Machine Learning on Graphs
Graph Neural Networks

Lesson created by Stéphane Nicolas / Pierre Héroux /
Sébastien Adam / PhD students

14 novembre 2024

Outline of the lecture

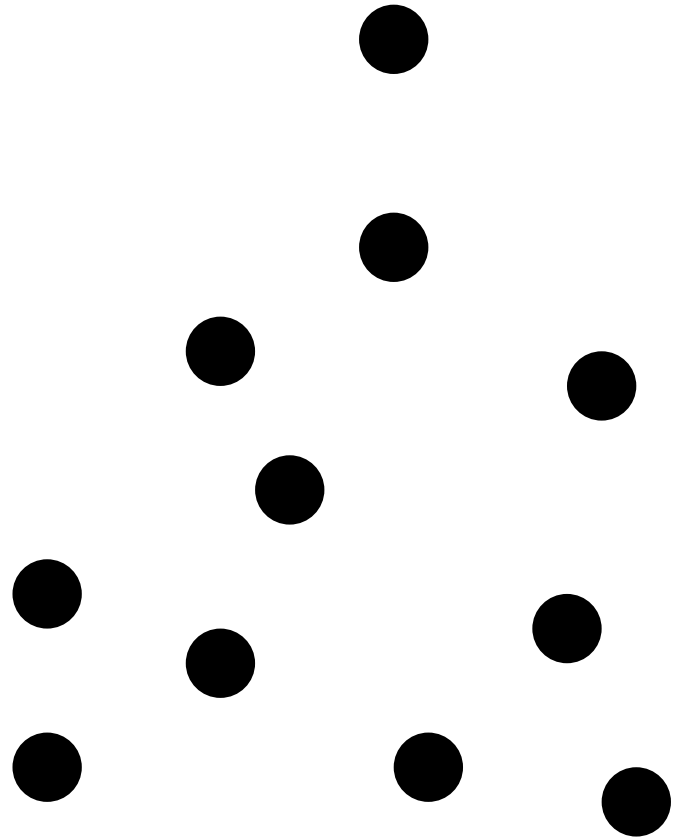
Graph definition and notations

$$G = (\mathcal{V}, \mathcal{E}, \mu, \xi)$$

Graph definition and notations

$G = (\mathcal{V}, \mathcal{E}, \mu, \xi)$

$\mathcal{V}$ is a set of nodes

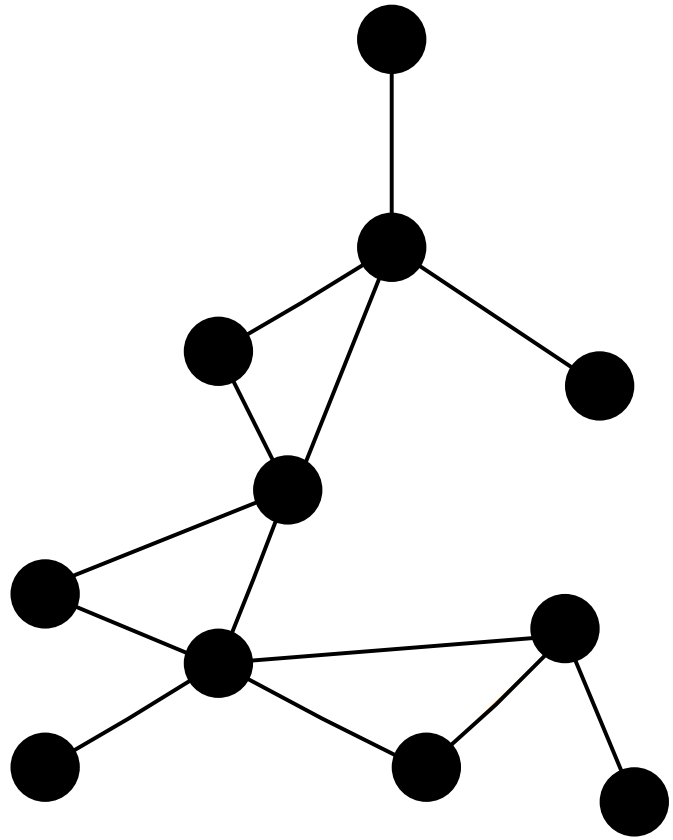


Graph definition and notations

$G = (\mathcal{V}, \mathcal{E}, \mu, \xi)$

$\mathcal{V}$ is a set of nodes

$\mathcal{E}$ is a set of edges linking pairs of nodes



Graph definition and notations

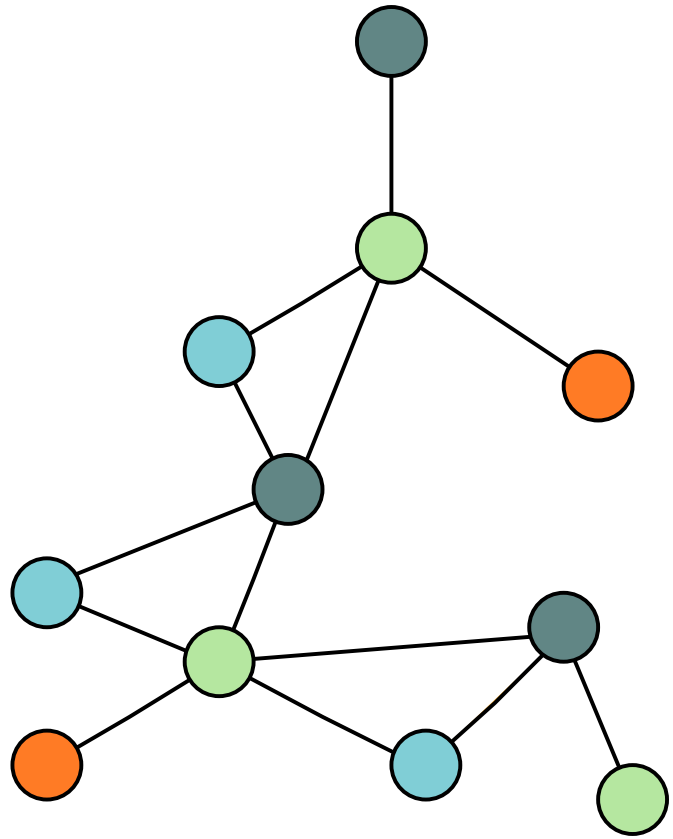
$G = (\mathcal{V}, \mathcal{E}, \mu, \xi)$

$\mathcal{V}$ is a set of nodes

$\mathcal{E}$ is a set of edges linking pairs of nodes

Each node can be attributed by

μ : the signal on the graph



Graph definition and notations

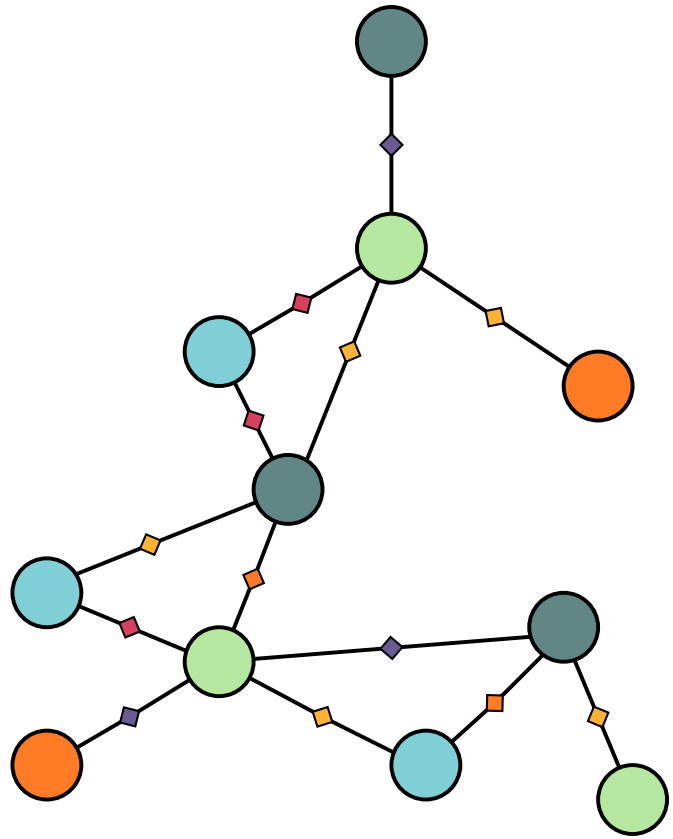
$G = (\mathcal{V}, \mathcal{E}, \mu, \xi)$

$\mathcal{V}$ is a set of nodes

$\mathcal{E}$ is a set of edges linking pairs of nodes

Each node can be attributed by μ : the signal on the graph

Each edge can also be attributed by ξ



Graph definition and notations

$G = (\mathcal{V}, \mathcal{E}, \mu, \xi)$

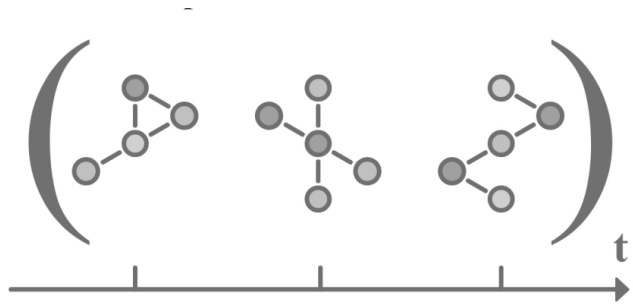
$\mathcal{V}$ is a set of nodes

$\mathcal{E}$ is a set of edges linking pairs of nodes

Each node can be attributed by μ : the signal on the graph

Each edge can also be attributed by ξ

NB : graph may evolve through time : $G_t = (\mathcal{V}, \mathcal{E}, \mu, \xi)_t$



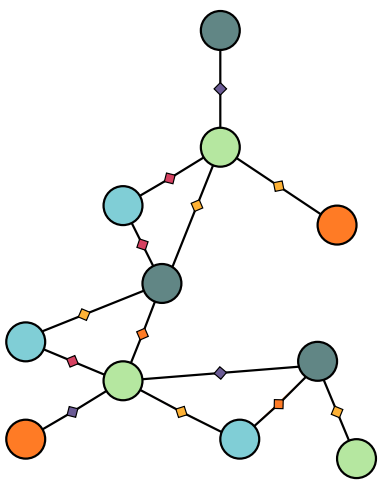
Different types of task (with image analogies)

What kind of decision can we take on graphs ?

Different types of task (with image analogies)

What kind of decision can we take on graphs ?

- Graph classification/regression (in : a graph / out : a label)



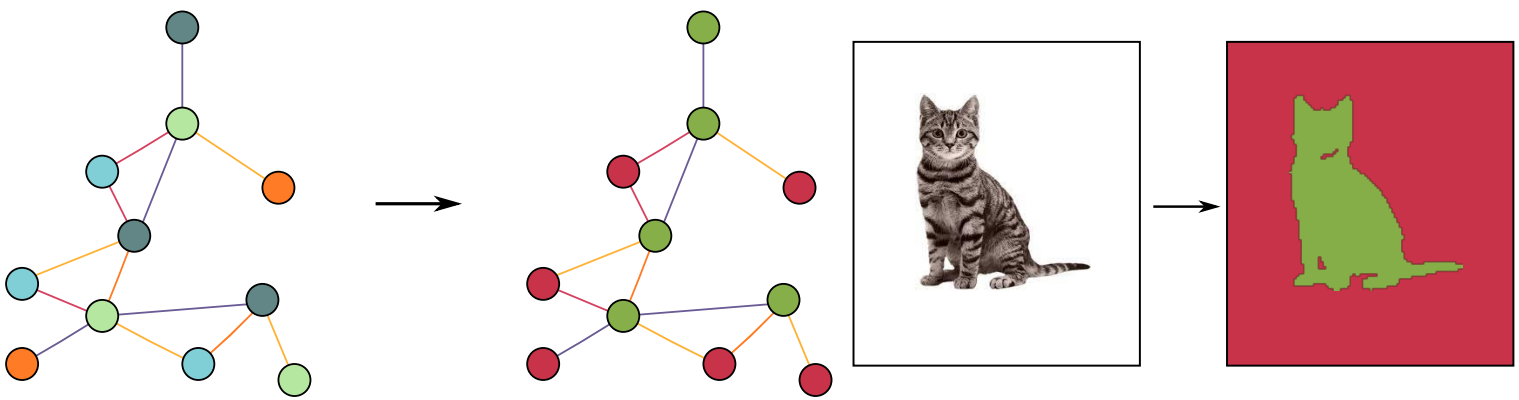
$\longrightarrow \{active, unactive\}$

$\longrightarrow \{cat, dog\}$

Different types of task (with image analogies)

What kind of decision can we take on graphs ?

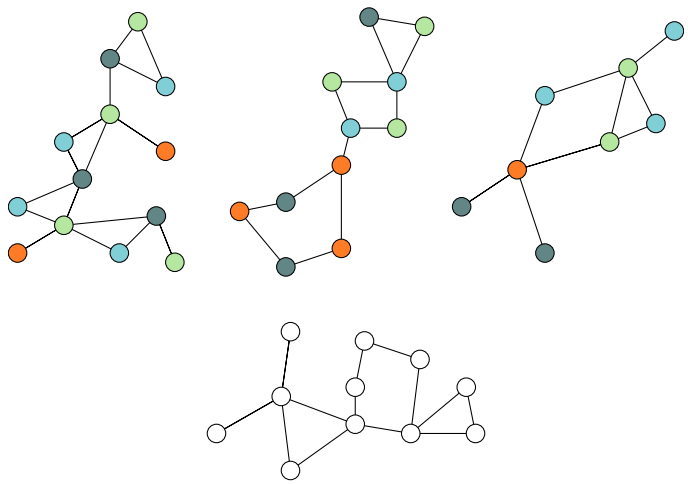
- Graph classification/regression (in : a graph / out : a label)
- Nodes classification/regression (in : a graph / out : a labelled graph) : 2 cases

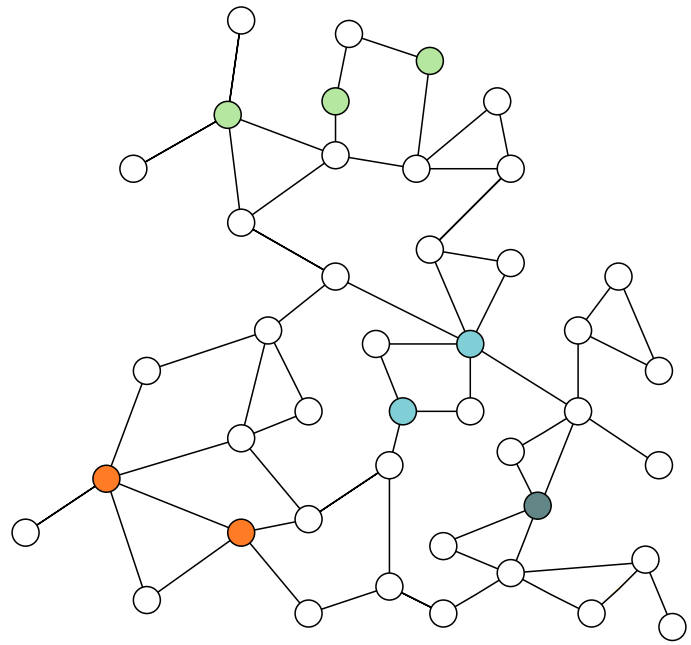


Different types of task (with image analogies)

2 particular cases in the case of graphs according to learning data

- Inductive tasks : the decision is taken on a "fully" unknown graph
- Transductive tasks : the decision is taken on a graph with some labeled training nodes (semi-sup)

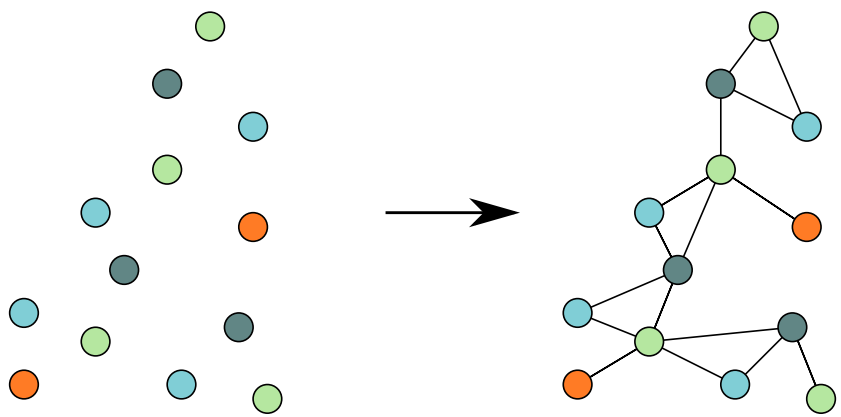




Different types of task (with image analogies)

What kind of decision can we take on graphs ?

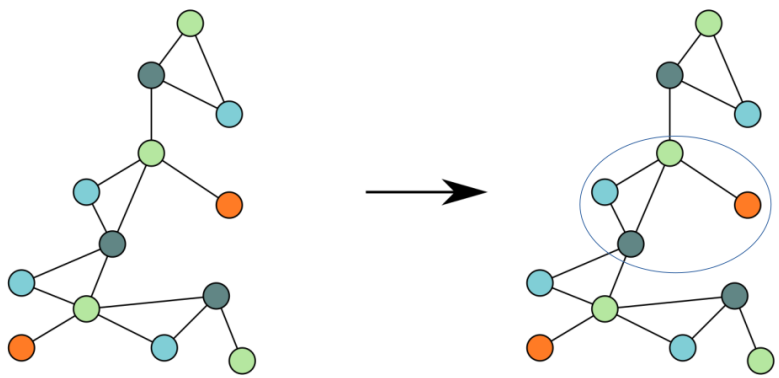
- Graph classification/regression (in : a graph / out : a label)
- Nodes classification/regression (in : a graph / out : a labelled graph) : 2 cases
- Link prediction (in : pair of nodes / out : existence of an edge)



Different types of task (with image analogies)

What kind of decision can we take on graphs ?

- Graph classification/regression (in : a graph / out : a label)
- Nodes classification/regression (in : a graph / out : a labelled graph) : 2 cases
- Link prediction (in : pair of nodes / out : existence of an edge)
- Community detection (in : a graph / out : a subgraph)



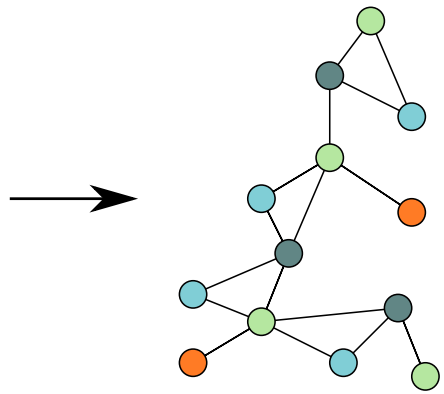
Different types of task (with image analogies)

What kind of decision can we take on graphs ?

- Graph classification/regression (in : a graph / out : a label)
- Nodes classification/regression (in : a graph / out : a labelled graph) : 2 cases
- Link prediction (in : pair of nodes / out : existence of an edge)
- Community detection (in : a graph / out : a subgraph)
- Graph generation (in : text-image-nothing / out : a graph)

A stop sign is flying in blue skies.

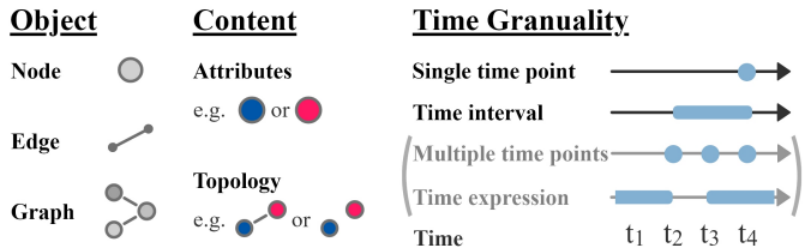
John is in the kitchen...
...The cat is near John



Different types of task (with image analogies)

What kind of decision can we take on graphs ?

- Graph classification/regression (in : a graph / out : a label)
- Nodes classification/regression (in : a graph / out : a labelled graph) : 2 cases
- Link prediction (in : pair of nodes / out : existence of an edge)
- Community detection (in : a graph / out : a subgraph)
- Graph generation (in : text-image-nothing / out : a graph)
- And many other tasks for dynamic graphs

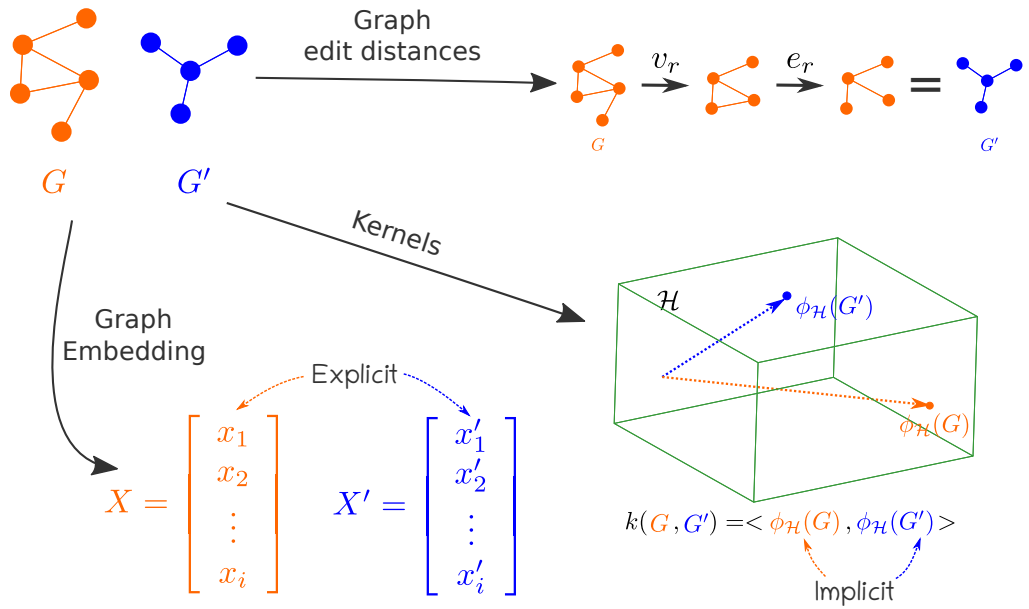


Machine learning on graph

Before the "Deep Learning Era"

Machine Learning on graphs is not a new subject. Graph classification is used in many application since 3/4 decades with 3 main strategies :

- Set graph features as input of stat. ML models (probing/embedding)
- Work in the (dis)similarity space through the GED or its variants
- Use implicit distance through the use of graph Kernels

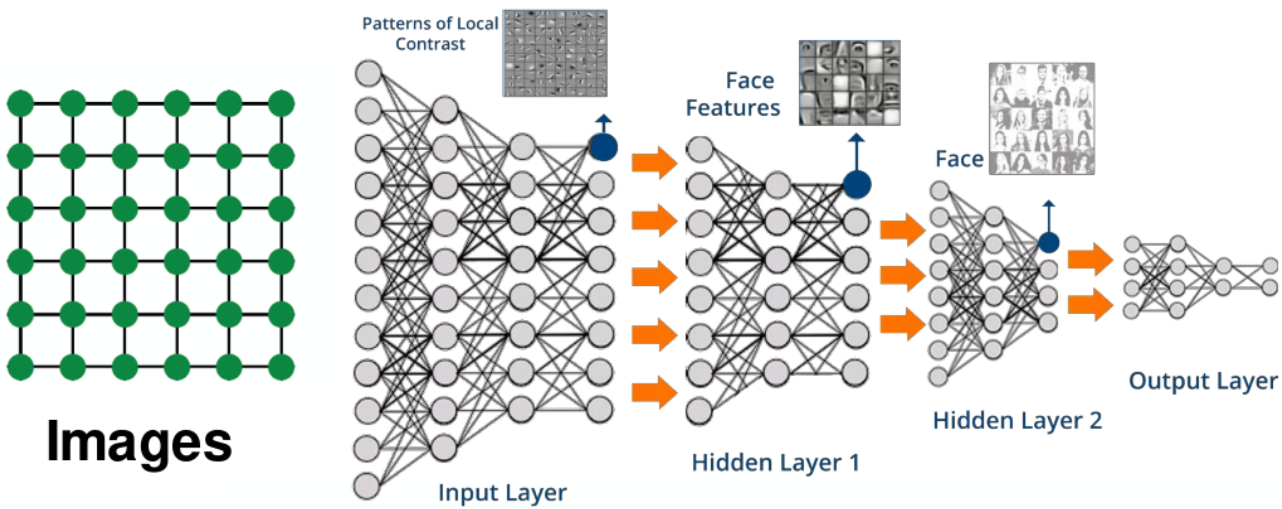


Outline of the lecture

Introduction to Graph Neural Networks

A new trend has emerged 5/6 years ago...

