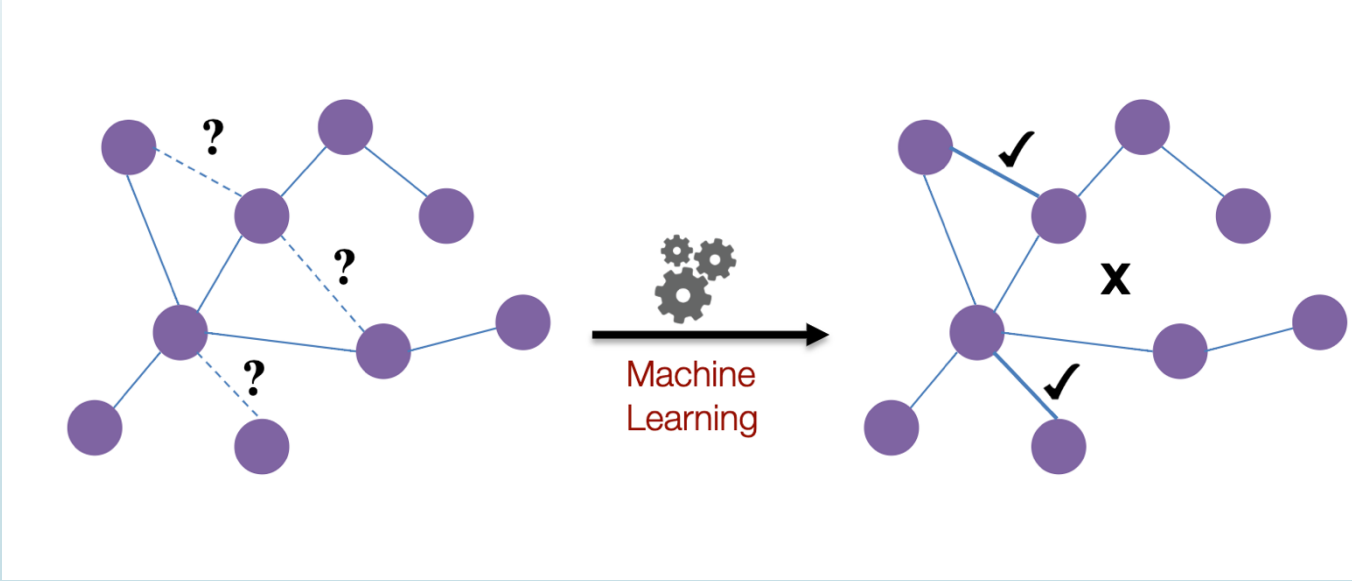
Prédiction de lien

- y_u : étiquettes pour les nœuds
- z_u : représetations de nœuds
- Appris pour la tâche