- Can modern and super-human image deep learning approaches (generally based on convolution) be transposed to the world of graphs ?
- Can we translate the key ideas of CNN :

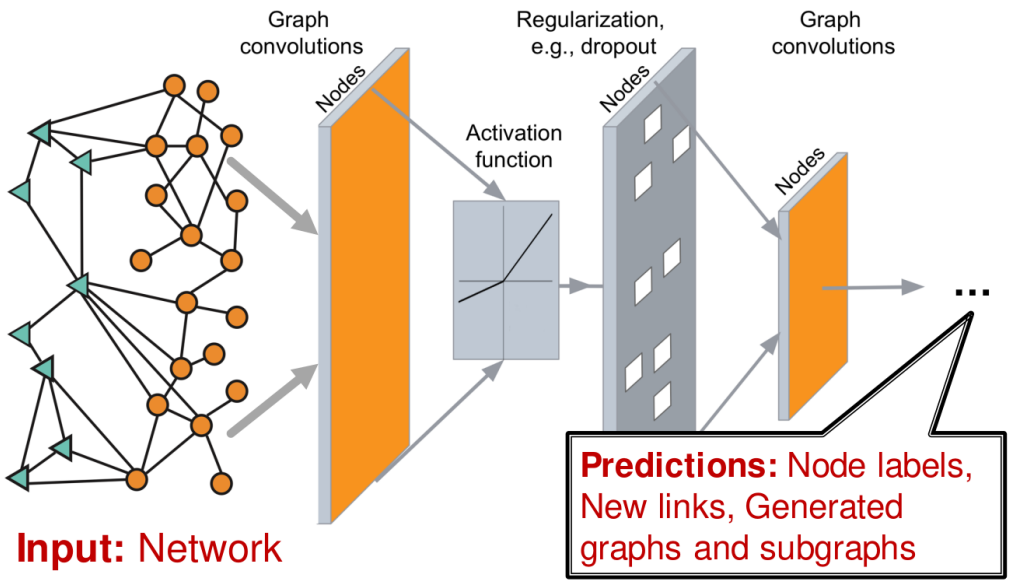


Introduction to Graph Neural Networks

A new trend has emerged 5/6 years ago...

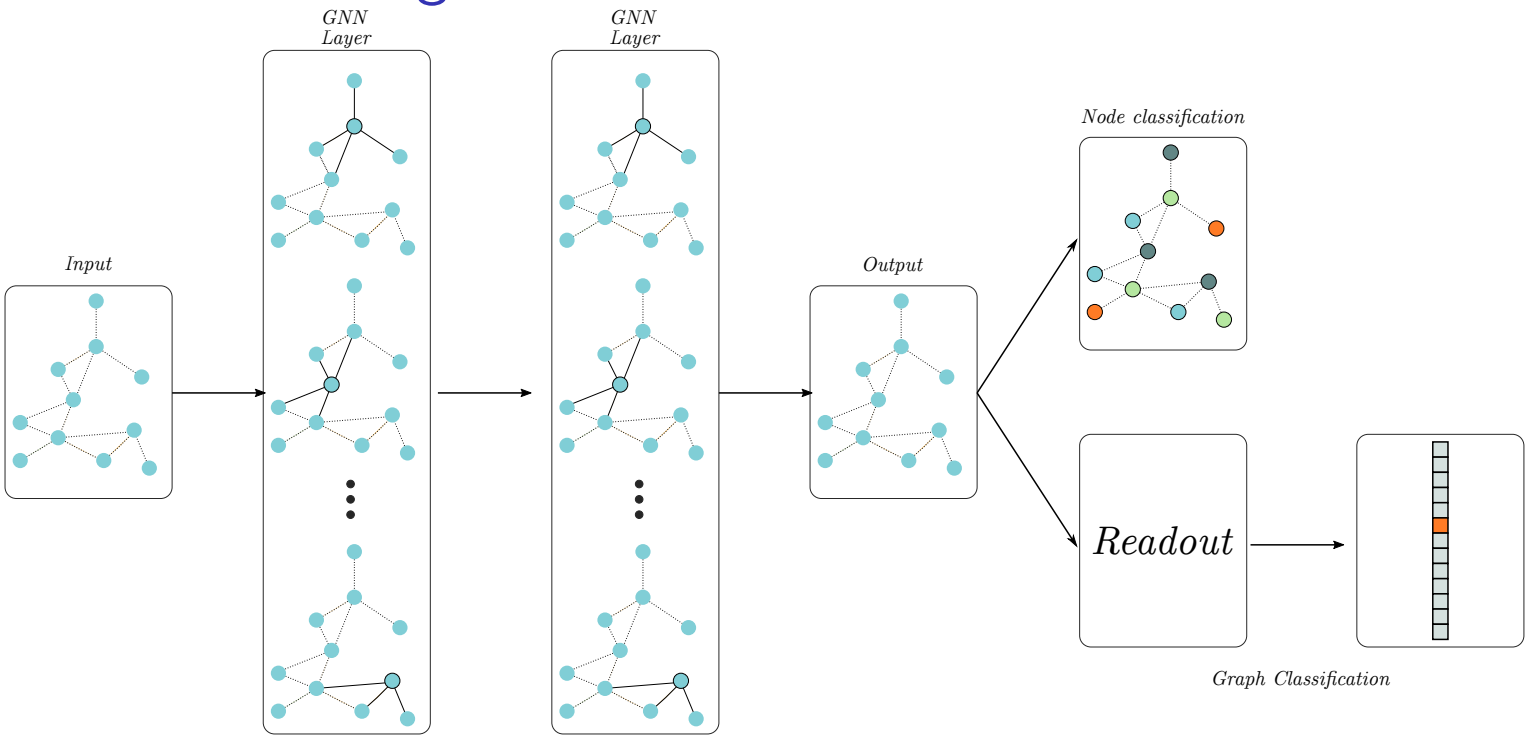
- Can modern and super-human image deep learning approaches (generally based on convolution) be transposed to the world of graphs ?
- Can we translate the key ideas of CNN :

  To something like :



Introduction to Graph Neural Networks

So what do we target ?



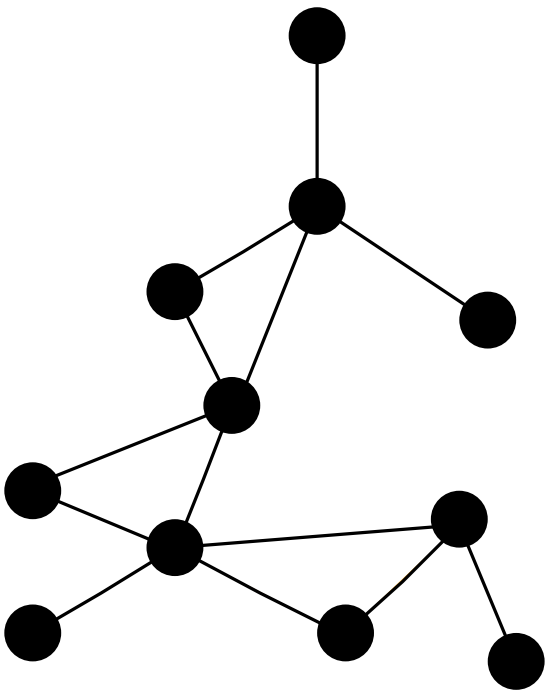
Introduction to Graph Neural Networks

What are the main key concepts of CNNs ?

- **Raw data (matrices or tensors) as input** : the representation space is learnt through layers → no hand-made feature computation
- **Weight sharing (≠ MLP)** : guided by the symmetries of the problem (inductive bias). For CNN : **translation-equivariance**
- **Parameterized differentiable operators** to enable gradient backpropagation
  - Convolutions
  - Pooling
  - Normalizations, skip connections...

Introduction to Graph Neural Networks

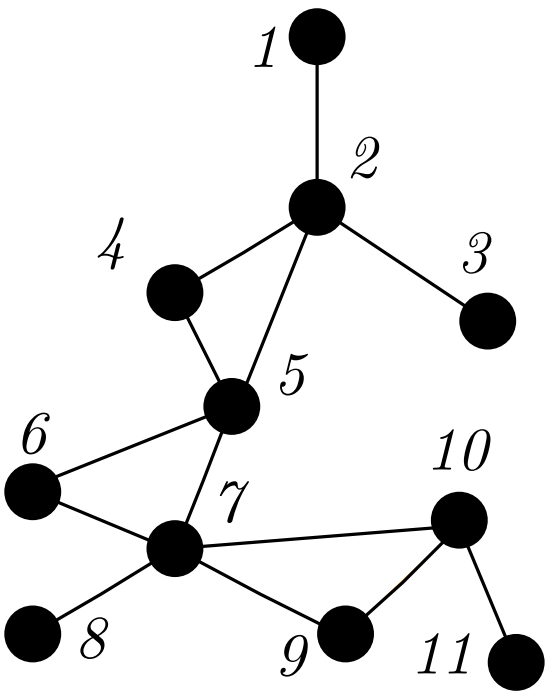
Graph representation as input of a differentiable layer sequence ?

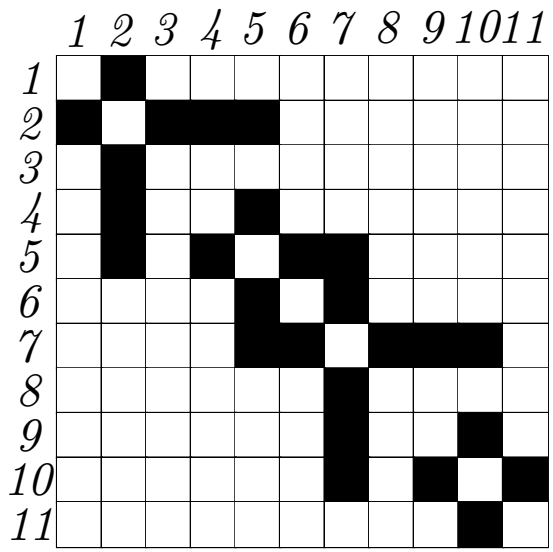


Introduction to Graph Neural Networks

Graph representation as input of a differentiable layer sequence ?

- "An" adjacency matrix $\mathbf{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$

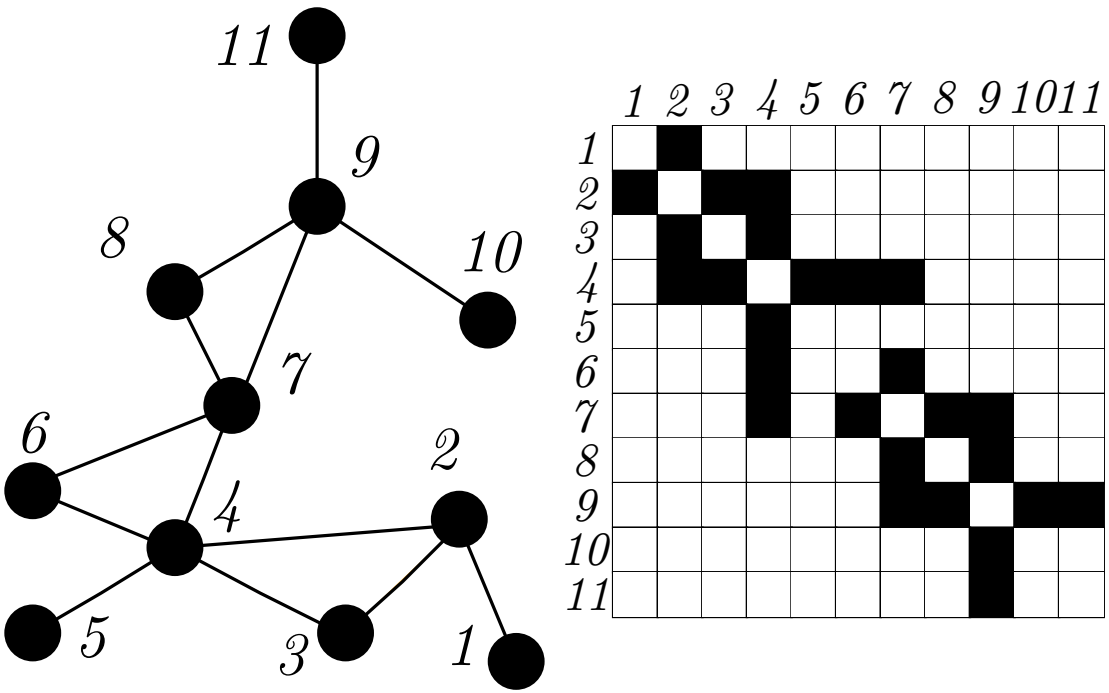




Introduction to Graph Neural Networks

Graph representation as input of a differentiable layer sequence ?

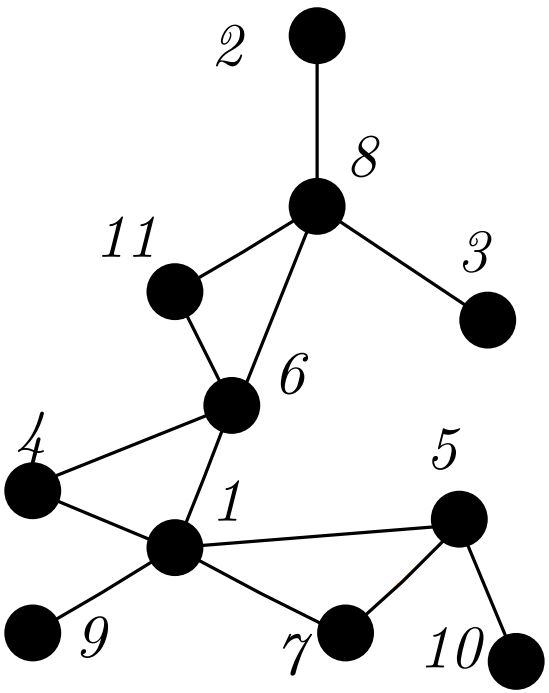
- "An" adjacency matrix $\mathbf{A} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$

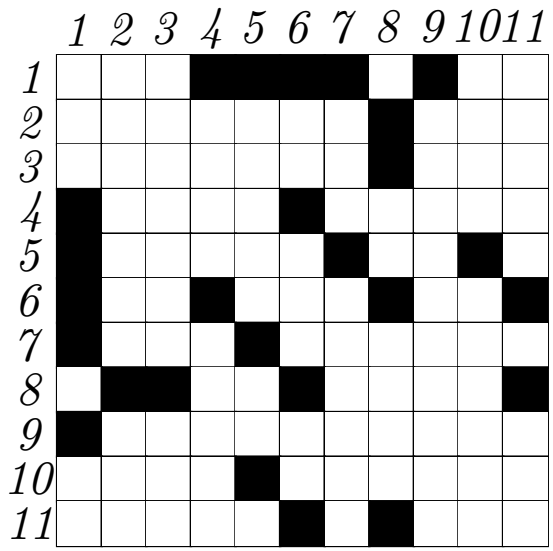


Introduction to Graph Neural Networks

Graph representation as input of a differentiable layer sequence ?

- "An" adjacency matrix $\mathbf{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$

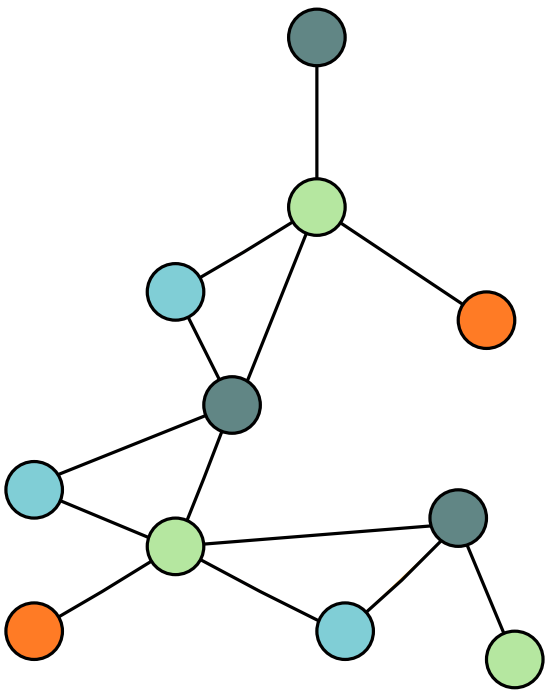


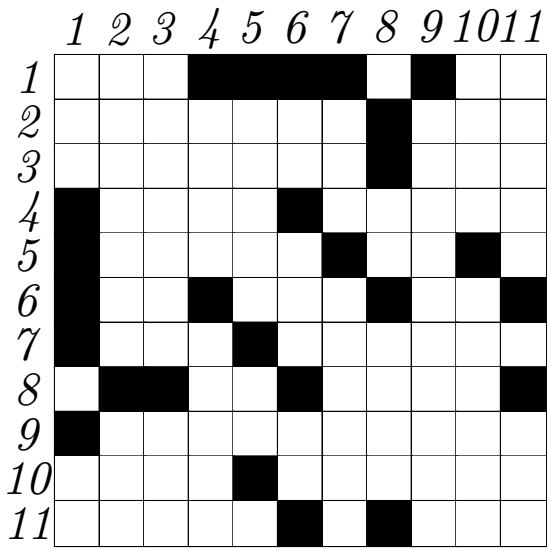


Introduction to Graph Neural Networks

Graph representation as input of a differentiable layer sequence ?

- "An" adjacency matrix $\mathbf{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$
- The node signal $\mathbf{H} \in \{\mathcal{L}_{\mathcal{V}}\}^{|\mathcal{V}|}$

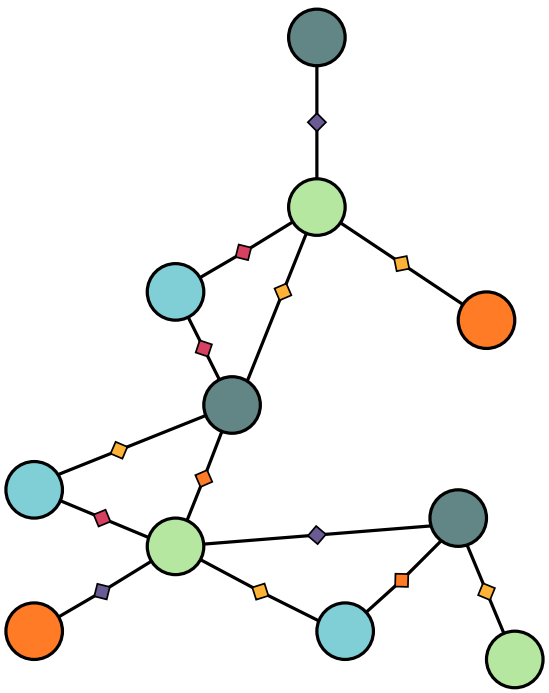


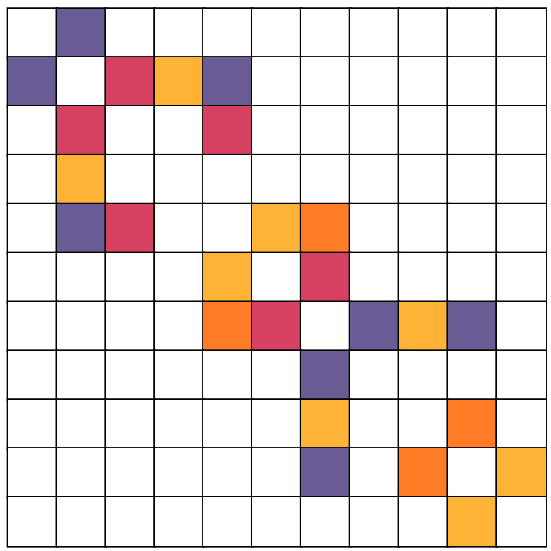


Introduction to Graph Neural Networks

Graph representation as input of a differentiable layer sequence ?

- "An" adjacency matrix $\mathbf{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$
- The node signal $\mathbf{H} \in \{\mathcal{L}_{\mathcal{V}}\}^{|\mathcal{V}|}$
- The edge signal : Replacing the matrix $\mathbf{A}$ by $\mathbf{A}_{\mathbf{e}} \in \{\mathcal{L}_{\mathcal{E}}\} \cup \{0\}^{|\mathcal{V}|\times|\mathcal{V}|}$





Introduction to Graph Neural Networks

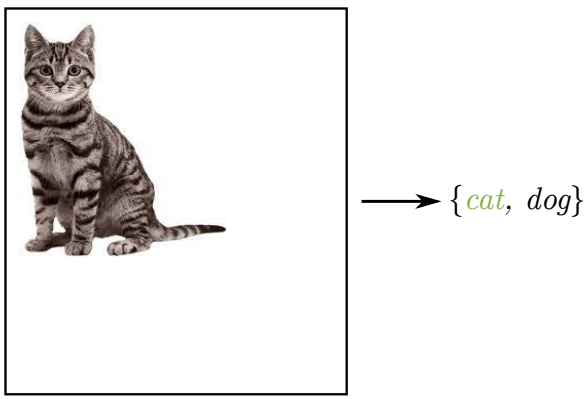
Why convolutions ?