Maximiser

$$p(A_{ij}) = \sigma(\mathbf{z_i^T z_j})$$

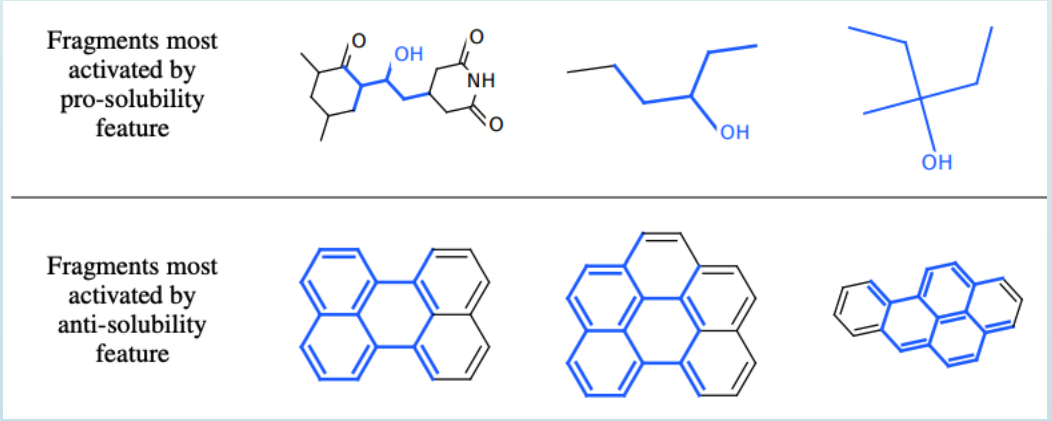
Sur l'ensemble d'entrainement



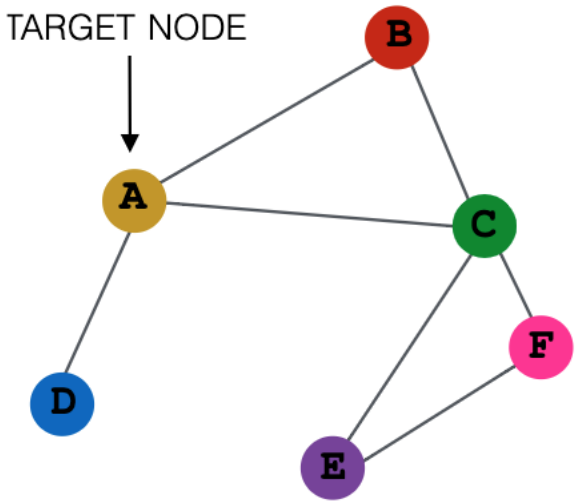
Prédiction à partir d'un graphe

- z_G: représentation pour le graphe (nous verrons comment l'obtenir)

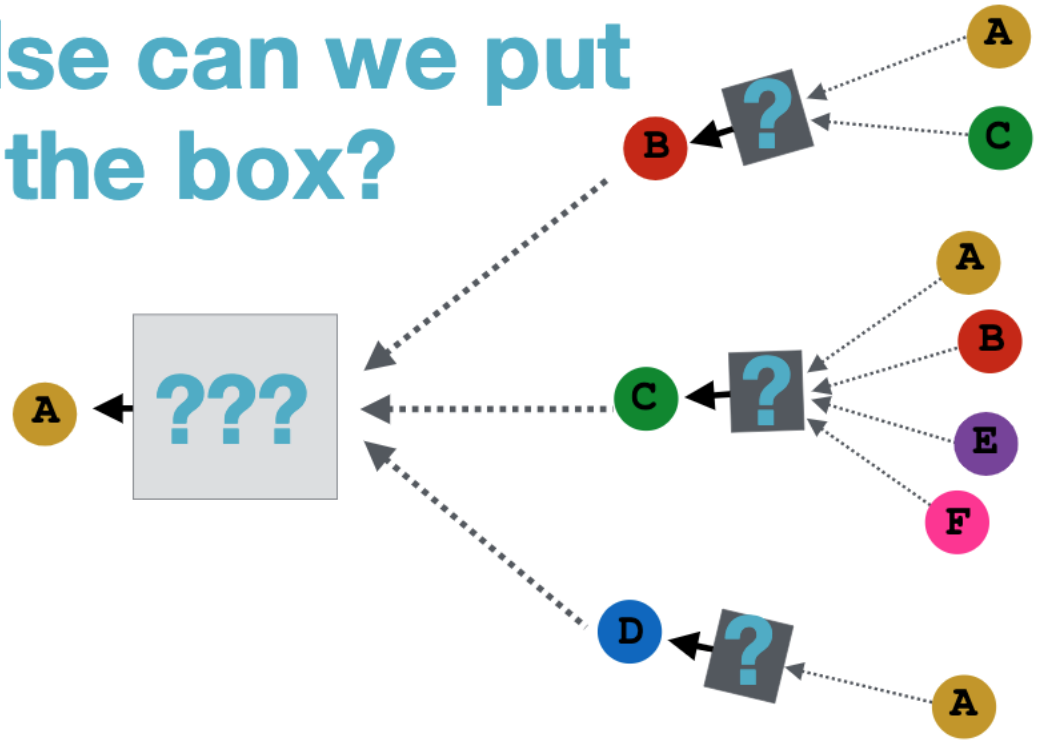
$$\mathcal{L} = \sum_{\mathcal{G}_i \in \mathcal{T}} \|\mathbf{MLP}(\mathbf{z}_{\mathcal{G}_i}) - y_{\mathcal{G}_i}\|_2^2,$$



What else can we put in the box?