- Because convolutions are translation invariant / equivariant operations
- Let's recall the principles of invariance / equivariance

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

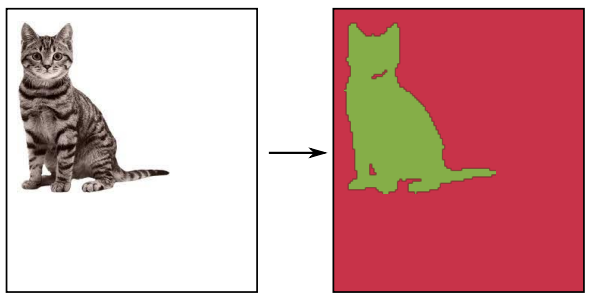
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Introduction to Graph Neural Networks

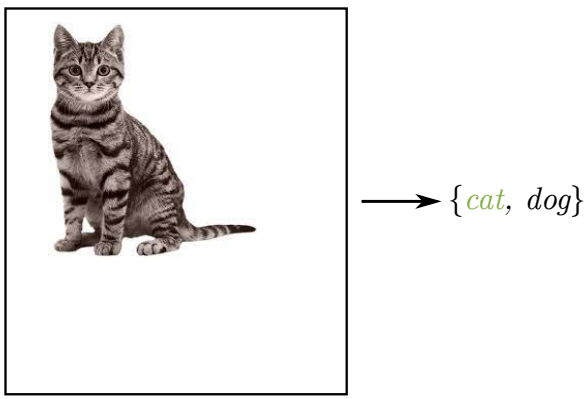
Why convolutions ?

- Because convolutions are translation invariant / equivariant operations
- Let's recall the principles of invariance / equivariance

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

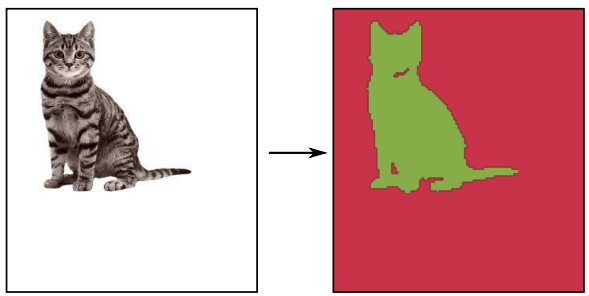
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Introduction to Graph Neural Networks

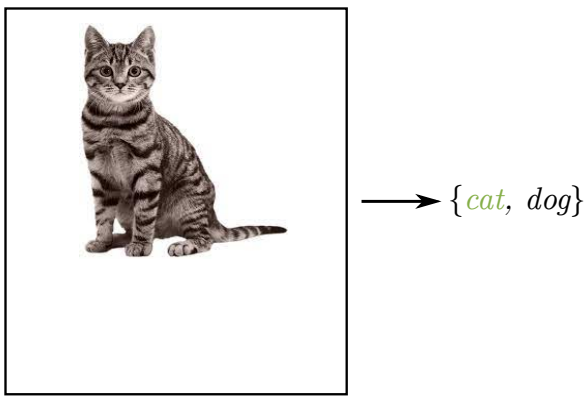
Why convolutions ?

- Because convolutions are translation invariant / equivariant operations
- Let's recall the principles of invariance / equivariance

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

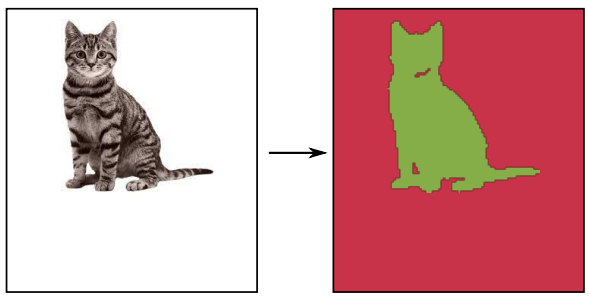
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Introduction to Graph Neural Networks

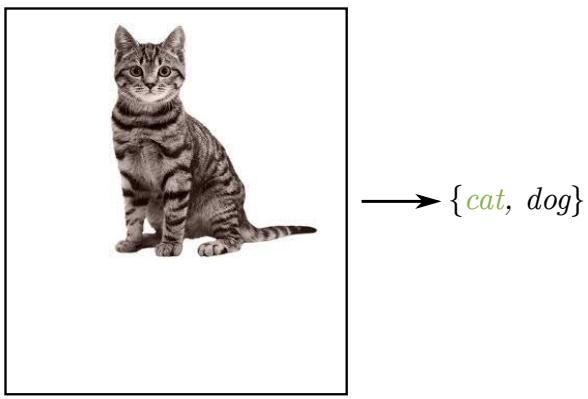
Why convolutions ?

- Because convolutions are translation invariant / equivariant operations
- Let's recall the principles of invariance / equivariance

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

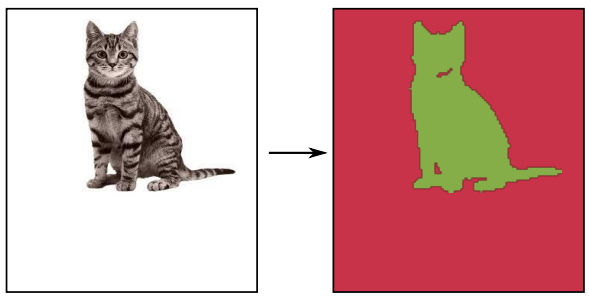
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Introduction to Graph Neural Networks

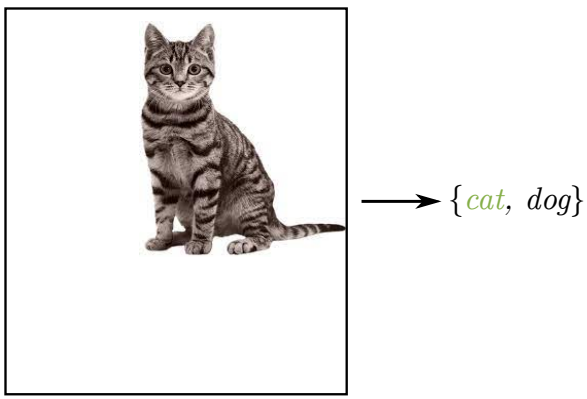
Why convolutions ?

- Because convolutions are translation invariant / equivariant operations
- Let's recall the principles of invariance / equivariance

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

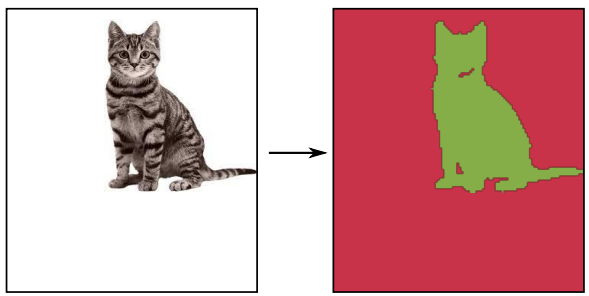
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Introduction to Graph Neural Networks

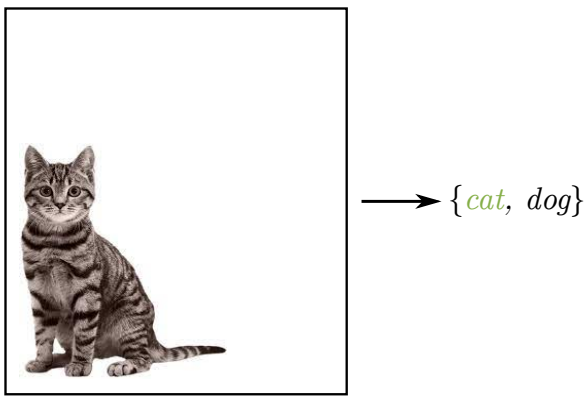
Why convolutions ?

- Because convolutions are translation invariant / equivariant operations
- Let's recall the principles of invariance / equivariance

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

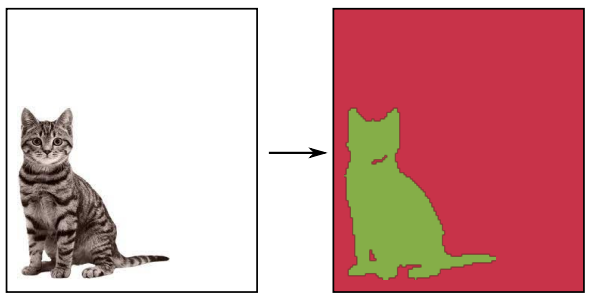
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Introduction to Graph Neural Networks

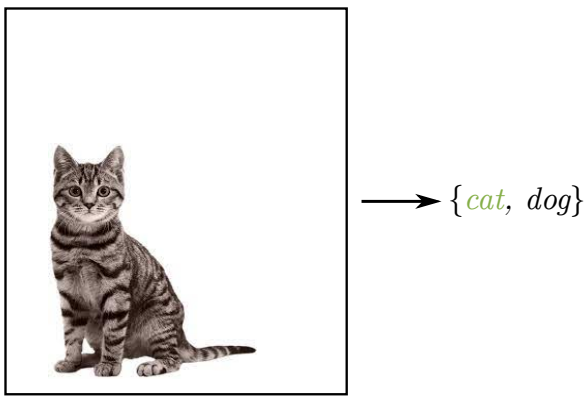
Why convolutions ?

- Because convolutions are translation invariant / equivariant operations
- Let's recall the principles of invariance / equivariance

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

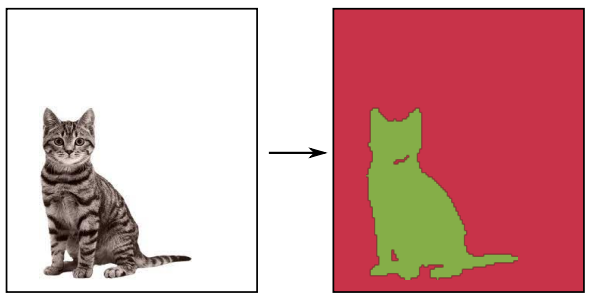
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Introduction to Graph Neural Networks

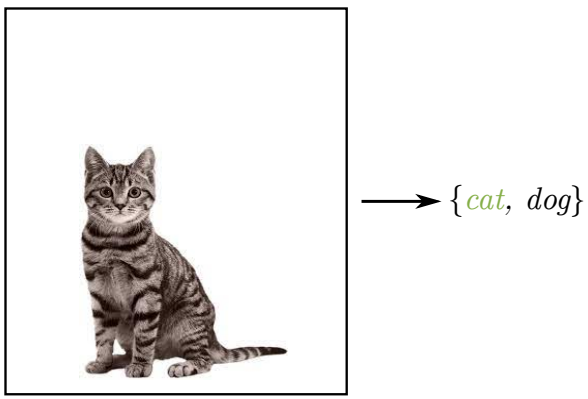
Why convolutions ?

- Because convolutions are translation invariant / equivariant operations
- Let's recall the principles of invariance / equivariance

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

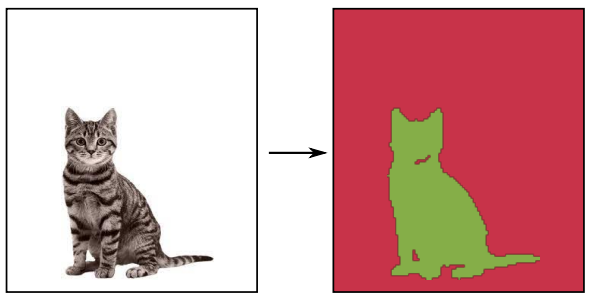
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Introduction to Graph Neural Networks

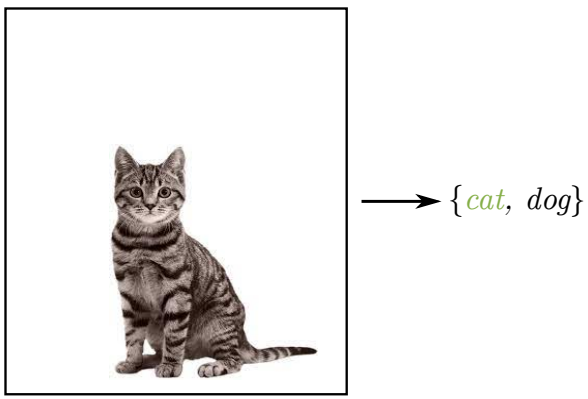
Why convolutions ?

- Because convolutions are translation invariant / equivariant operations
- Let's recall the principles of invariance / equivariance

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

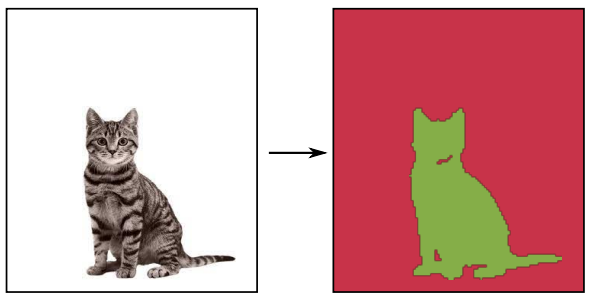
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Introduction to Graph Neural Networks

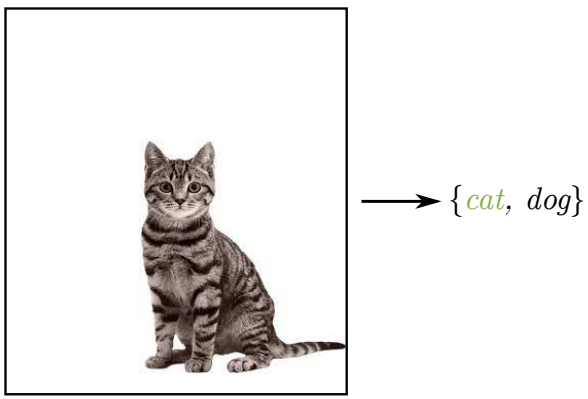
Why convolutions ?

- Because convolutions are translation invariant / equivariant operations
- Let's recall the principles of invariance / equivariance

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

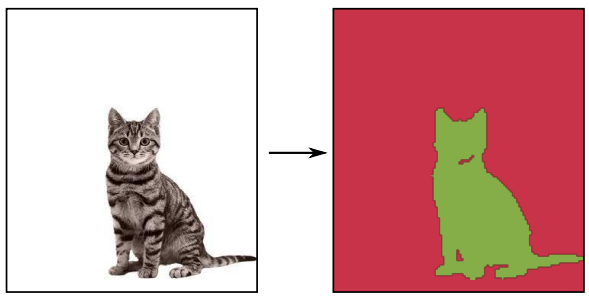
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



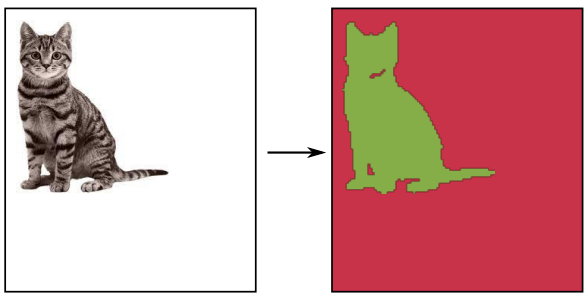
Introduction to Graph Neural Networks

How sharing weight : which equivariance ?

Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

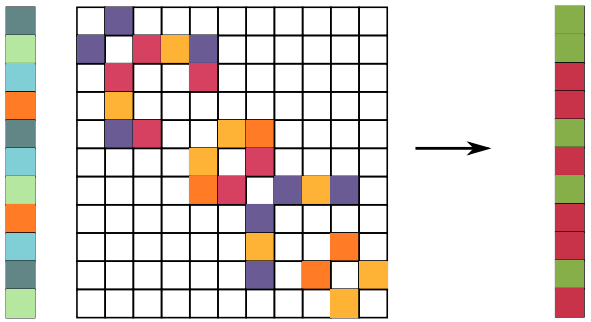
$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Permutation Equivariance :

$$G \to \{\blacksquare, \blacksquare\}^{|\mathcal{V}|}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = \mathbf{P}f(\mathbf{A}, \mathbf{H})$$



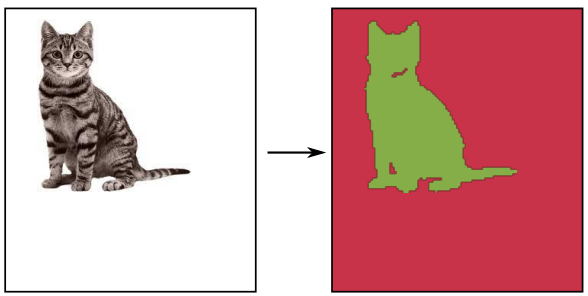
Introduction to Graph Neural Networks

How sharing weight : which equivariance ?

Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

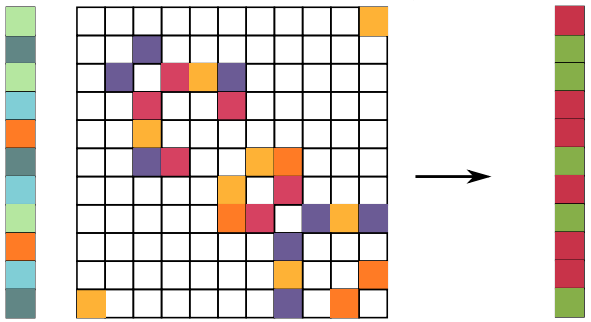
$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Permutation Equivariance :

$$G \to \{\blacksquare, \blacksquare\}^{|\mathcal{V}|}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = \mathbf{P}f(\mathbf{A}, \mathbf{H})$$



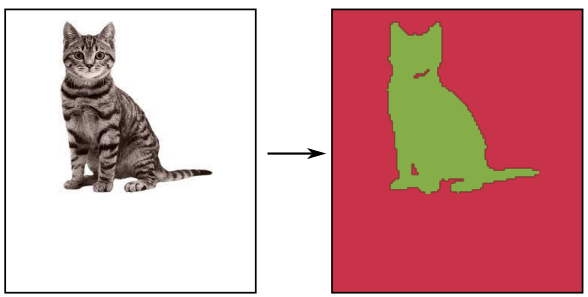
Introduction to Graph Neural Networks

How sharing weight : which equivariance ?

Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

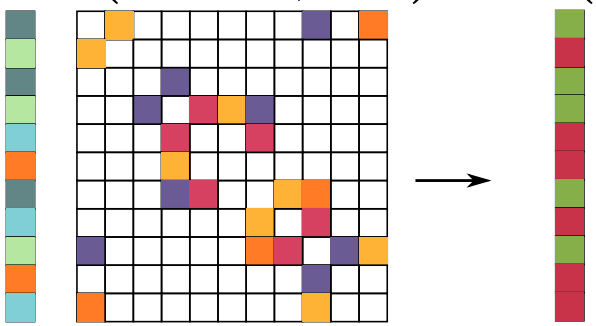
$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Permutation Equivariance :

$$G \to \{\blacksquare, \blacksquare\}^{|\mathcal{V}|}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = \mathbf{P}f(\mathbf{A}, \mathbf{H})$$



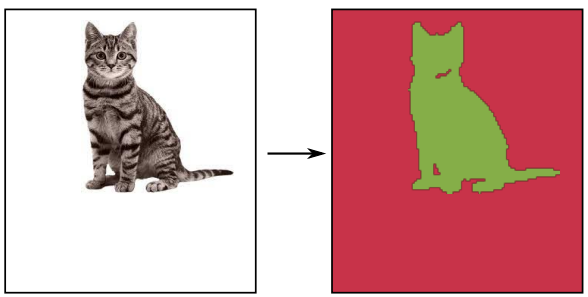
Introduction to Graph Neural Networks

How sharing weight : which equivariance ?

Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

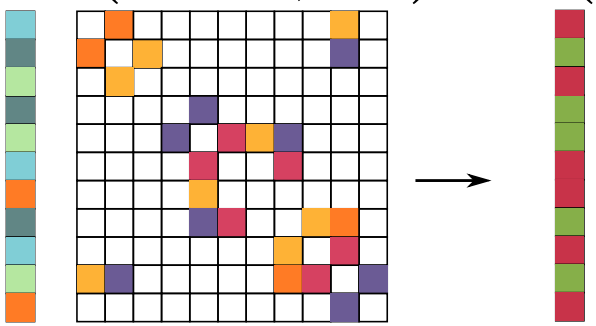
$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Permutation Equivariance :

$$G \to \{\blacksquare, \blacksquare\}^{|\mathcal{V}|}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = \mathbf{P}f(\mathbf{A}, \mathbf{H})$$



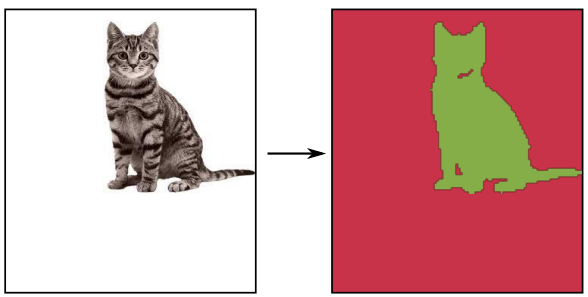
Introduction to Graph Neural Networks

How sharing weight : which equivariance ?

Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

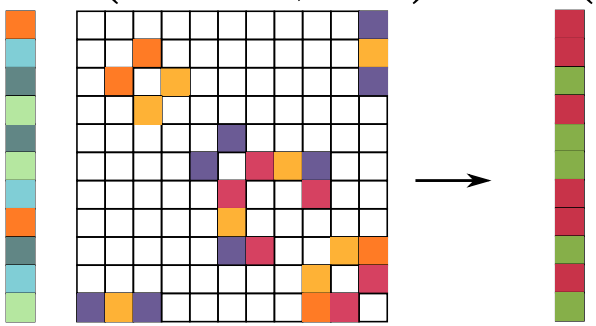
$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Permutation Equivariance :

$$G \to \{\blacksquare, \blacksquare\}^{|\mathcal{V}|}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = \mathbf{P}f(\mathbf{A}, \mathbf{H})$$



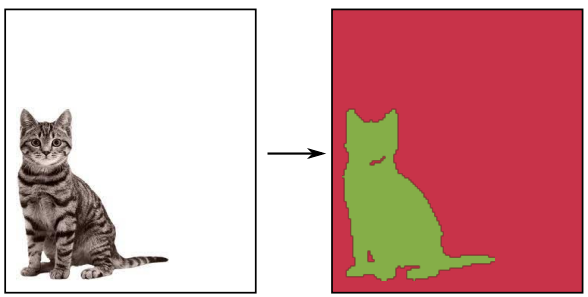
Introduction to Graph Neural Networks

How sharing weight : which equivariance ?

Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

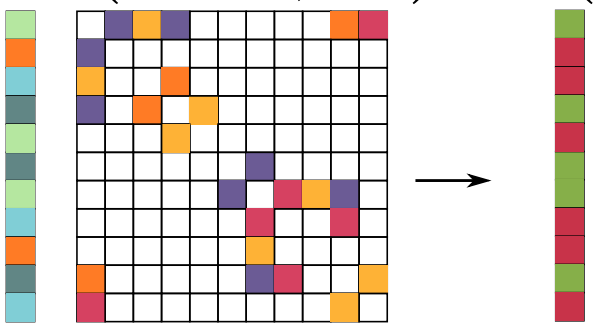
$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Permutation Equivariance :

$$G \to \{\blacksquare, \blacksquare\}^{|\mathcal{V}|}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = \mathbf{P}f(\mathbf{A}, \mathbf{H})$$



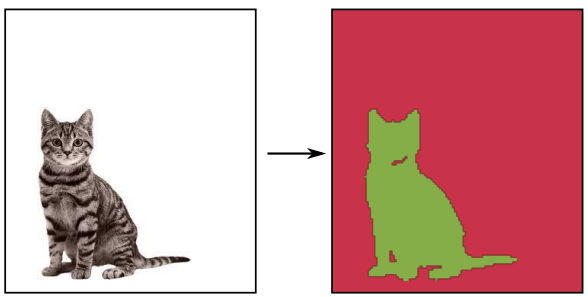
Introduction to Graph Neural Networks

How sharing weight : which equivariance ?

Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

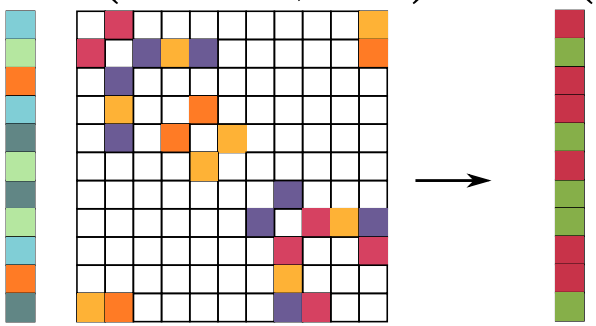
$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Permutation Equivariance :

$$G \to \{\blacksquare, \blacksquare\}^{|\mathcal{V}|}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = \mathbf{P}f(\mathbf{A}, \mathbf{H})$$



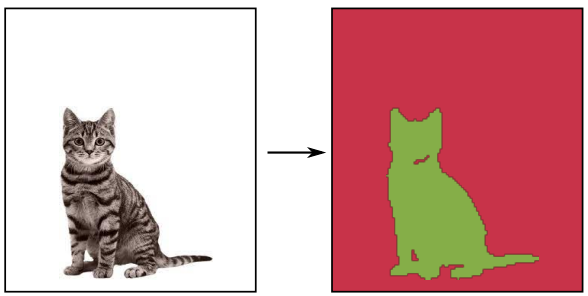
Introduction to Graph Neural Networks

How sharing weight : which equivariance ?

Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

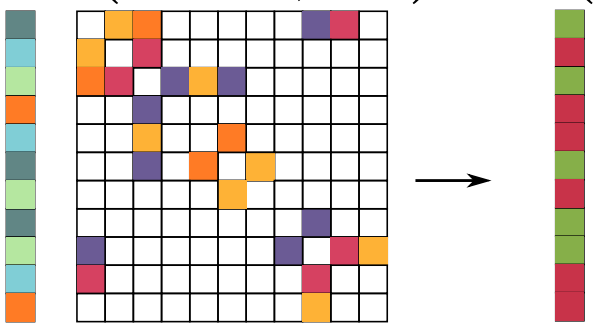
$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Permutation Equivariance :

$$G \to \{\blacksquare, \blacksquare\}^{|\mathcal{V}|}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = \mathbf{P}f(\mathbf{A}, \mathbf{H})$$



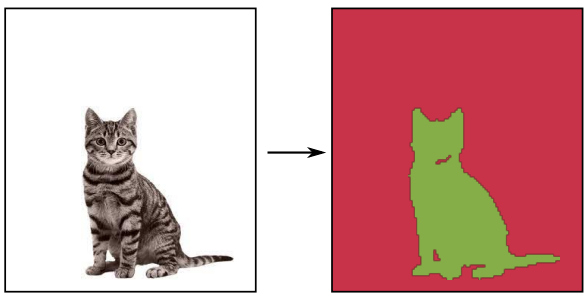
Introduction to Graph Neural Networks

How sharing weight : which equivariance ?

Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

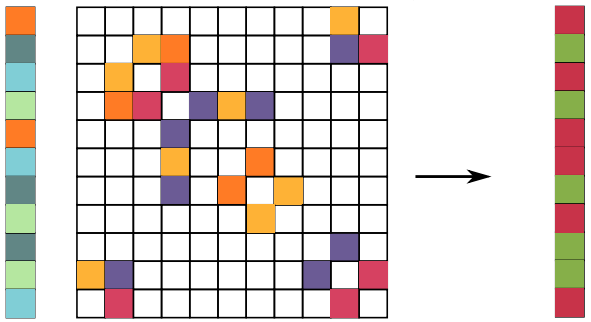
$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Permutation Equivariance :

$$G \to \{\blacksquare, \blacksquare\}^{|\mathcal{V}|}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = \mathbf{P}f(\mathbf{A}, \mathbf{H})$$



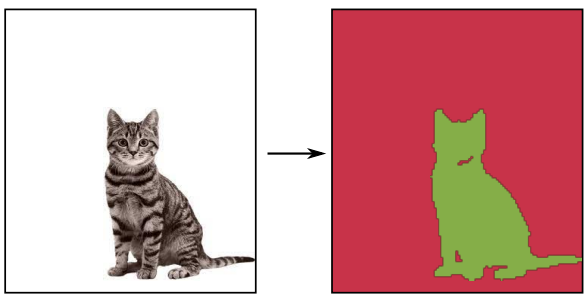
Introduction to Graph Neural Networks

How sharing weight : which equivariance ?

Translation Equivariance :

$$\mathbb{I} \to \{\blacksquare, \blacksquare\}^{m \times n}$$

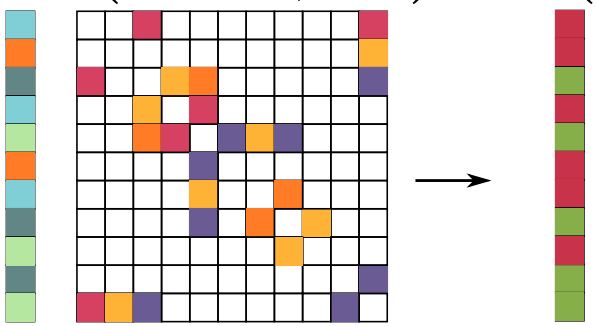
$$f(t(\mathbf{X})) = t(f(\mathbf{X}))$$



Permutation Equivariance :

$$G \to \{\blacksquare, \blacksquare\}^{|\mathcal{V}|}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = \mathbf{P}f(\mathbf{A}, \mathbf{H})$$



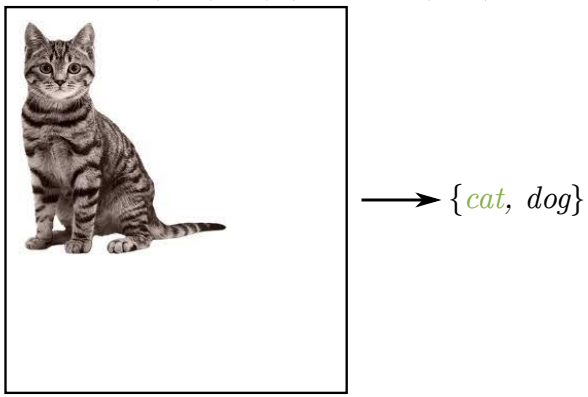
Introduction to Graph Neural Networks

Which group invariance if decision are taken at the graph level ?

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

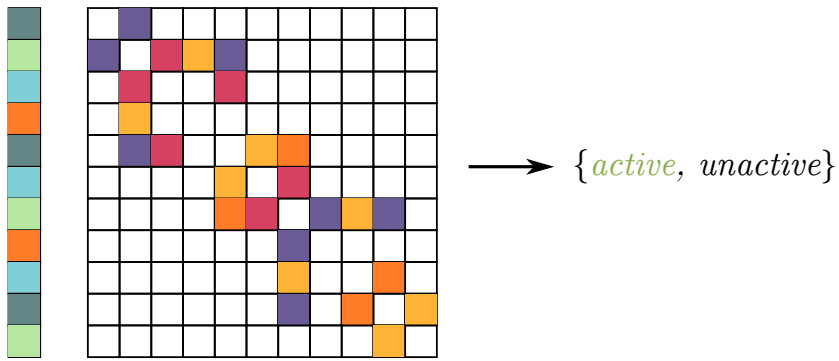
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Permutation Invariance :

$$G \to \{active, unactive\}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = f(\mathbf{A}, \mathbf{H})$$



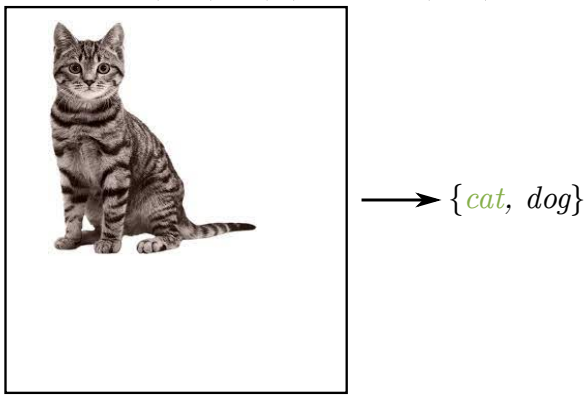
Introduction to Graph Neural Networks

Which group invariance if decision are taken at the graph level ?

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

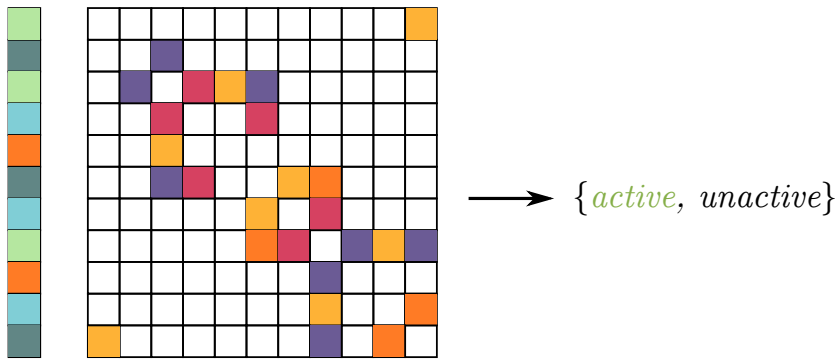
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Permutation Invariance :

$$G \to \{active, unactive\}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = f(\mathbf{A}, \mathbf{H})$$



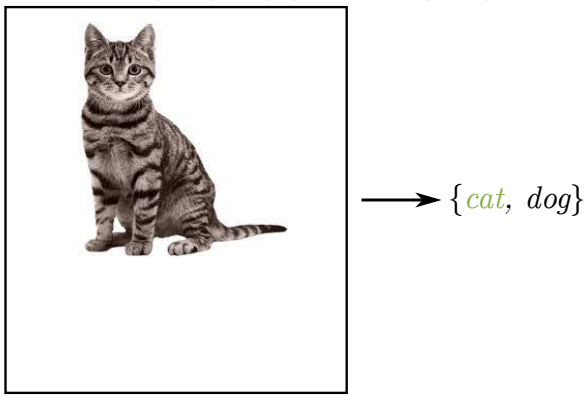
Introduction to Graph Neural Networks

Which group invariance if decision are taken at the graph level ?

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

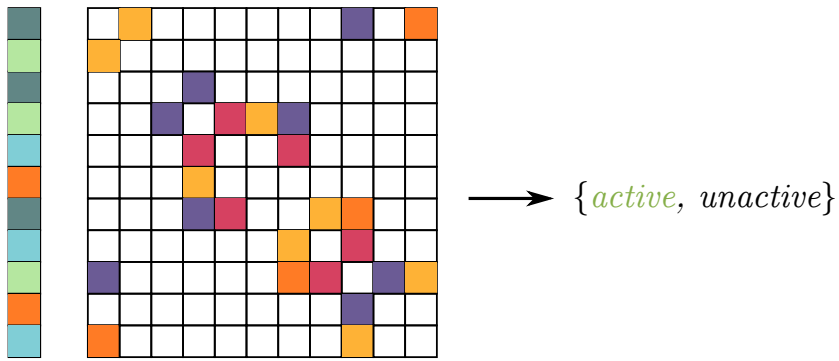
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Permutation Invariance :

$$G \to \{active, unactive\}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = f(\mathbf{A}, \mathbf{H})$$



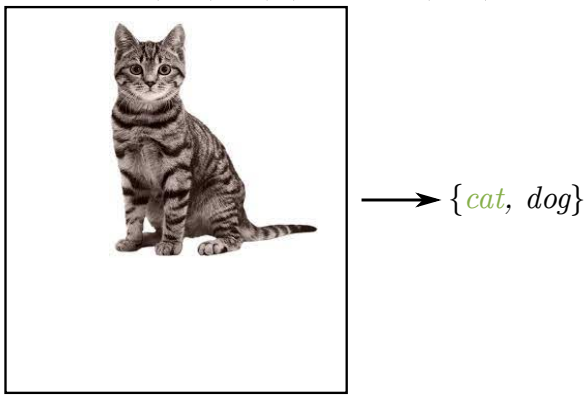
Introduction to Graph Neural Networks

Which group invariance if decision are taken at the graph level ?

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

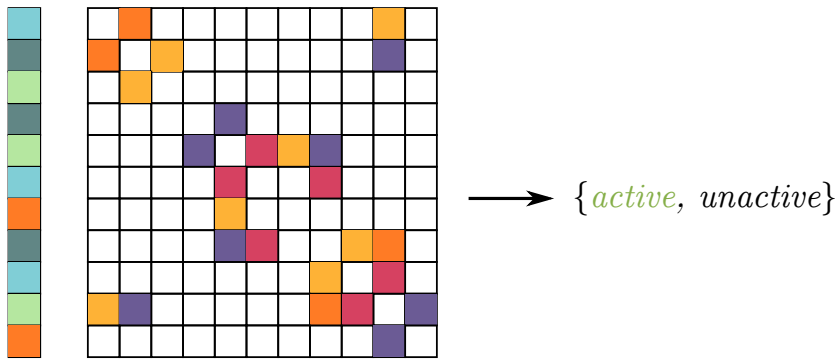
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Permutation Invariance :

$$G \to \{active, unactive\}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = f(\mathbf{A}, \mathbf{H})$$



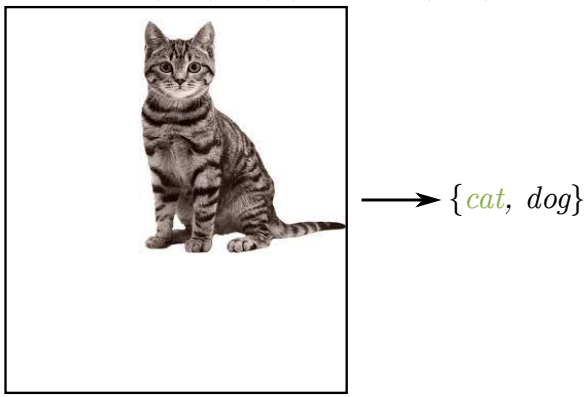
Introduction to Graph Neural Networks

Which group invariance if decision are taken at the graph level ?

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

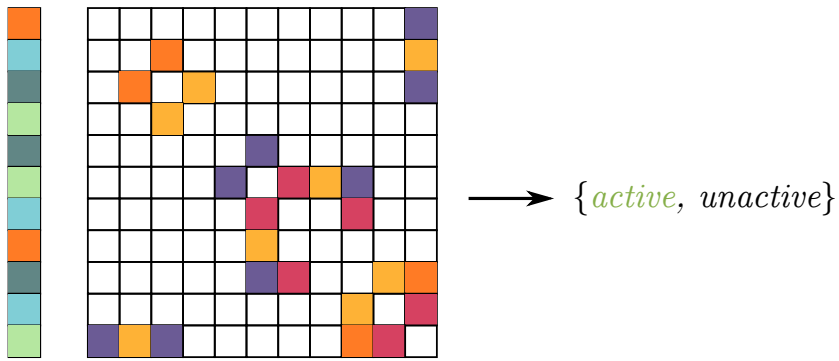
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Permutation Invariance :

$$G \to \{active, unactive\}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = f(\mathbf{A}, \mathbf{H})$$



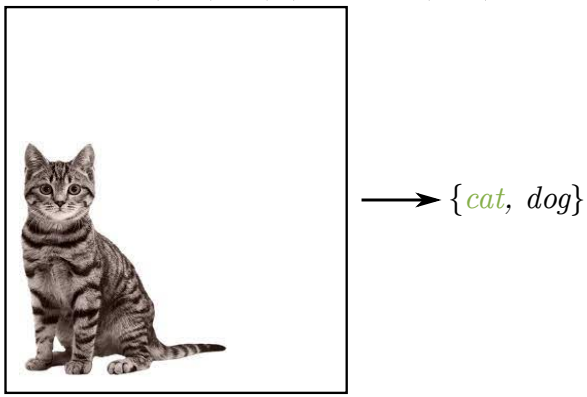
Introduction to Graph Neural Networks

Which group invariance if decision are taken at the graph level ?

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

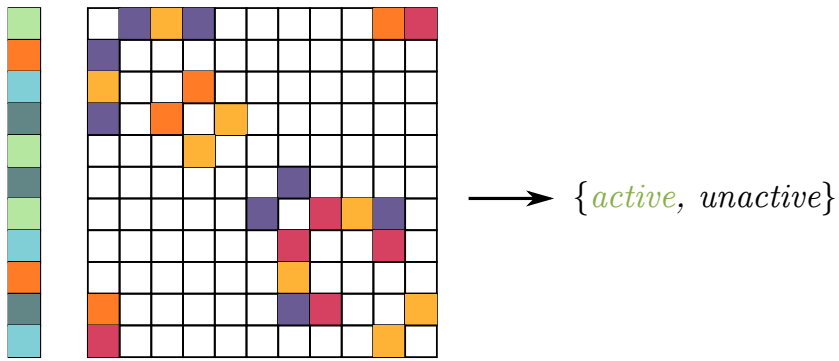
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Permutation Invariance :

$$G \to \{active, unactive\}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = f(\mathbf{A}, \mathbf{H})$$



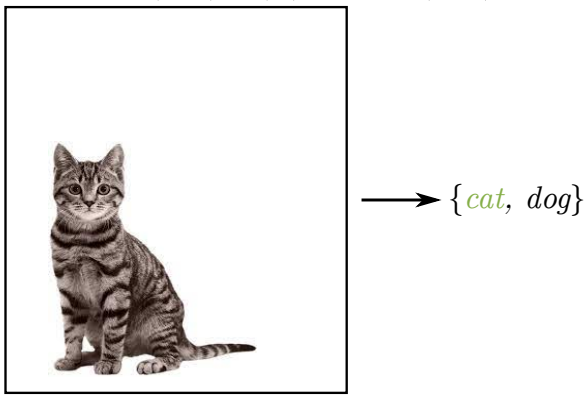
Introduction to Graph Neural Networks

Which group invariance if decision are taken at the graph level ?

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

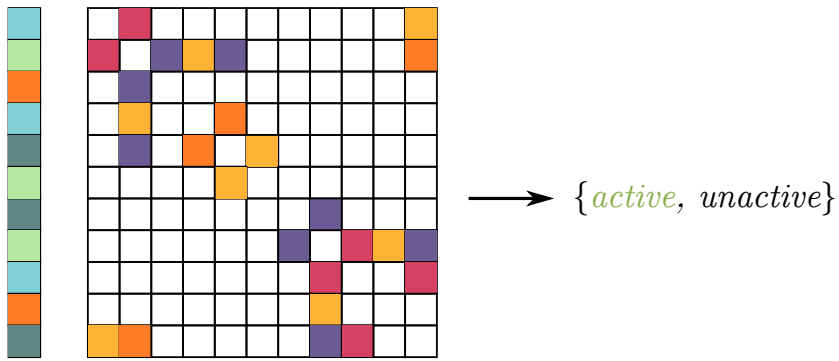
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Permutation Invariance :

$$G \to \{active, unactive\}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = f(\mathbf{A}, \mathbf{H})$$



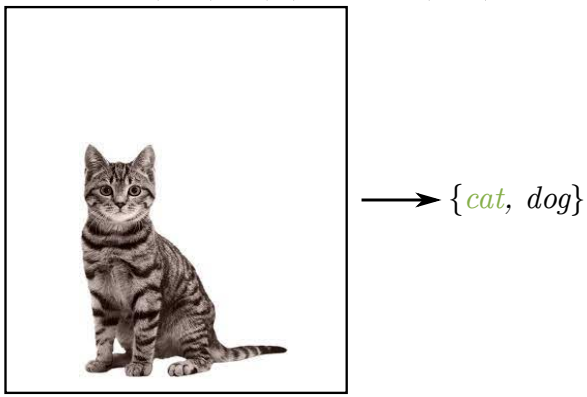
Introduction to Graph Neural Networks

Which group invariance if decision are taken at the graph level ?

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

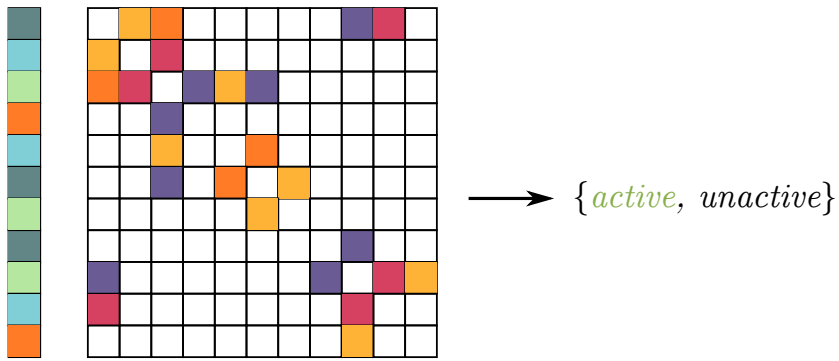
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Permutation Invariance :

$$G \to \{active, unactive\}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = f(\mathbf{A}, \mathbf{H})$$



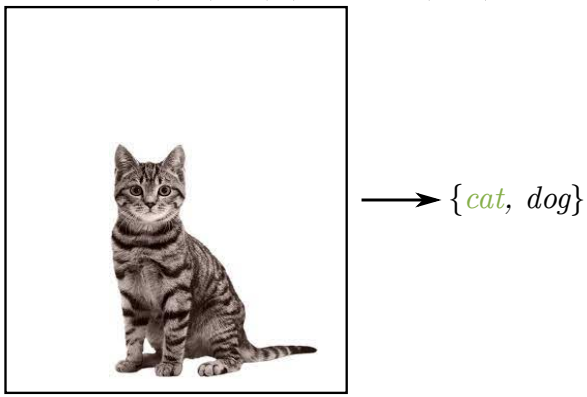
Introduction to Graph Neural Networks

Which group invariance if decision are taken at the graph level ?