INPUT GRAPH

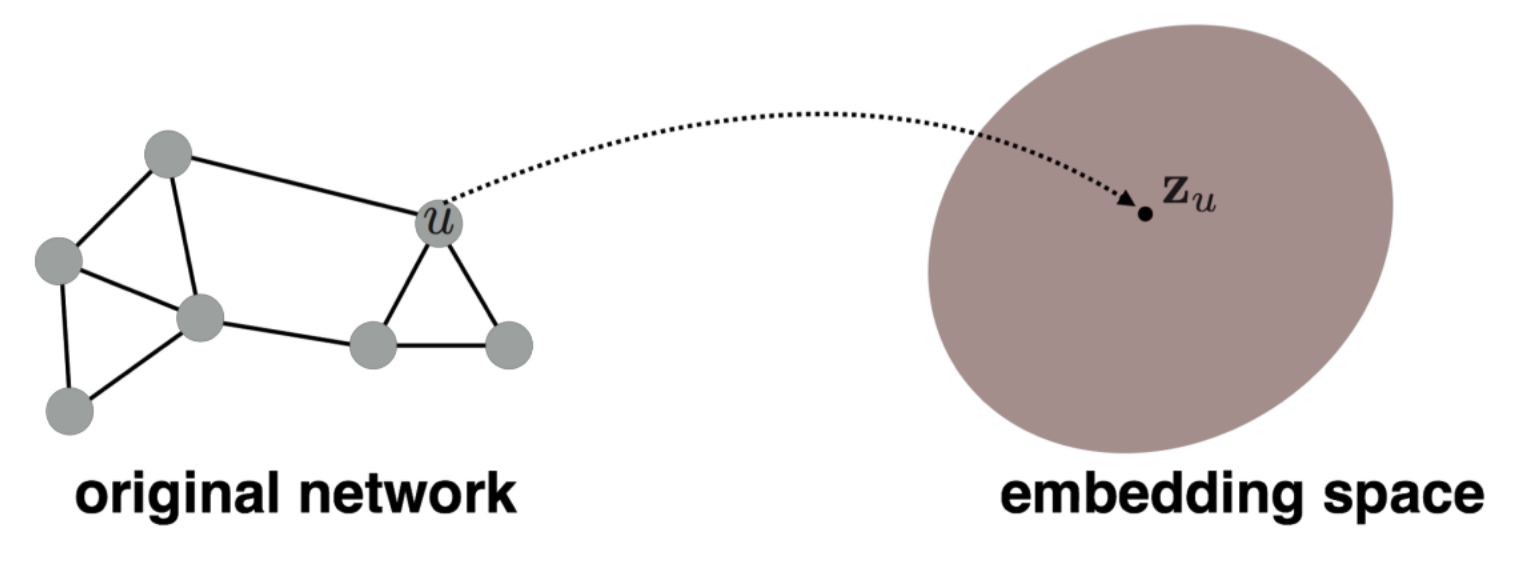


Représentation de (sous)graphe

- Refs: **Convolutional Networks on Graphs for Learning Molecular Fingerprints** https://arxiv.org/abs/1509.09292
- Li et al. 2016. **Gated Graph Sequence Neural Networks. ICLR**.
- Ying et al, 2018. **Hierarchical Graph Representation Learning with Differentiable Pooling**. NeurIPS.

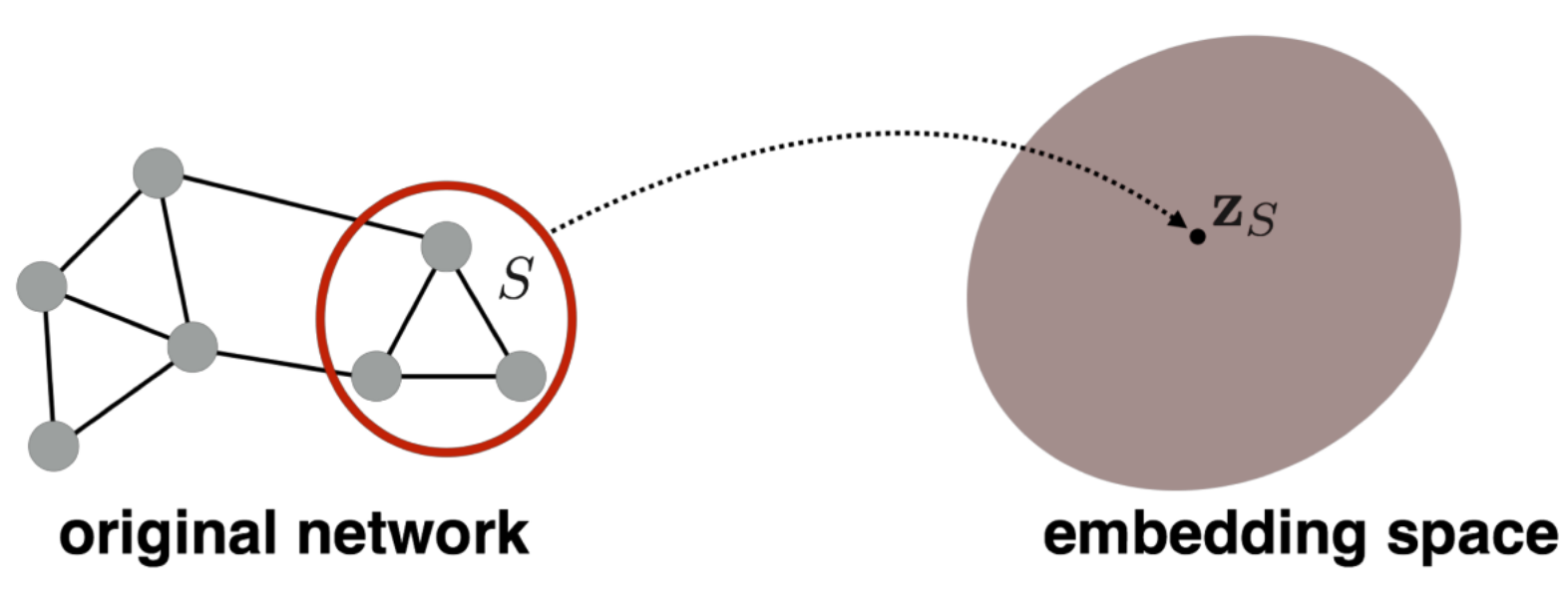
(Sub) Graph embeddings

- So far we have focused on node-level embeddings…



(Sub) Représentation de graphe

- Qu'en est-il de la représentation d'un sous-graphe



Approche 1

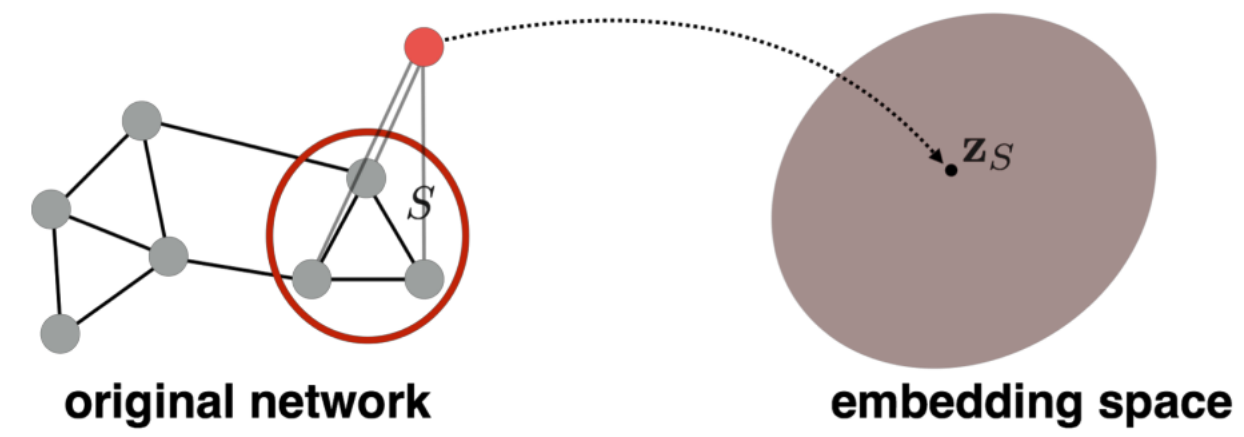
- Idée simple : additionner les representations