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

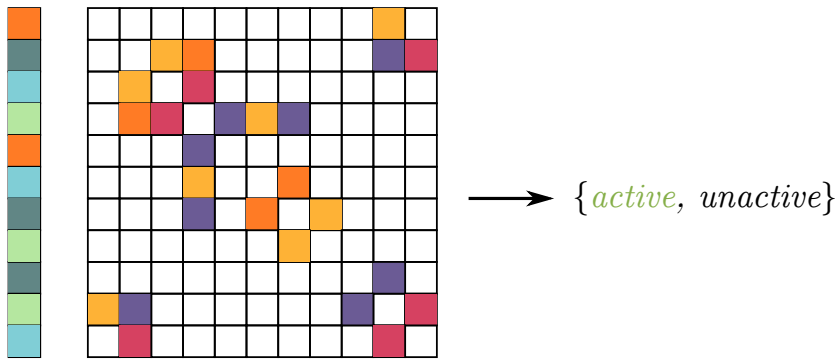
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Permutation Invariance :

$$G \to \{active, unactive\}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = f(\mathbf{A}, \mathbf{H})$$



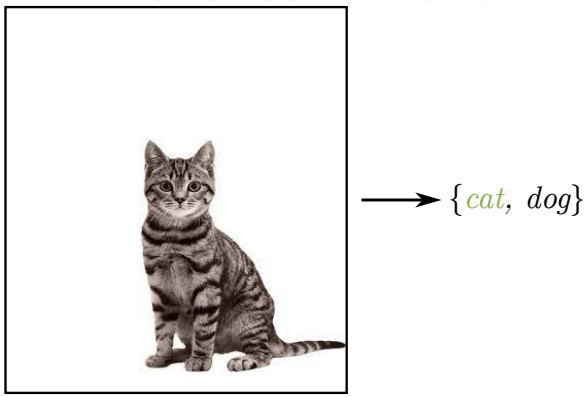
Introduction to Graph Neural Networks

Which group invariance if decision are taken at the graph level ?

Translation Invariance :

$$\mathbb{I} \to \{cat, dog\}$$

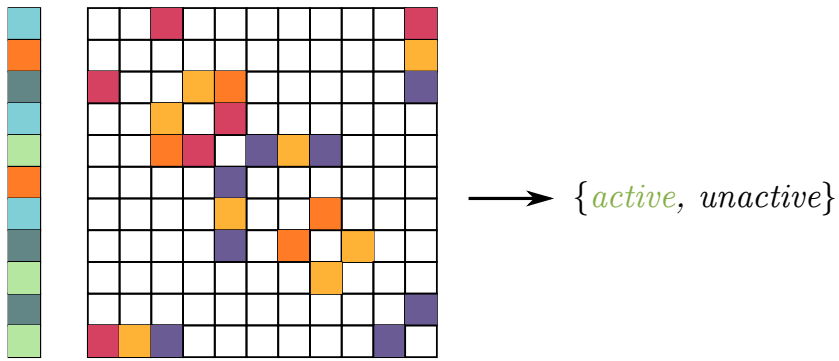
$$f(t(\mathbf{X})) = f(\mathbf{X})$$



Permutation Invariance :

$$G \to \{active, unactive\}$$

$$f(\mathbf{P}^T\mathbf{A}\mathbf{P}, \mathbf{P}\mathbf{H}) = f(\mathbf{A}, \mathbf{H})$$



Introduction to Graph Neural Networks

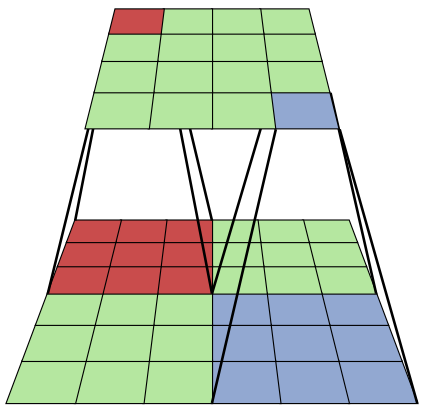
Ok : we need equivariant "convolutions" on graphs !

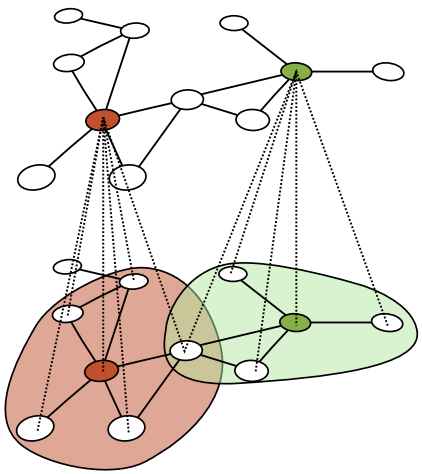
Convolutions on images are well defined.

- A pixel always has the same number of neighbors
- Neighbors weights can be linked to a relative position.

Convolutions on graphs :

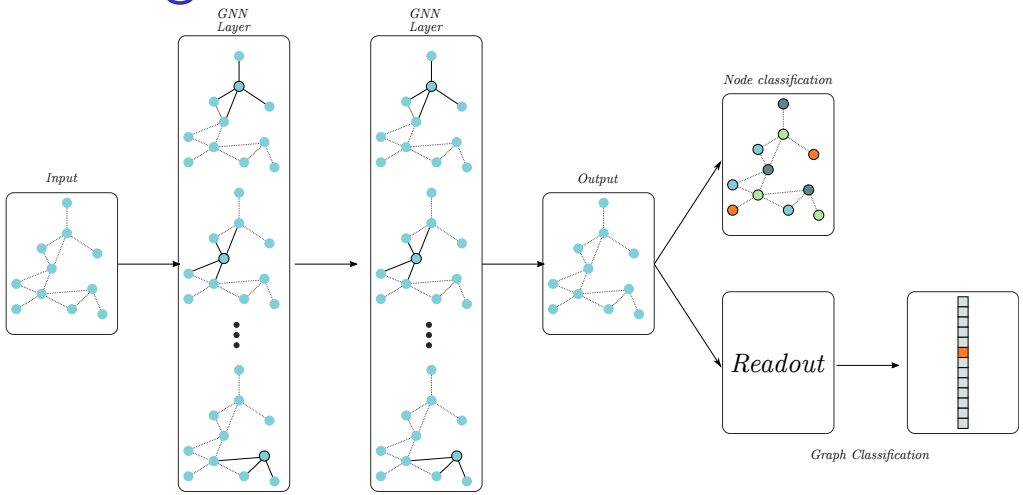
- A node does not always have the same number of neighbors
- No relative position on neighborhood





Introduction to Graph Neural Networks

So what do we target ?



- A GNN layer updates each node hidden states h_u (for node u) using :

$$h_u^{k+1} = f_k(h_u^{(k)}, \{h_v^{(k)} \forall v \in |\mathcal{V}|\}, \{e_{v_i,v_j} \forall (v_i, v_j) \in |\mathcal{E}|\})$$

- Two strategies co-exist for the design of the fonction f_k :
  - Spatial design
  - Spectral design

Outline of the lecture

Spatial GNNs

General framework : Message Passing Neural Networks (MPNN)

The equation of the preceding slide

$$h_u^{k+1} = f_k(h_u^{(k)}, \{h_v^{(k)} \forall v \in |\mathcal{V}|\}, \{e_{v_i,v_j} \forall (v_i, v_j) \in |\mathcal{E}|\})$$

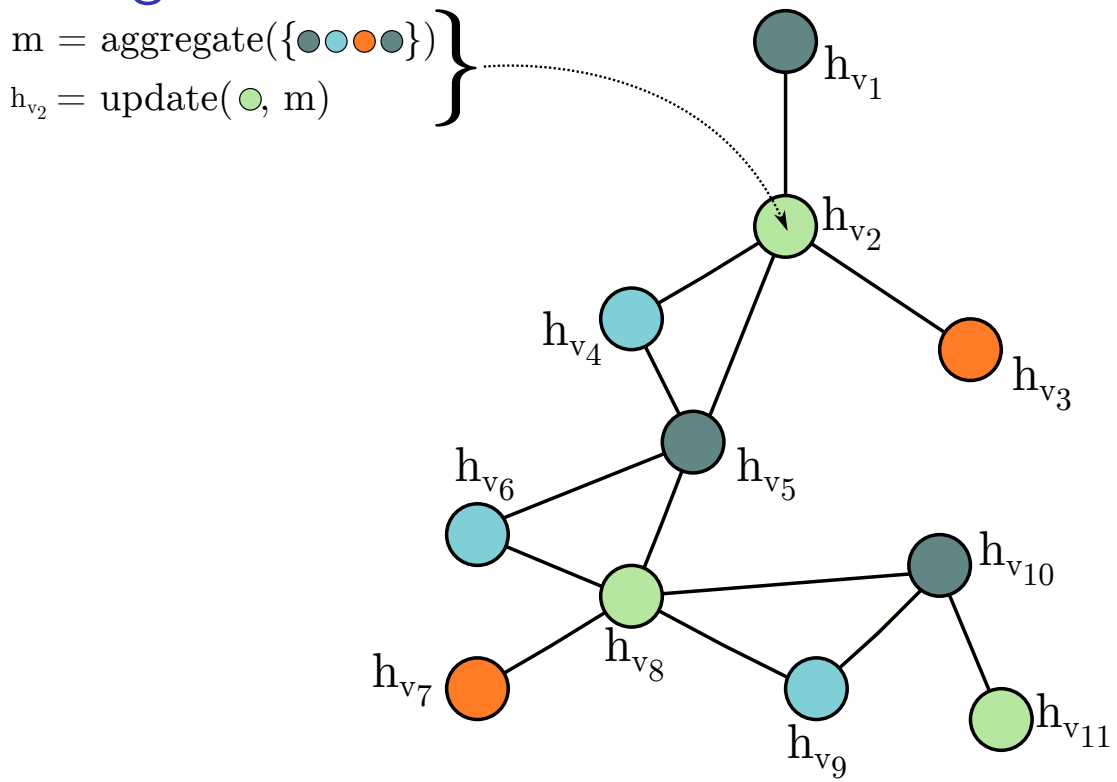
is generally written as a composition of functions `update` and `aggregate`

$$\mathbf{h}_u^{(l+1)} = \texttt{update}(\mathbf{h}_u^{(l)}, \texttt{aggregate}(\{\mathbf{h}_v^{(l)}, \forall v \in \mathcal{N}(u)\}))$$

- `update` and `aggregate` functions define the GNN behaviour
- `aggregate` function must be **invariant** to permutations

Spatial GNNs

Message Passing Neural Networks : illustration

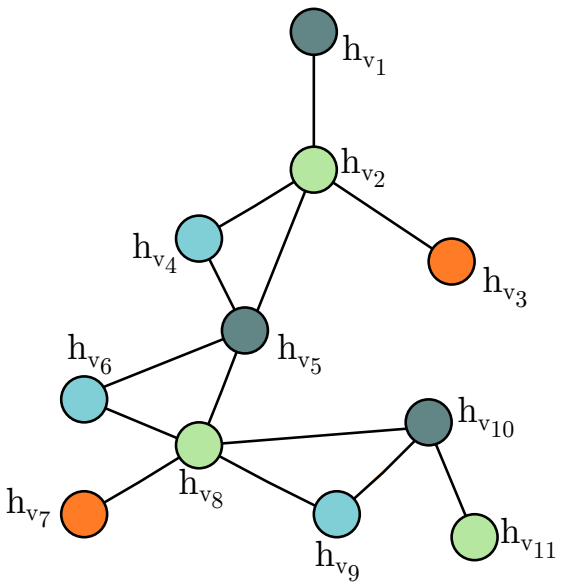


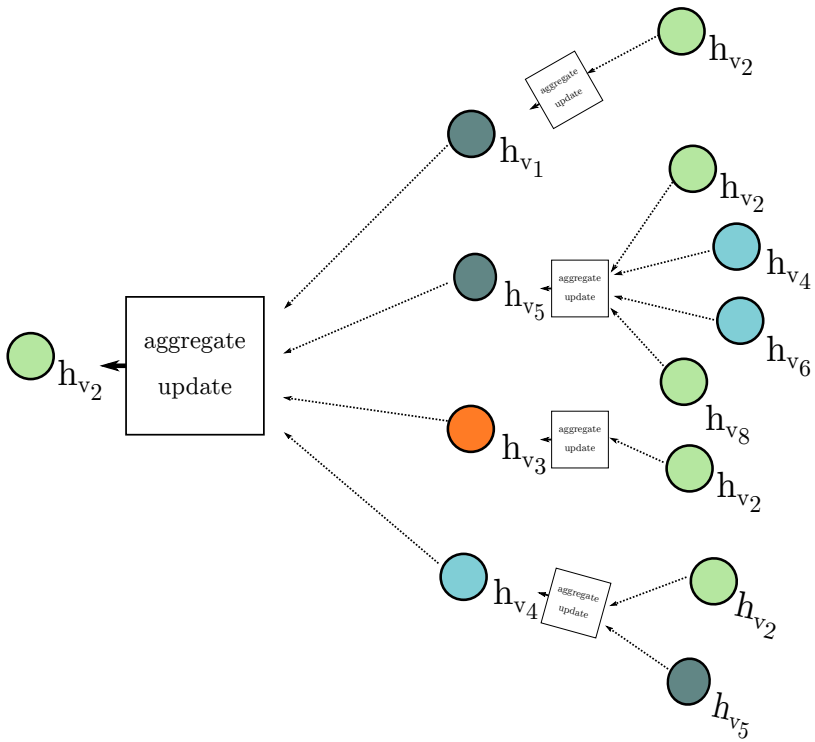
- NB 1 : This figure should remind you of something
- NB 2 : Limited context

Spatial GNNs

Message Passing Neural Networks : illustration

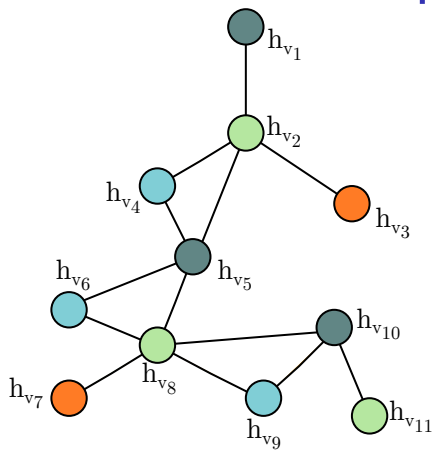
- Adding GNN layers allows to take into account a wider neighborhood
- K layers capture information from a K-hop neighborhood
- Fun Fact : the computation graph is based on the processed graph





Spatial GNNs

Message Passing Neural Networks : implementation tricks

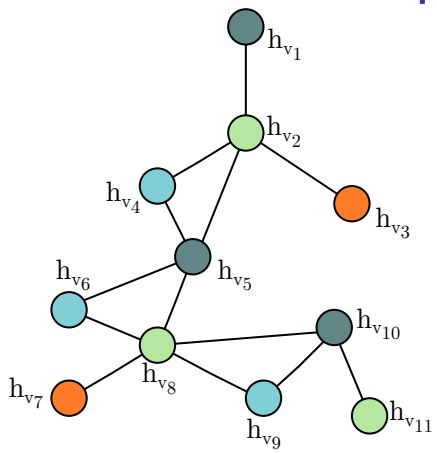


$$\begin{aligned}\mathbf{h}_{v_2}^{l+1} &= \sigma(\mathbf{h}_{v_1}^{l}\mathbf{W}^{l} + \mathbf{h}_{v_3}^{l}\mathbf{W}^{l} + \mathbf{h}_{v_4}^{l}\mathbf{W}^{l} + \mathbf{h}_{v_5}^{l}\mathbf{W}^{l}) \\ &= \sigma\Big(\sum_{v_i \in \mathcal{N}(v_2)} \mathbf{h}_{v_i}^{l}\mathbf{W}^{l}\Big)\end{aligned}$$

Written in a matrix form (using H, I, A and some learnable parameters matrices W), it gives :

Spatial GNNs

Message Passing Neural Networks : implementation tricks



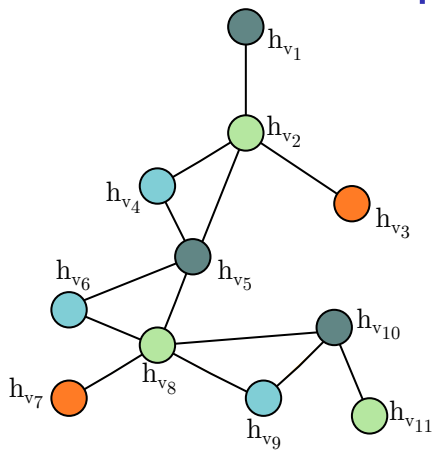
$$\begin{aligned}\mathbf{h}_{v_2}^{l+1} &= \sigma(\mathbf{h}_{v_1}^{l}\mathbf{W}^{l} + \mathbf{h}_{v_3}^{l}\mathbf{W}^{l} + \mathbf{h}_{v_4}^{l}\mathbf{W}^{l} + \mathbf{h}_{v_5}^{l}\mathbf{W}^{l}) \\ &= \sigma\Big(\sum_{v_i \in \mathcal{N}(v_2)} \mathbf{h}_{v_i}^{l}\mathbf{W}^{l}\Big)\end{aligned}$$

Written in a matrix form (using H, I, A and some learnable parameters matrices W), it gives :

- Simplest way : $\mathbf{H}^{l+1} = \sigma(\mathbf{A}\mathbf{H}^{l}\mathbf{W}^{l})$

Spatial GNNs

Message Passing Neural Networks : implementation tricks



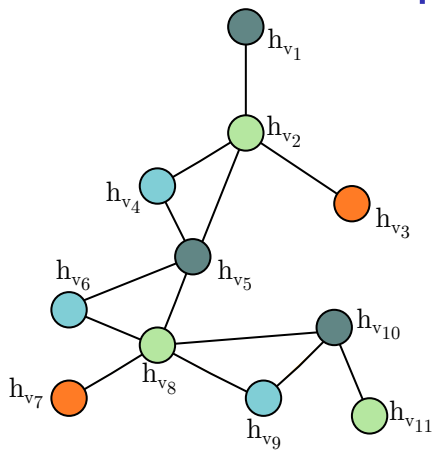
$$\begin{aligned}\mathbf{h}_{v_2}^{l+1} &= \sigma(\mathbf{h}_{v_2}^{l}\mathbf{W}^{l} + \mathbf{h}_{v_1}^{l}\mathbf{W}^{l} + \mathbf{h}_{v_3}^{l}\mathbf{W}^{l} + \mathbf{h}_{v_4}^{l}\mathbf{W}^{l} + \mathbf{h}_{v_5}^{l}\mathbf{W}^{l}) \\ &= \sigma(\mathbf{h}_{v_2}^{l}\mathbf{W}^{l} + \sum_{v_i \in \mathcal{N}(v_2)} \mathbf{h}_{v_i}^{l}\mathbf{W}^{l})\end{aligned}$$

Written in a matrix form (using H, I, A and some learnable parameters matrices W), it gives :

- Simplest way : $\mathbf{H}^{l+1} = \sigma(\mathbf{A}\mathbf{H}^{l}\mathbf{W}^{l})$
- Vanilla GNN : $\mathbf{H}^{l+1} = \sigma(\mathbf{C}\mathbf{H}^{l}\mathbf{W}^{l})$ with $\mathbf{C} = \mathbf{I} + \mathbf{A}$

Spatial GNNs

Message Passing Neural Networks : implementation tricks



$$\begin{aligned}\mathbf{h}_{v_2}^{l+1} &= \sigma(\mathbf{h}_{v_2}^{l}\,\mathbf{W}_0^{l} + \mathbf{h}_{v_1}^{l}\,\mathbf{W}_1^{l} + \mathbf{h}_{v_3}^{l}\,\mathbf{W}_1^{l} + \mathbf{h}_{v_4}^{l}\,\mathbf{W}_1^{l} + \mathbf{h}_{v_5}^{l}\,\mathbf{W}_1^{l}) \\ &= \sigma(\mathbf{h}_{v_2}^{l}\,\mathbf{W}_0^{l} + \sum_{v_i \in \mathcal{N}(v_2)} \mathbf{h}_{v_i}^{l}\,\mathbf{W}_1^{l})\end{aligned}$$

Written in a matrix form (using H, I, A and some learnable parameters matrices W), it gives :

- Simplest way : $\mathbf{H}^{l+1} = \sigma(\mathbf{A}\mathbf{H}^{l}\mathbf{W}^{l})$
- Vanilla GNN : $\mathbf{H}^{l+1} = \sigma(\mathbf{C}\mathbf{H}^{l}\mathbf{W}^{l})$ with $\mathbf{C} = \mathbf{I} + \mathbf{A}$
- General GNN : $\mathbf{H}^{l+1} = \sigma(\sum_k \mathbf{C}_k\mathbf{H}^{l}\mathbf{W}_k^{l})$ with $\mathbf{C}_1 = \mathbf{I}$ and $\mathbf{C}_2 = \mathbf{A}$

Spatial GNNs

Message Passing Neural Networks : old reference models

Some popular spatial GNN models :

- Graph Convolutional Network (GCN) [kipf2016semi] :

$$\mathbf{H}^{(l+1)} = \sigma((\widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) + \mathbb{1}\mathbf{b})$$

- Graph Attention Network (GAT) [velickovic2017graph] :

$$\mathbf{H}^{(l+1)} = \sigma((\mathbf{C}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) + \mathbf{1}\mathbf{b}) \text{ with } \mathbf{C}_{ij} = \text{softmax}_j(\sigma([\mathbf{h}_i^{(l)}\mathbf{W}^{(l)} || \mathbf{h}_j^{(l)}\mathbf{W}^{(l)}])\mathbf{a}$$

- Graph Isomorphism Network (GIN) [xu2018how] :

$$\mathbf{h}_u^{(l+1)} = \texttt{MLP}((1 + \epsilon^{(l)}).\mathbf{h}_u^{(l)} + \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(l)})$$

Outline of the lecture

Spectral GNNs

Key ideas

Idea : The Convolution theorem says - **"There is a one-to-one bijection between convolution in spatial domain and Hadamard (element-wise) multiplication in Fourier domain and vice-versa"**

Consequence : To define convolution in the graphs domain, an idea is to project the graph in the Fourier domain, do a simple element-wise multiplication in Fourier domain and apply inverse Fourier transform to get back to spatial domain.