$$\mathbf{z}_S = \sum_{v \in S} \mathbf{z}_v$$

- Utilisé par Duveneaud et al. 2016 (pour la classification des molécules)

Approche 2

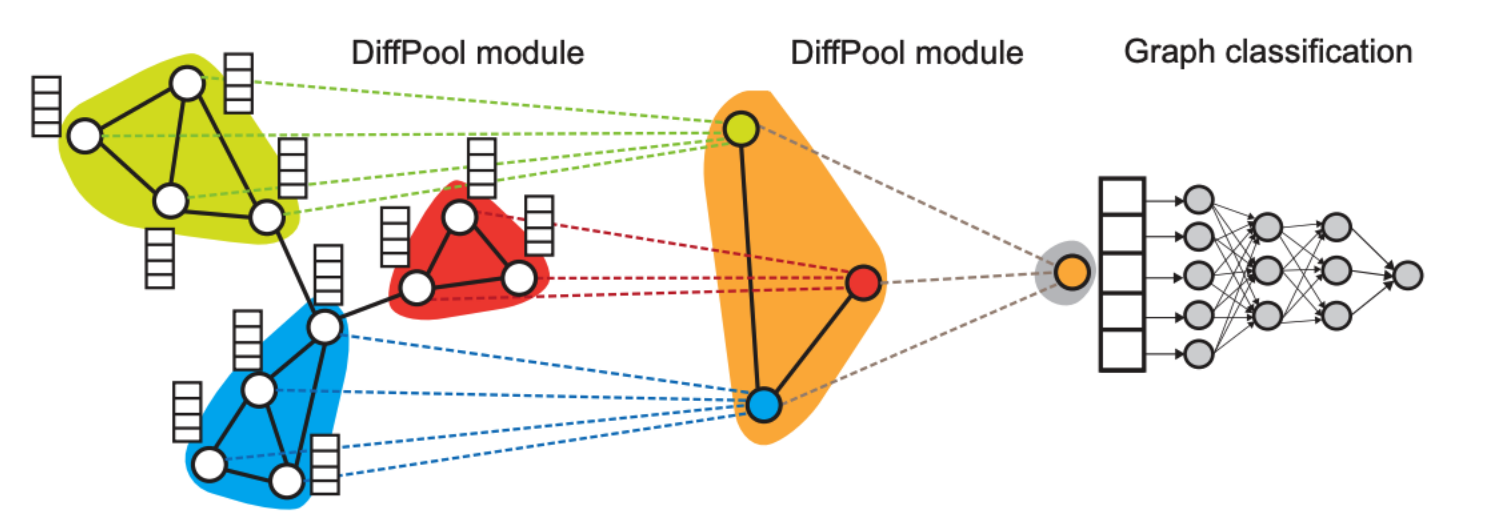
- Introduisez un **nœud virtuel** qui représente le sous-graphe et obtenez la représentation de ce nouveau nœud.