Problem : How to define the Fourier Transform in the graph domain ?

Something useful you may ignore : the complex exponential used in the FT are the eigenfunctions of the Laplace operator

Spectral GNNs

Elements of Spectral Graph Theory

Tool : graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$ or Normalized version $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$

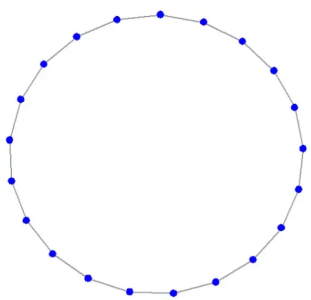
$\mathbf{L}$ is a real valued symetric matrix $\rightarrow$ it can be decomposed into :

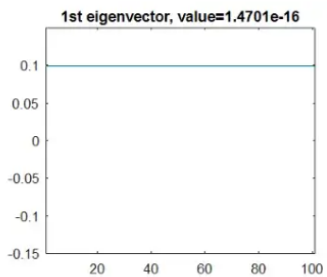
$$\mathbf{L} = \mathbf{U}.\mathrm{diag}(\boldsymbol{\lambda}).\mathbf{U}^T$$

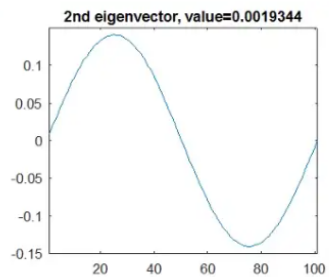
Side note : If u is one of the columns of U (i.e. an eigenvector of L). Then :

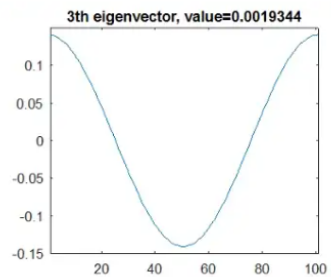
$$u^T L u = \lambda_u = \sum_{i<j,(i,j)\in E} (u(i) - u(j))^2$$

"Fun" fact : let's plot the eigenvectors of a simple "ring" graph :









Spectral GNNs

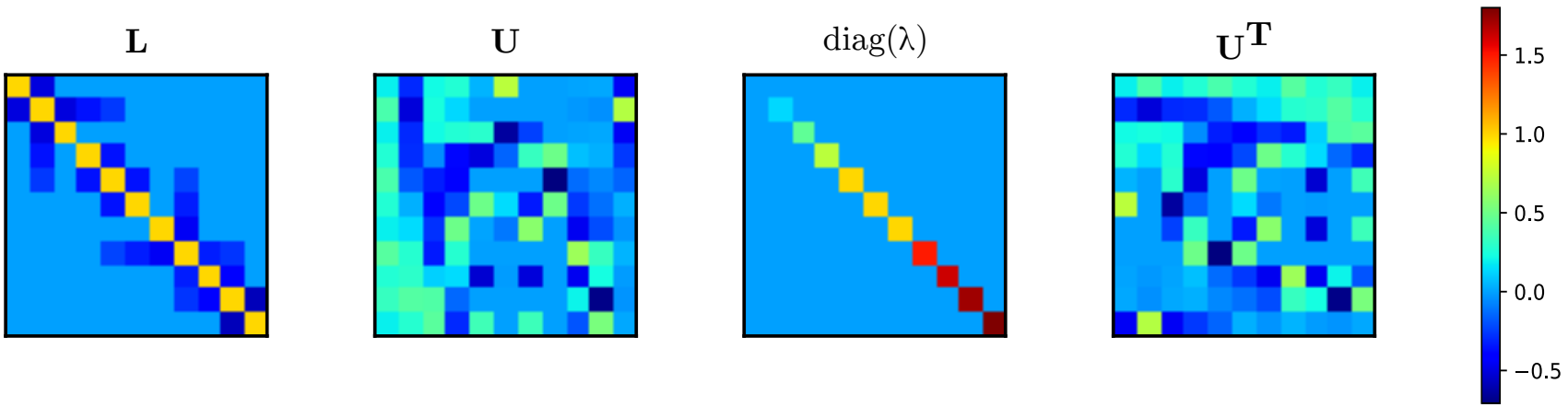
Elements of Spectral Graph Theory

- Idea of spectral GNN : compute a Fourier transform of the graph :
  Fourier transform : $\tilde{x} = \mathbf{U}^T x$. Inverse Fourier transform $x = \mathbf{U}\tilde{x}$
- Use the convolution theorem which says that a convolution can be implemented as a product in the Fourier Domain Spectral Filtering : $\mathbf{H}^{l+1} = \mathbf{U}.\text{diag}(F(\boldsymbol{\lambda})).\mathbf{U}^T\mathbf{H}^l$

Spectral GNNs

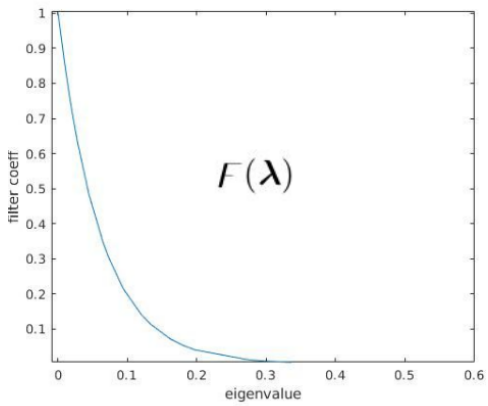
Elements of Spectral Graph Theory

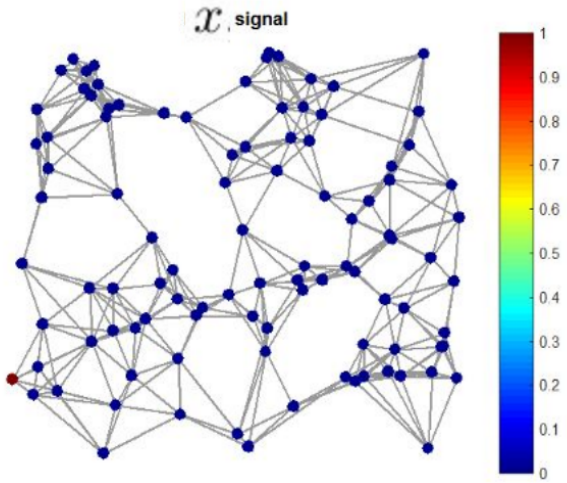
- Idea of spectral GNN : compute a Fourier transform of the graph :
  Fourier transform : $\tilde{x} = \mathbf{U}^T x$. Inverse Fourier transform $x = \mathbf{U}\tilde{x}$
- Use the convolution theorem which says that a convolution can be implemented as a product in the Fourier Domain Spectral Filtering : $\mathbf{H}^{l+1} = \mathbf{U}.\text{diag}(F(\boldsymbol{\lambda})).\mathbf{U}^T\mathbf{H}^l$

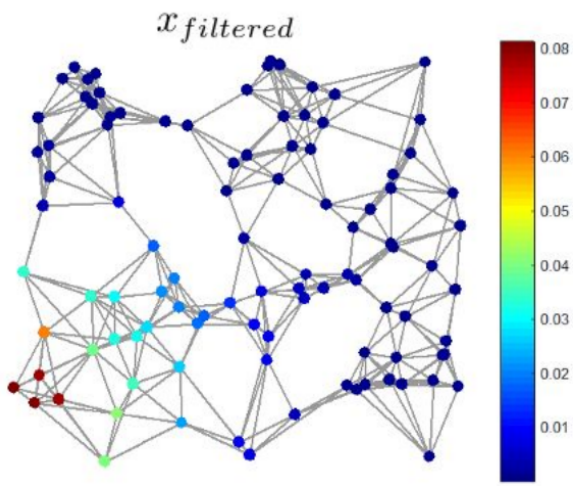


Spectral GNNs

Effect of filtering







Principle of non parametric spectral GNNs

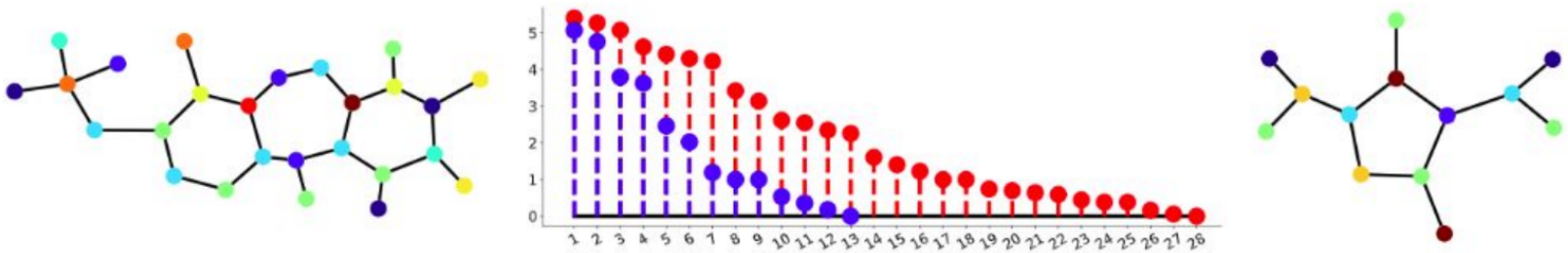
Learn the function $F(\boldsymbol{\lambda})$, i.e. the weights $F^{(l,j)} \in \mathbb{R}^{(n \times f_l)}$ used in :

$$H_j^{(l+1)} = \sigma\left(\sum_{i=1}^{f_l} U \operatorname{diag}(F_i^{(l,j)}) U^T H_i^{(l)}\right)$$

Spectral GNNs

Problem when learning $F(\boldsymbol{\lambda})$

- For multi-graph tasks : variable size of eigenvectors and variable eigenvalues



Solution : Parameterized functions of λ :

Idea : $F_i^{(l,j)} = B[W_{i,j}^{(l,1)}, \ldots, W_{i,j}^{(l,s_e)}]^T$ with $B \in \mathbb{R}^{n \times s_e}$ and $B_{i,j} = \varphi_j(\lambda_i)$

- polynomial : $\varphi_j(\lambda_i) = \lambda_i^j$
- Chebyshev $\rightarrow$ ChebNet
- Cayley $\rightarrow$ CayleyNet

Outline of the lecture

Bridging the Gap between Spatial and Spectral

Contribution of LITIS (ICLR 2021)

M. Balcilar shows that under specific conditions, both spatial and spectral approach can be expressed as :

$$H^{(l+1)} = \sigma(\sum_s C^{(s)} H^{(l)} W^{(l,s)})$$

where $C^{(s)} \in \mathbb{R}^{n \times n}$ is the s-th convolution support
This framework also enables to compare the frequency profiles of different models using

$$\Phi_s(\boldsymbol{\lambda}) = \text{diag}^{-1}(U^\top C^{(s)} U)$$

Bridging the Gap between Spatial and Spectral

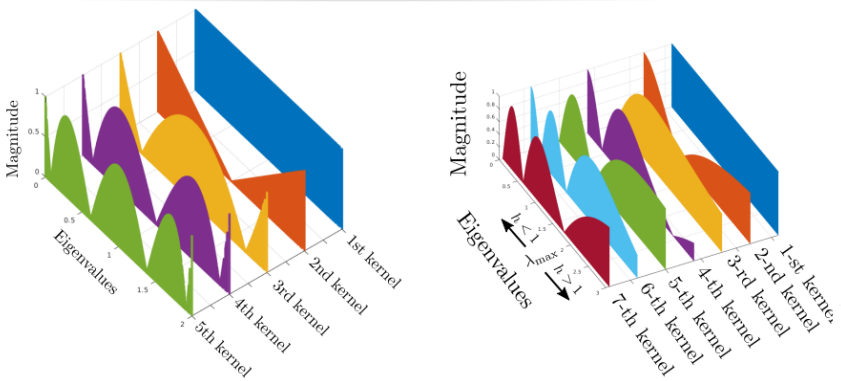
Values of C and $\Phi_s(\boldsymbol{\lambda})$ for different models

	Design	Support Type	Convolution Matrix	Frequency Response
MLP	Spectral	Fixed	$C = I$	$\Phi(\boldsymbol{\lambda}) = \mathbf{1}$
GCN	Spatial	Fixed	$C = \tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}$	$\Phi(\boldsymbol{\lambda}) \approx \mathbf{1} - \boldsymbol{\lambda}\overline{p}/(\overline{p}+1)$
GIN	Spatial	Trainable	$C = A + (1+\epsilon)I$	$\Phi(\boldsymbol{\lambda}) \approx \overline{p}\left(\frac{1+\epsilon}{\overline{p}} + \mathbf{1} - \boldsymbol{\lambda}\right)$
GAT	Spatial	Trainable	$C^{(s)}_{v,u} = e_{v,u} / \sum_{k\in\tilde{\mathcal{N}}(v)} e_{v,k}$	NA
			$C^{(1)} = I$	$\Phi_1(\boldsymbol{\lambda}) = \mathbf{1}$
CayleyNet	Spectral	Trainable	$C^{(2r)} = Re(\rho(hL)^r)$	$\Phi_{2r}(\boldsymbol{\lambda}) = \cos(r\theta(h\boldsymbol{\lambda}))$
			$C^{(2r+1)} = Re(\mathbf{i}\rho(hL)^r)$	$\Phi_{2r+1}(\boldsymbol{\lambda}) = -\sin(r\theta(h\boldsymbol{\lambda}))$
			$C^{(1)} = I$	$\Phi_1(\boldsymbol{\lambda}) = \mathbf{1}$
ChebNet	Spectral	Fixed	$C^{(2)} = 2L/\lambda_{\max} - I$	$\Phi_2(\boldsymbol{\lambda}) = 2\boldsymbol{\lambda}/\lambda_{\max} - \mathbf{1}$
			$C^{(s)} = 2C^{(2)}C^{(s-1)} - C^{(s-2)}$	$\Phi_s(\boldsymbol{\lambda}) = 2\Phi_2(\boldsymbol{\lambda})\Phi_{s-1}(\boldsymbol{\lambda}) - \Phi_{s-2}(\boldsymbol{\lambda})$

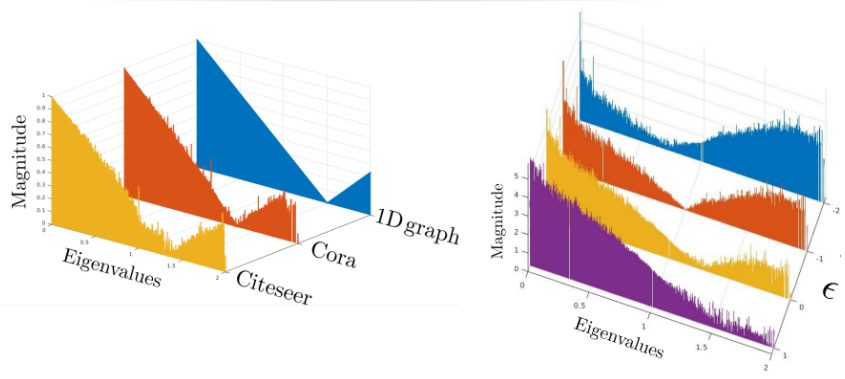
Bridging the Gap between Spatial and Spectral

Frequency response for different models

ChebNet and CayleyNet Freq. Res.



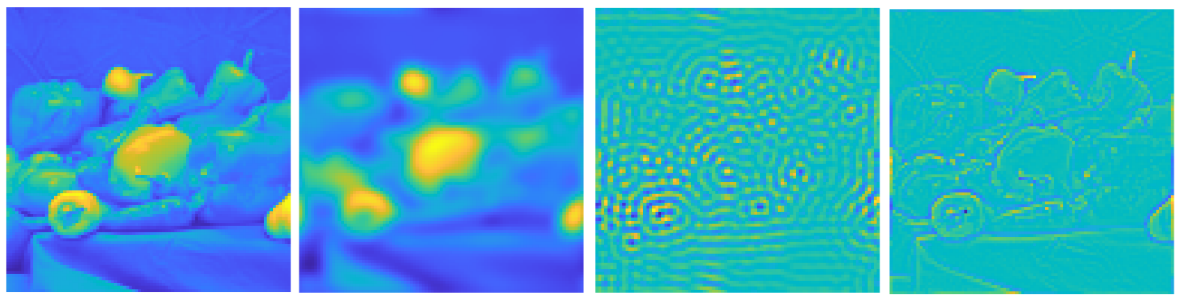
GCN and GIN Freq. Res.



Bridging the Gap between Spatial and Spectral

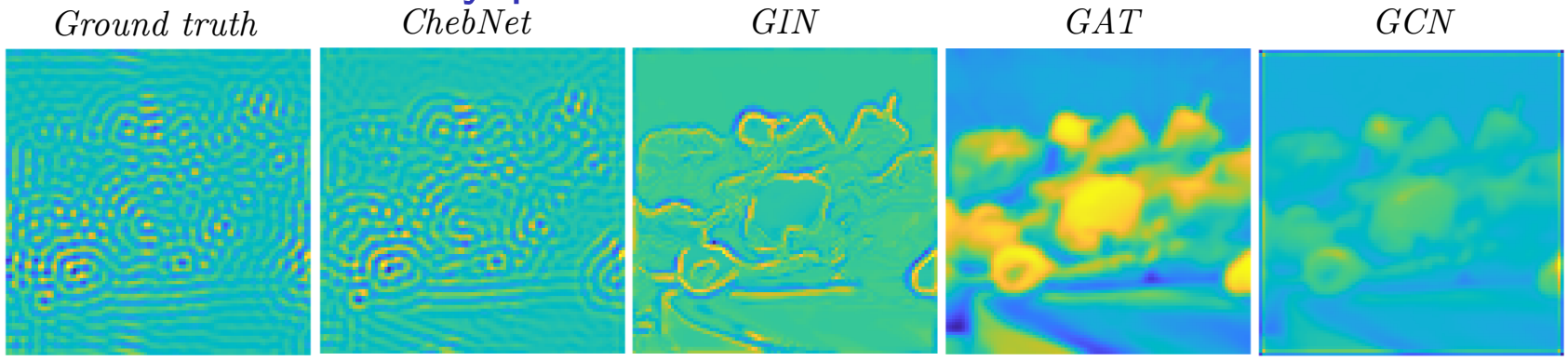
Illustration on a toy problem

- An input image is filtered with low, band and high pass filter
- Images are converted into a grid
- Inputs and outputs are the pixel intensity



Bridging the Gap between Spatial and Spectral

Illustration on a toy problem



Prediction Target	GCN	GIN	GAT	ChebNet
Low-pass filter (Φ_1)	15.55	11.01	10.50	3.44
Band-pass filter (Φ_2)	79.72	63.24	79.68	17.30
High-pass filter (Φ_3)	29.51	14.27	29.10	2.04

Results given in sum square error (Lower is better)

Conclusion about this introduction on GNNs

Advantages of Spatial GNNs

- Quick and simple
- Edge attributes can be included

Limitations of Spatial GNNs

- Are not able to produce bandpass filtering

Advantages of Spectral GNNs

- Able to produce bandpass filtering

Limitations of Spectral GNNs

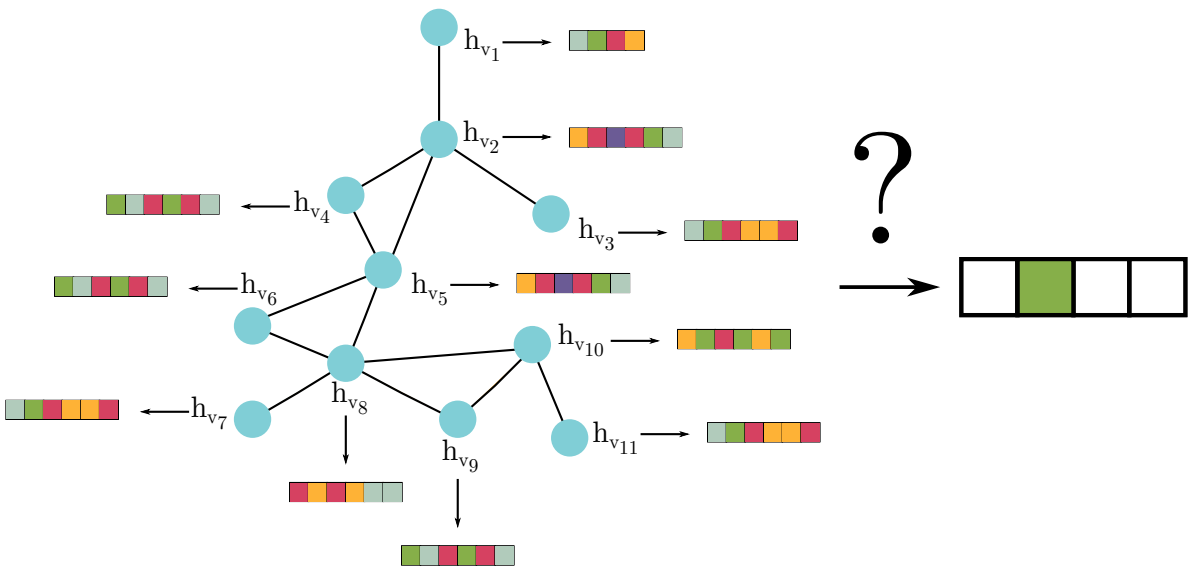
- Computation of the eigen decomposition is heavy
- Edge attribute can not be included
- Transferability problem of filter learned of eigenvalues

Outline of the lecture

Graph Level Prediction

The problem

- GNNs compute nodes embedding
- Need to reduce to a fixed size vector for graph level prediction
- 3 different strategies have emerged



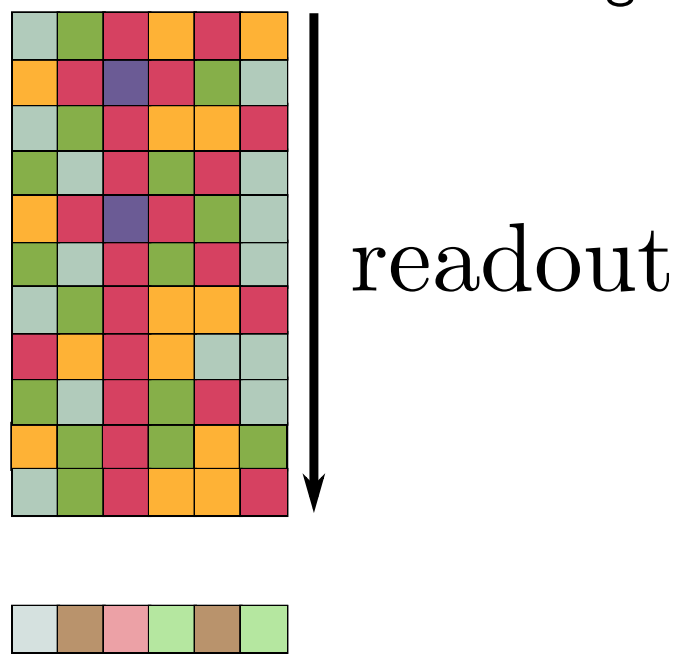
Graph Level Prediction

Readout functions

Compute a fixed size vector as a function of the node embedding

Some examples of readout functions :

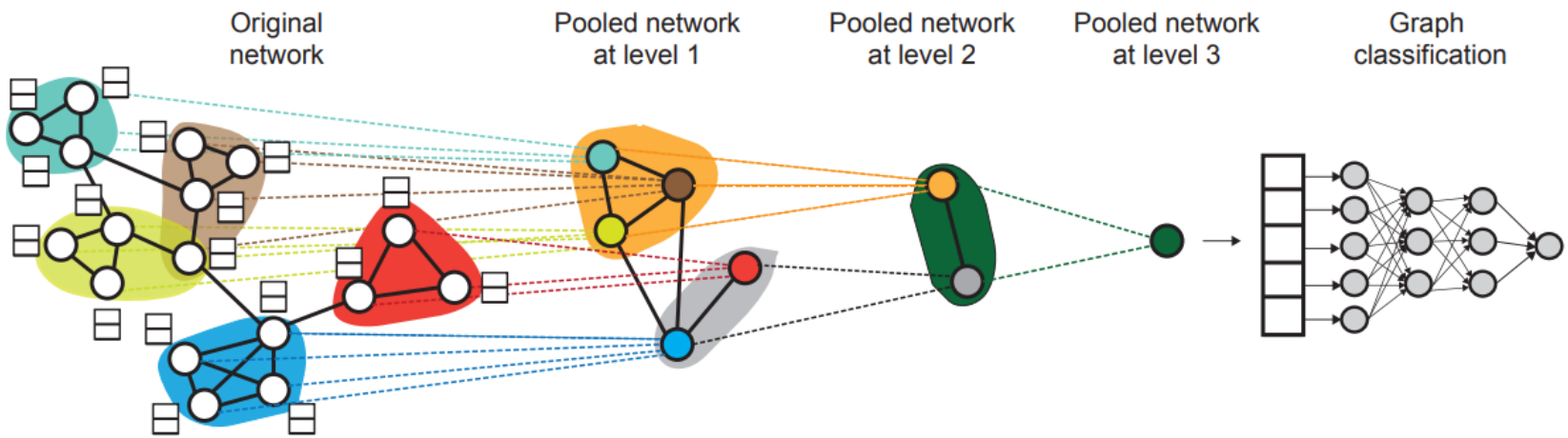
- `mean`
- `max`
- `sum`
- `weighted sum`
- `covariance`



Graph Level Prediction

Graph Pooling

- Global pooling methods : pool each node embedding into one single node (similar to readout functions)
- Hierarchical pooling methods : compute intermediate graphs which are passed in the following layer.



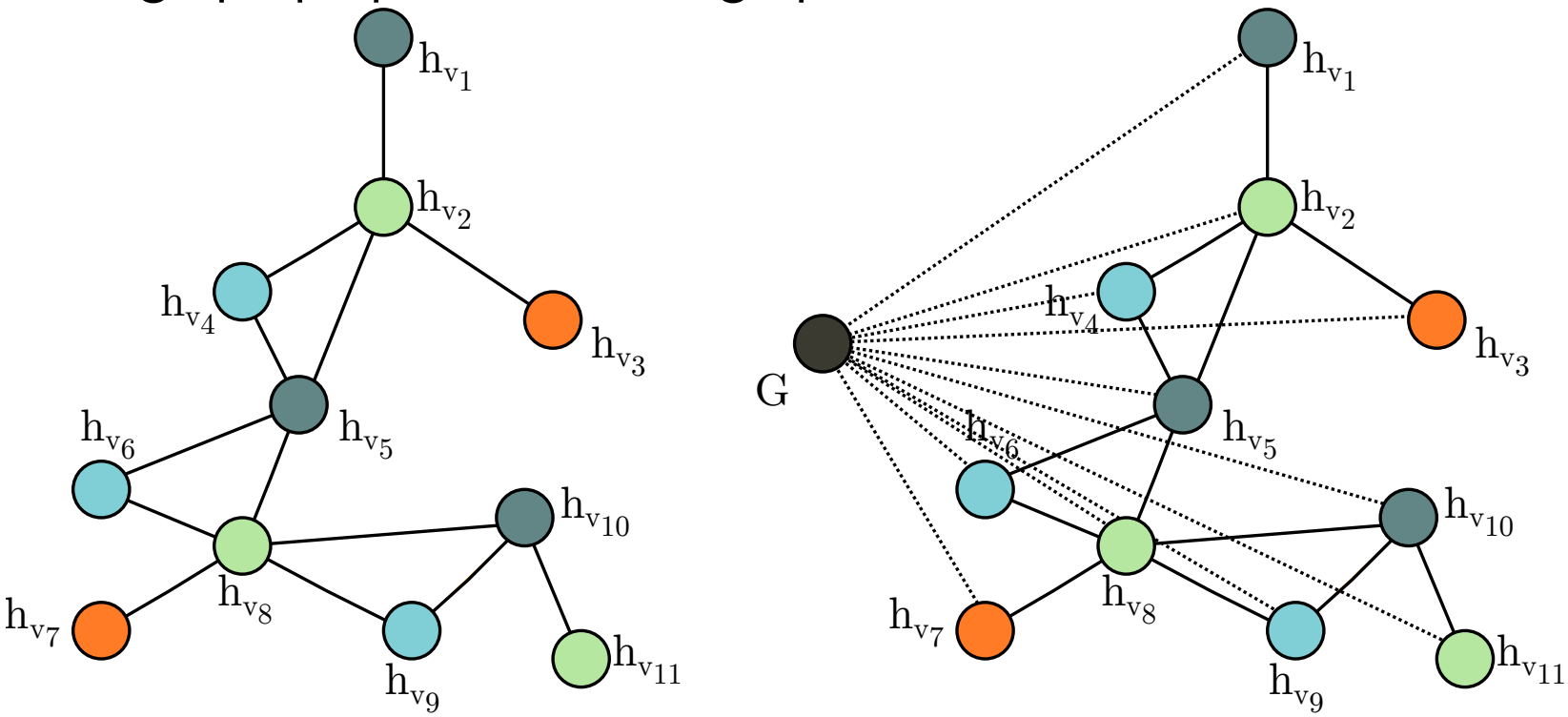
[ying2018hierarchical]

Graph Level Prediction

Fully connected node

Add a fully connected node corresponding to the graph himself

Predict graph properties on the graph's node

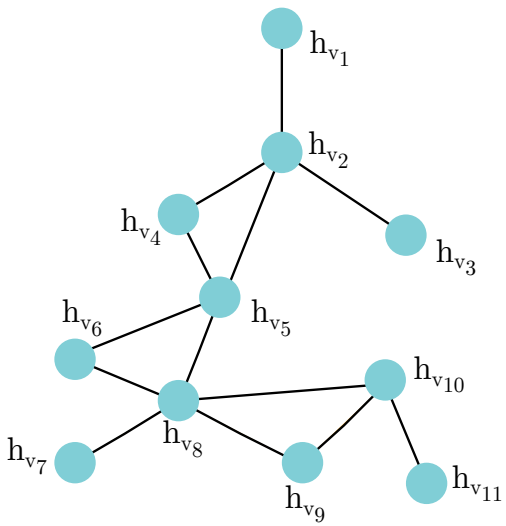


Outline of the lecture

"Hot" research questions about GNNs ?

Problem statement : Under-Reaching

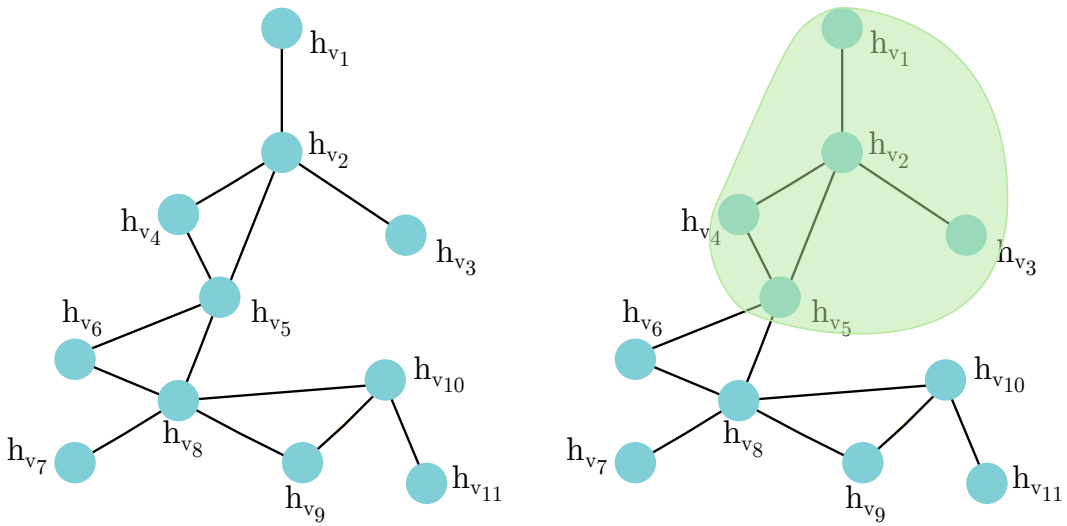
- The context used to update each node (receptive fields) at layer l depends on the number of previous layer.
- To allow a node to receive information from other node at a radius of K, GNNs needs to have at least K layers.
- Problems that rely on long range interaction require as many GNN layers as the range of the interaction.



"Hot" research questions about GNNs ?

Problem statement : Under-Reaching

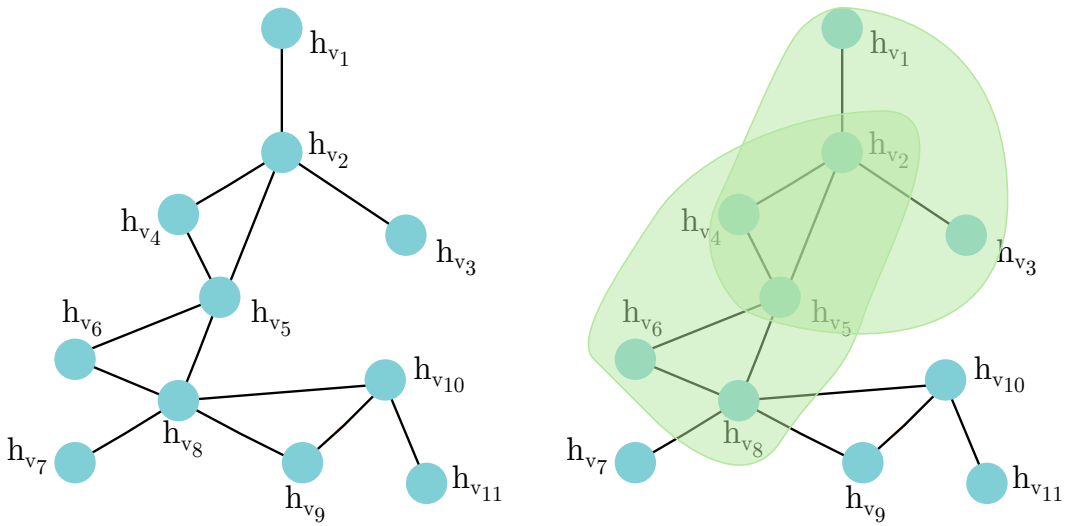
- The context used to update each node (receptive fields) at layer l depends on the number of previous layer.
- To allow a node to receive information from other node at a radius of K, GNNs needs to have at least K layers.
- Problems that rely on long range interaction require as many GNN layers as the range of the interaction.



"Hot" research questions about GNNs ?

Problem statement : Under-Reaching

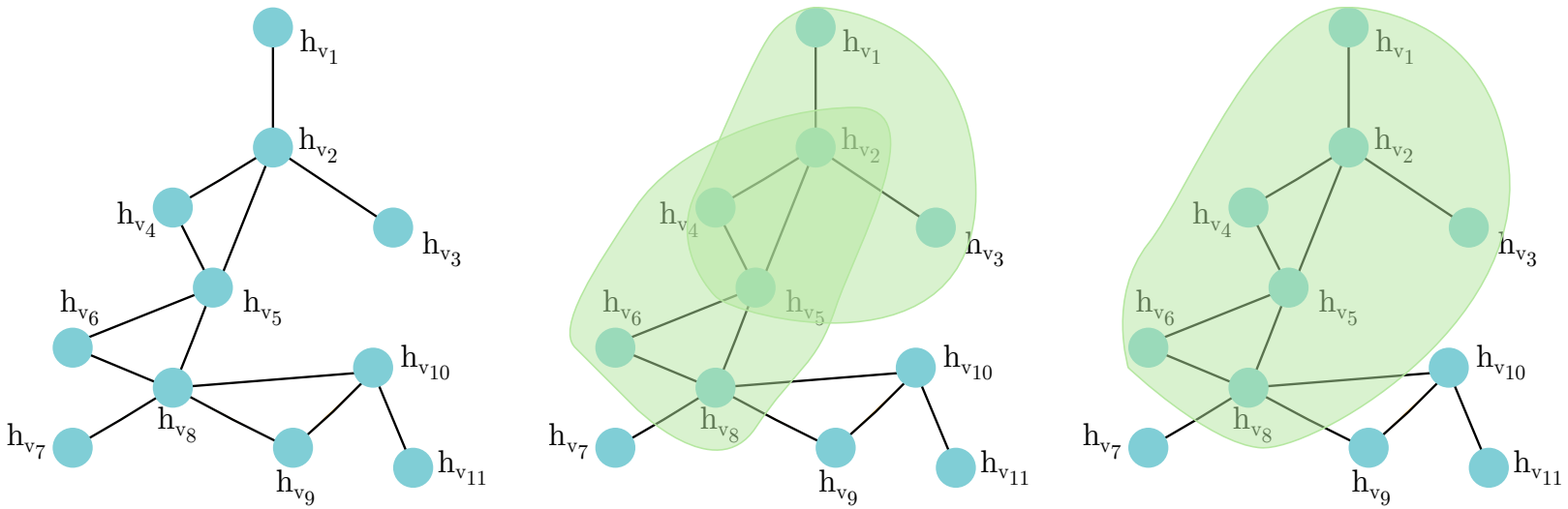
- The context used to update each node (receptive fields) at layer l depends on the number of previous layer.
- To allow a node to receive information from other node at a radius of K, GNNs needs to have at least K layers.
- Problems that rely on long range interaction require as many GNN layers as the range of the interaction.



"Hot" research questions about GNNs ?

Problem statement : Under-Reaching

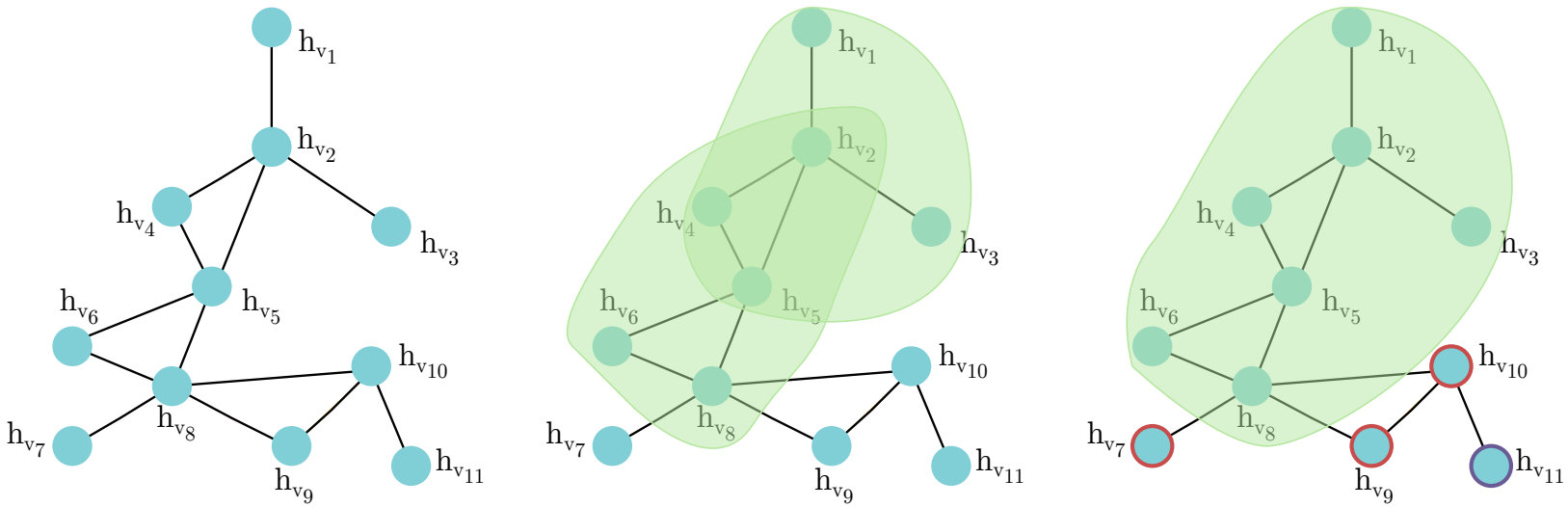
- The context used to update each node (receptive fields) at layer l depends on the number of previous layer.
- To allow a node to receive information from other node at a radius of K, GNNs needs to have at least K layers.
- Problems that rely on long range interaction require as many GNN layers as the range of the interaction.



"Hot" research questions about GNNs ?

Problem statement : Under-Reaching

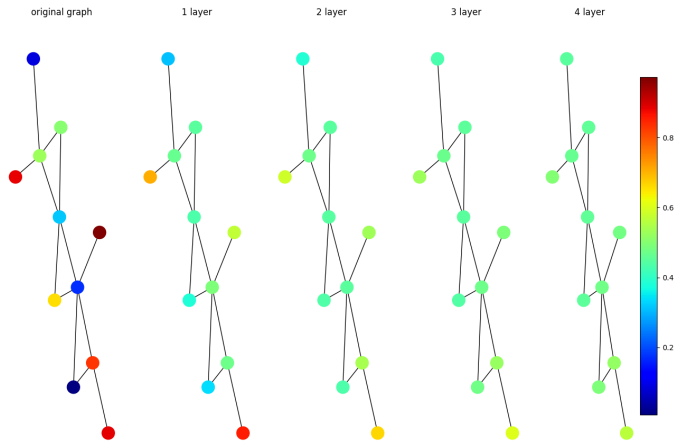
- The context used to update each node (receptive fields) at layer l depends on the number of previous layer.
- To allow a node to receive information from other node at a radius of K, GNNs needs to have at least K layers.
- Problems that rely on long range interaction require as many GNN layers as the range of the interaction.



"Hot" research questions about GNNs ?

Problem statement : Over-Smoothing

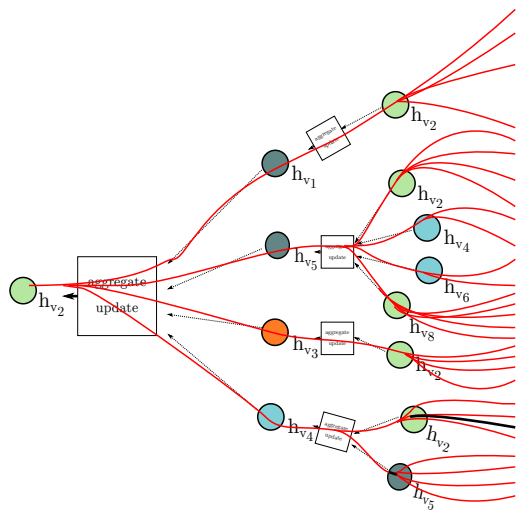
- Most GNNs are known to be limited to low-pass filtering.
- The effects of stacking low pass filters over images are well-known.
- The same effects appears with graphs.



"Hot" research questions about GNNs ?

Problem statement : Over-Squashing

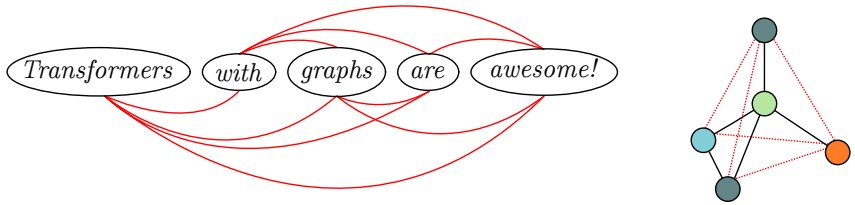
- Dimensionality of computed nodes hidden states at layer K has to be big enough to capture nodes information at radius K
- Receptive fields grows exponentially, while GNNs compress information into fixed-size vectors.



"Hot" research questions about GNNs ?

A recent solution to these problems

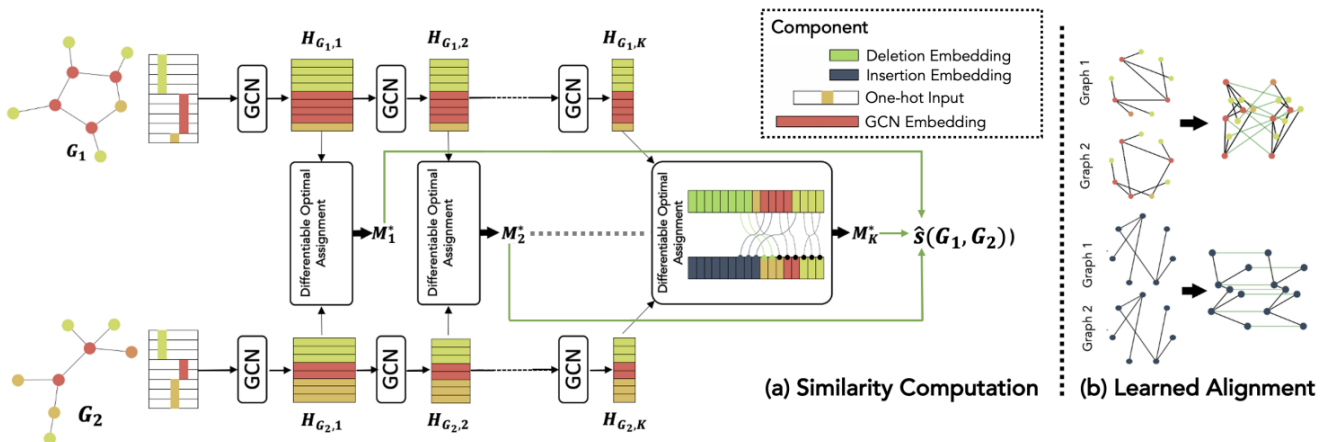
- Previous limitations mostly come from the relation between receptive fields and the number of layers
- One way to increase the receptive fields would be through fully connected graphs
- Many recent works study the use of transformers with graph neural network with different ideas (e.g. positionnal encoding)



"Hot" research questions about GNNs ?