- Utilisé par Li et al. 2016

Approach 3

- **regrouper hiérarchiquement** les nœuds.

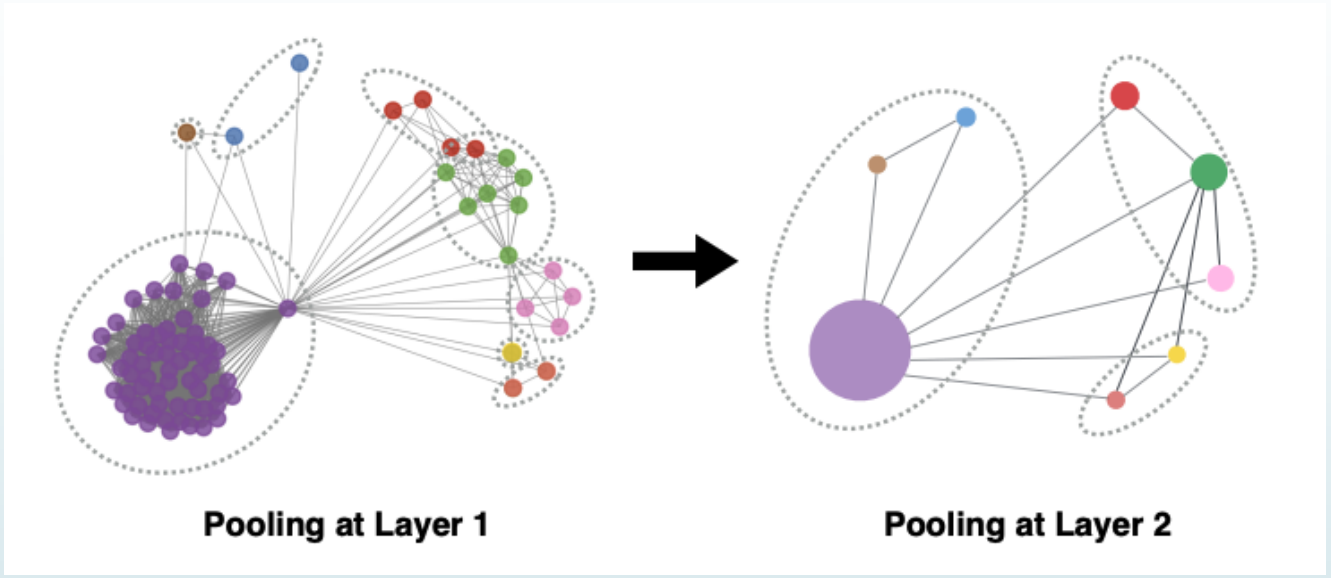


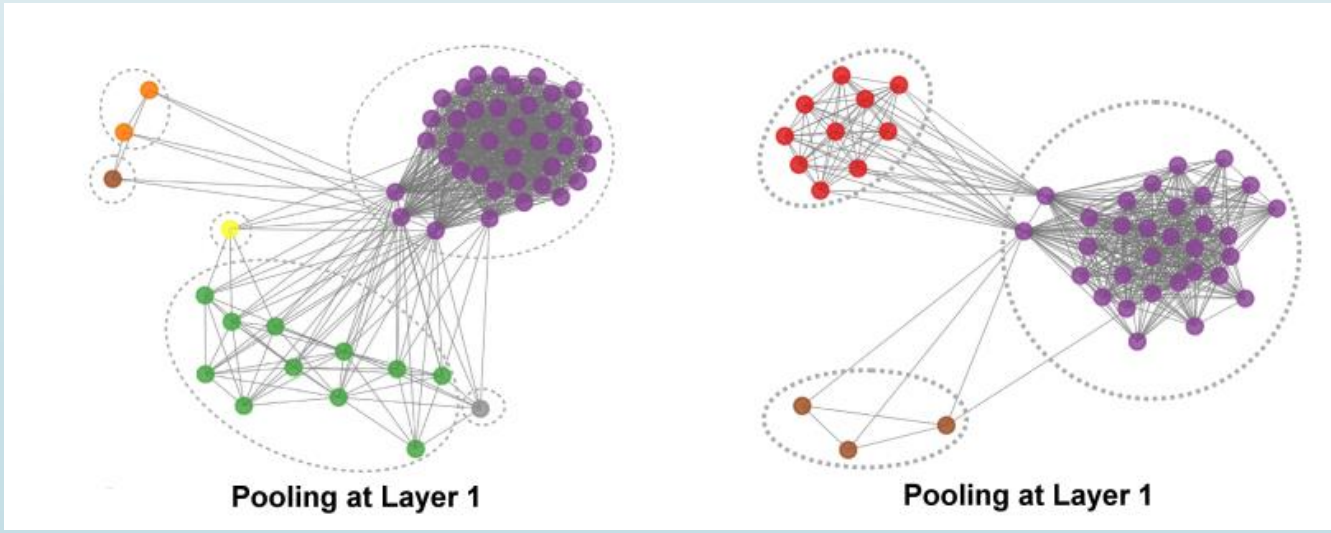
- Proposé par Ying et al. 2018

Approche 3

- Idée : Apprendre à regrouper hiérarchiquement les nœuds.
- Vue d'ensemble de base :
  - 1. GNN sur le graphe et obtenez des représentations de nœuds.
  - 2. Regroupez (clustering) les représentations de nœuds pour créer un graphe plus « grossier ».
  - 3. GNN sur le graphique « grossier ».
  - 4. Répétez.
- Différentes approches pour le regroupement (clustering) :
  - Clustering "souple" via des poids softmax appris (Ying et al., 2018)
  - Clustering "rigide" (Cangea et al., 2018 and Gao et al., 2018)

Approach 3


Pooling at Layer 1
Pooling at Layer 2


Pooling at Layer 1
Pooling at Layer 1

Conclusion

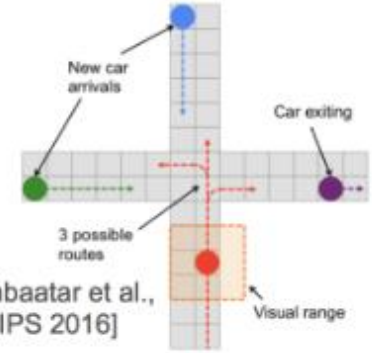
— **Deep learning on graphs works and is very effective!**

— Exciting area: lots of new applications and extensions (hard to keep up)

Relational reasoning

[Santoro et al., NIPS 2017]

Multi-Agent RL



[Sukhbaatar et al., NIPS 2016]

GCN for recommendation on 16 billion edge graph!

[Leskovec lab, Stanford]

Résumé: GNNs

- Récapitulatif : générez des représentations de nœuds en agrégeant les informations de voisinage.
  - Permet **le partage** des paramètres dans l'encodeur.
  - Permet un apprentissage inductif (généralisation aux noeuds non-vus).

Résumé Global sur les Graphes

- Représentations peu-profondes:
  - Un vecteur par nœud (peu prendre trop de place pour les gros graphes)
  - 'Des représentations similaires pour des nœuds similaires'
  - Trois grandes notions de similitudes:
    - Adjacences
    - Voisinage (k-hop,…)
    - Mache aléatoire.
- Représentation profonde:
  - Apprend un réseau de neurone pour extraire une représentation par nœud en fonction des voisins
  - Utilise les mêmes notions de similitudes que les représentations peu profondes.
  - Avantages (par rapport aux représentations peu profondes):
    - **Peu généraliser à des nœuds et des graphes non-vus pendant l'entraînement**
    - **Utilise les caractéristiaues des nœuds.**

Resources

- http://snap.stanford.edu/proj/embeddings-www/
- https://jian-tang.com/files/AAAI19/aaai-grltutorial-part2-gnns.pdf