How using GNN's for learning to compute graph metrics

- We saw that for computing GED, a key problem is to include the graph structure in the node similarities
- Can GNNs be useful for this task ?
- Aldo will talk you about this exciting subject next week !



"Hot" research questions about GNNs ?

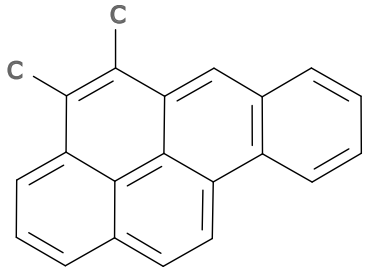
Problem statement : how to qualify GNNs expressive power ?

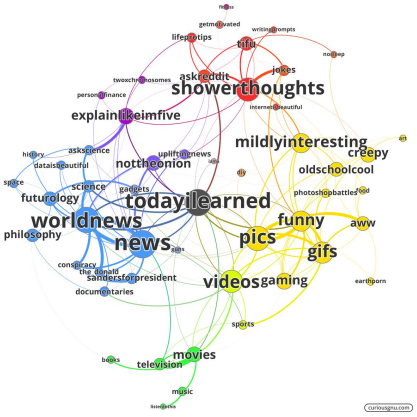
- MLP are known to be universal approximator
- What are the representation properties of GNNs ?
- Two ways are explored for characterizing "structural" expressive power :
  - distinguishing non-isomorphic graphs (relation with WL test)
  - counting substructures.
- For the "signal" aspect, spectral properties are used
- Jason will talk about these fascinating aspects next week

Outline of the lecture

A word about reference datasets

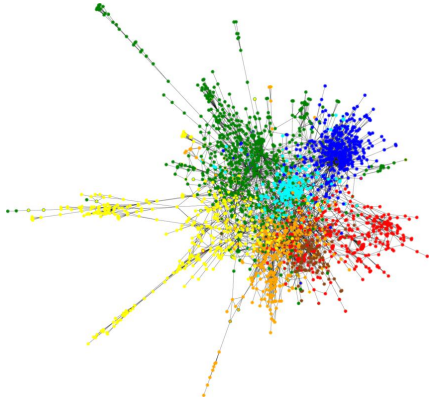
- **Graph Classification/Regression**
  Molecules dataset : QM9, TOX, HIV.



- 

  Inductive Node Classification/Regression
  Reddit, PPI

- **Transductive Node Classification/Regression**
  Citation Dataset : Cora, Citeseer, PubMed



A word about reference datasets

Outline of the lecture

Few recent application examples

At the node level : the protein folding problem

Few recent application examples

At the node level : the protein folding problem

- Protein = 3D-structured arrangement of a amino-acid sequence

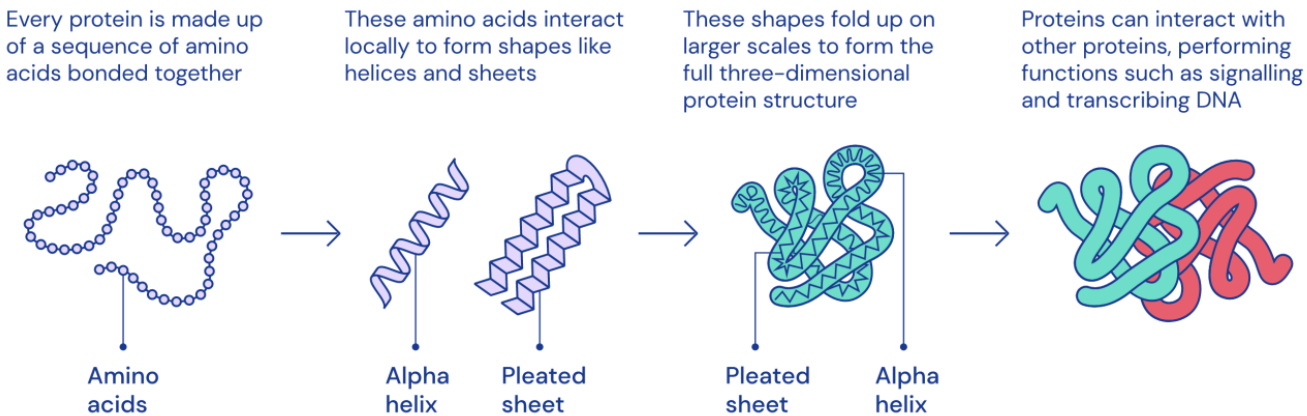
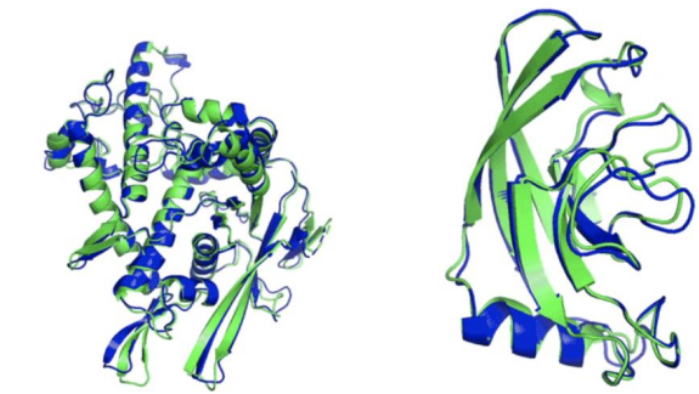


Image credit: DeepMind

Few recent application examples

At the node level : the protein folding problem

- Protein = 3D-structured arrangement of a amino-acid sequence
- Machine learning task : predict the 3D structure from the sequence



Few recent application examples

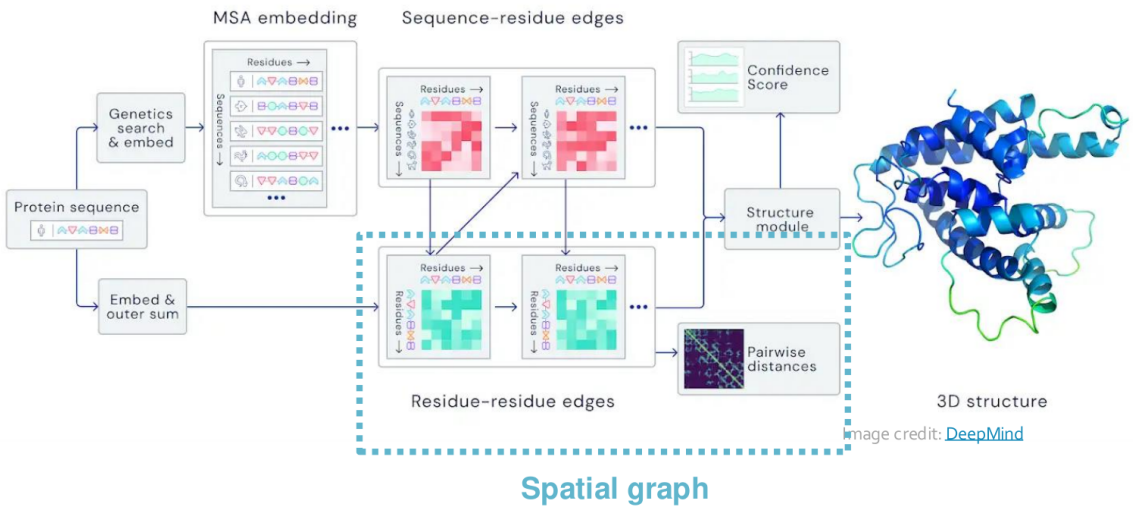
At the node level : the protein folding problem

- Protein = 3D-structured arrangement of a amino-acid sequence
- Machine learning task : predict the 3D structure from the sequence
- Very recent result : the AlphaFold model [DeepMind, 2020]

Few recent application examples

At the node level : the protein folding problem

- Protein = 3D-structured arrangement of a amino-acid sequence
- Machine learning task : predict the 3D structure from the sequence
- Very recent result : the AlphaFold model [DeepMind, 2020]
- Key idea : spatial graph + GNN for predicting the position of amino-acid



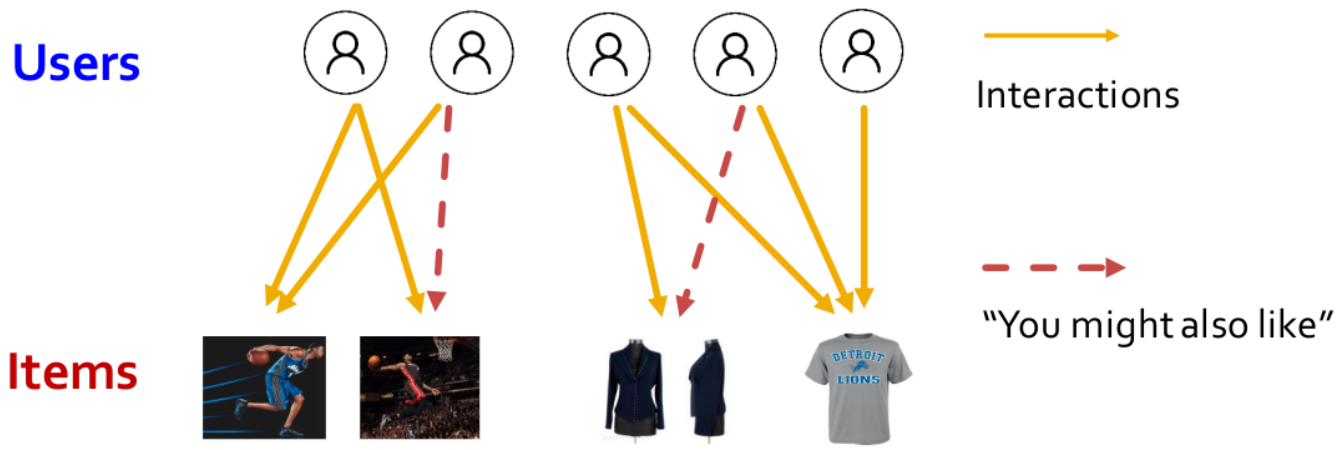
Few recent application examples

At the edge level :

Few recent application examples

At the edge level :

- Recommender systems :
  - nodes represent user and items, edges give user-item interactions
  - Machine learning task : predict most likely links

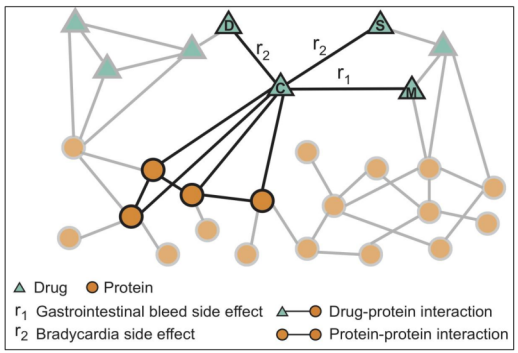


Few recent application examples

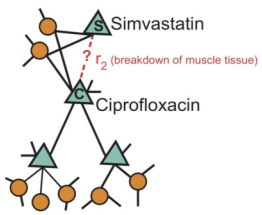
At the edge level :

- Recommender systems :
  - nodes represent user and items, edges give user-item interactions
  - Machine learning task : predict most likely links
- Biomedical link prediction :
  - nodes represent drugs and proteins, edges give interactions
  - Machine learning task : given a pair of drugs, predict adverse side effects

- **Nodes**: Drugs & Proteins
- **Edges**: Interactions



Query: How likely will Simvastatin and Ciprofloxacin, when taken together, break down muscle tissue?



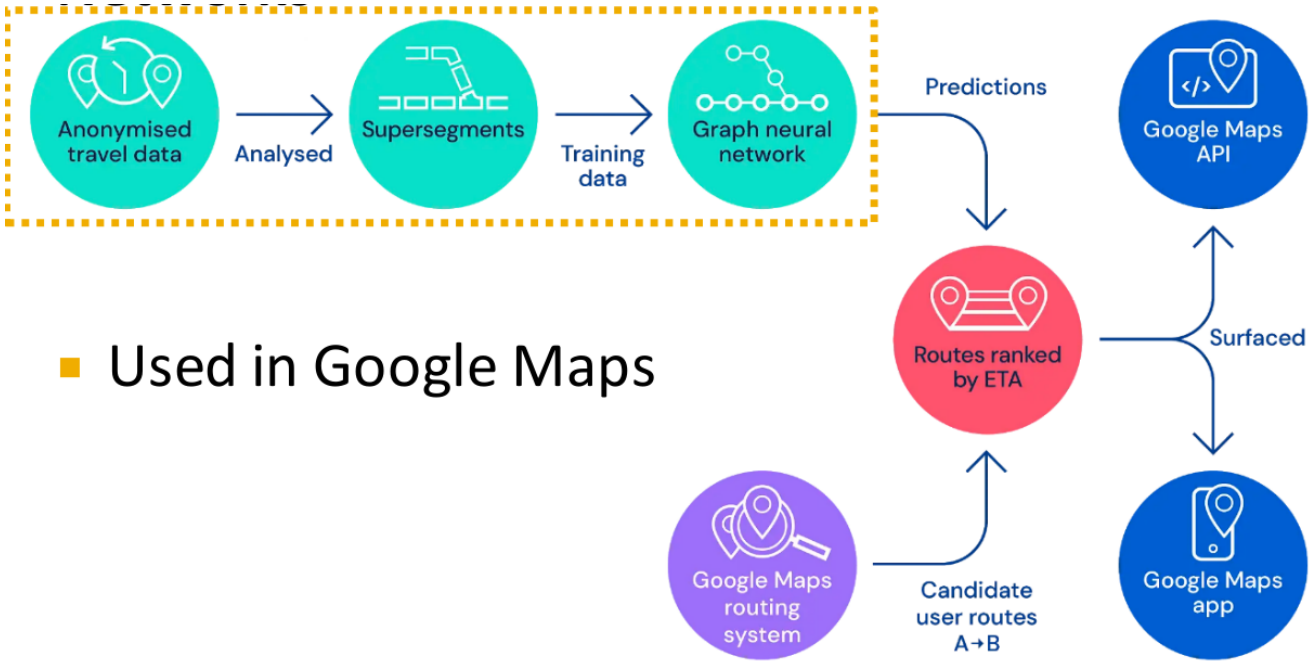
Few recent application examples

At the subgraph level : estimated time of arrival (ETA)

Few recent application examples

At the subgraph level : estimated time of arrival (ETA)

- Nodes = Road segments
- Edges = Connectivity between road segments
- Machine learning task : predict the time of arrival
- Key idea : Use of a graph neural network



- Used in Google Maps

THE MODEL ARCHITECTURE FOR DETERMINING OPTIMAL ROUTES AND THEIR TRAVEL TIME.

Image credit: DeepMind

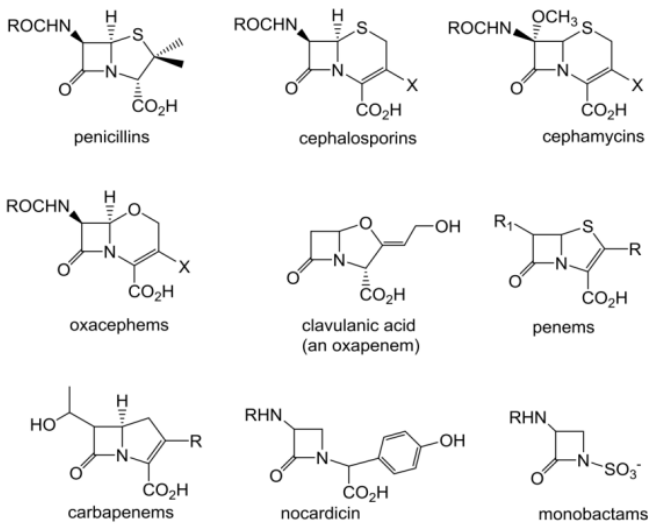
Few recent application examples

At the graph level : Antibiotic discovery

Few recent application examples

At the graph level : Antibiotic discovery

- Nodes = Atoms
- Edges = Chemical bonds
- Machine learning task : predict promising molecules from a pool of candidates



Konaklieva, Monika I. "Molecular targets of β-lactam-based antimicrobials: beyond the usual suspects." Antibiotics 3.2 (2014): 128-142.

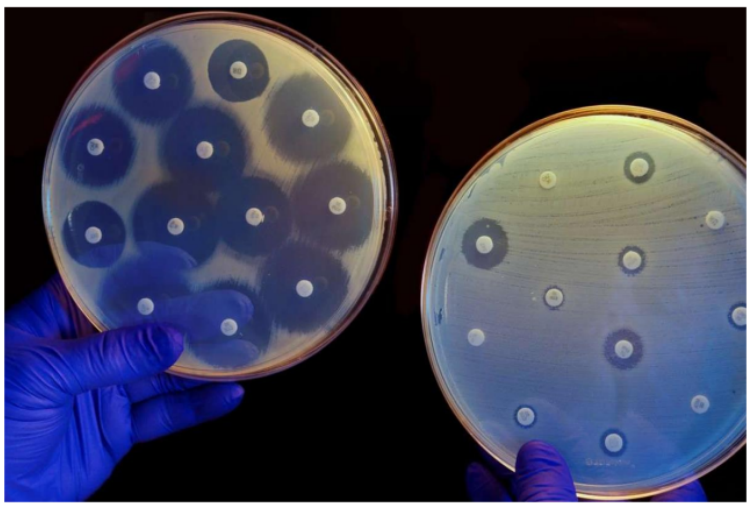
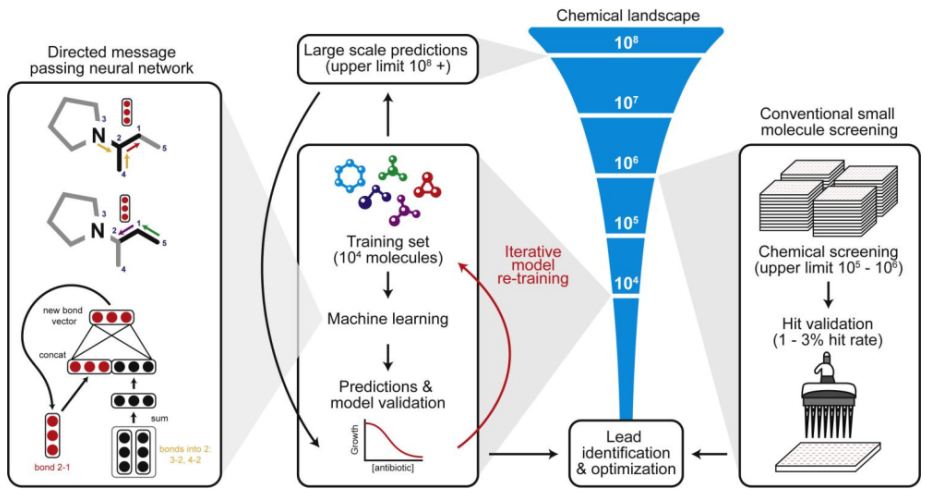


Image credit: CNN

Few recent application examples

At the graph level : Antibiotic discovery

- Nodes = Atoms
- Edges = Chemical bonds
- Machine learning task : predict promising molecules from a pool of candidates
- Key idea : Use of a graph neural network



Stokes, Jonathan M., et al. "A deep learning approach to antibiotic discovery." Cell 180.4 (2020): 688-702.

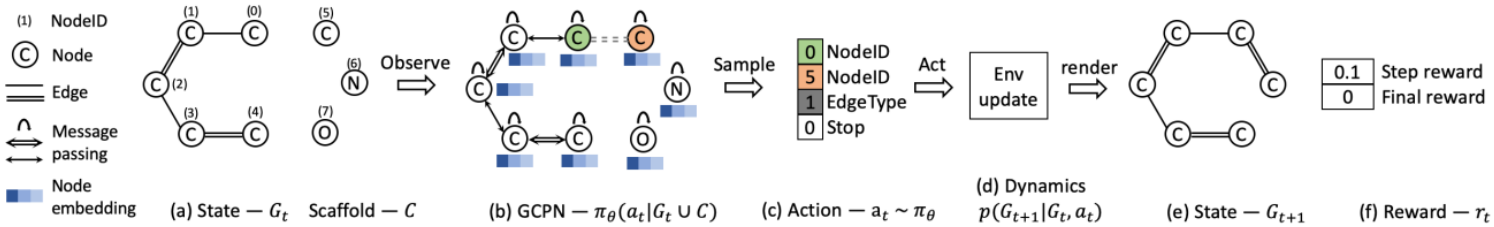
Few recent application examples

Case of a graph generation task

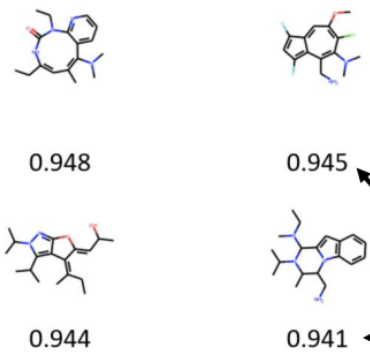
Few recent application examples

Case of a graph generation task

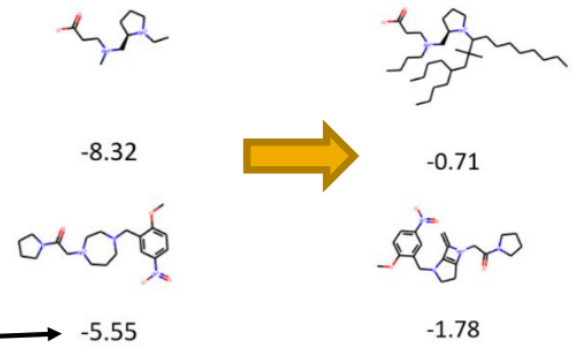
- Nodes = Atoms
- Edges = Chemical bonds
- Machine learning task : Generate or optimize molecules by combining GNN and RL



Use case 1: Generate novel molecules with high Drug likeness value



Use case 2: Optimize existing molecules to have desirable properties



Drug likeness

Few recent application examples

Case of dynamic graphs

Few recent application examples

Case of dynamic graphs

- Nodes = Particles
- Edges = Interaction between particles
- Machine learning task : Predict how a graph will evolve