

# Machine learning for graphs and with graphs

## Graph neural networks

Titouan Vayer & Pierre Borgnat  
email: [titouan.vayer@inria.fr](mailto:titouan.vayer@inria.fr), [pierre.borgnat@ens-lyon.fr](mailto:pierre.borgnat@ens-lyon.fr)

October 24, 2023



# Table of contents

---

From neural networks...

The basic ideas

Logistic regression and one layer neural-network

Convolutional neural networks

... to unsupervised node embeddings techniques...

A chronological start

... to graph neural networks

Learning with graphs

What is a GNN ?

A bit of group theory

Invariance and equivariance

Message-passing neural networks

Examples of GNN

The whole pipeline

Expressivity of GNN

Conclusion

# Table of contents

---

From neural networks...

The basic ideas

Logistic regression and one layer neural-network

Convolutional neural networks

... to unsupervised node embeddings techniques...

A chronological start

... to graph neural networks

Learning with graphs

What is a GNN ?

A bit of group theory

Invariance and equivariance

Message-passing neural networks

Examples of GNN

The whole pipeline

Expressivity of GNN

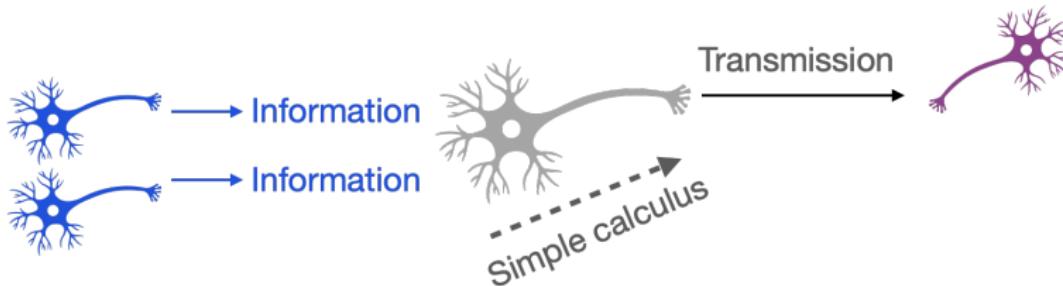
Conclusion

# What is a neural network ?

---

Neural network is a certain family of functions **parametrized by weights**.

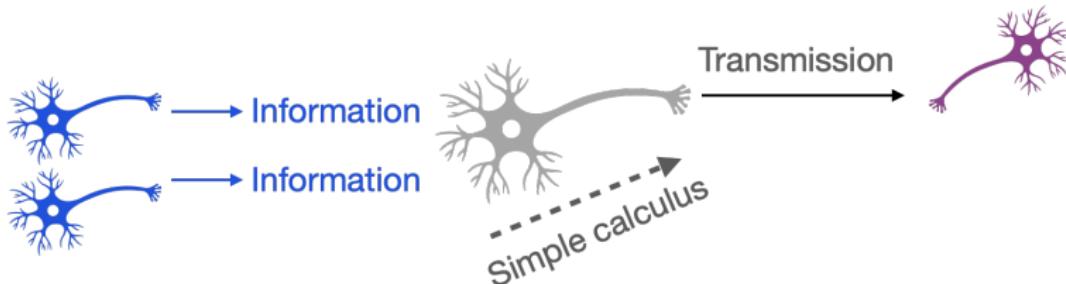
Built upon a biological analogy Rosenblatt 1958



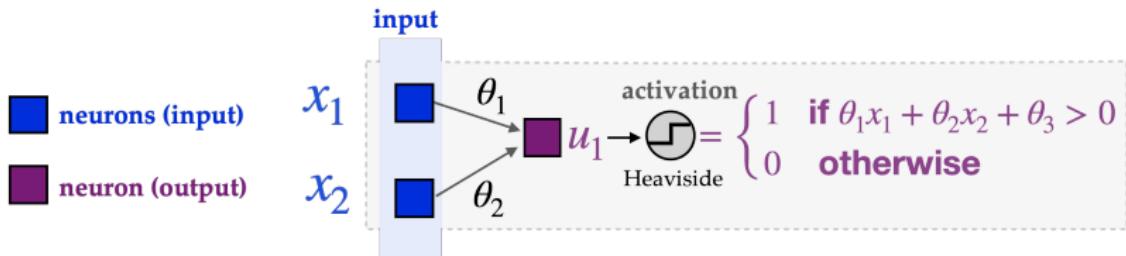
# What is a neural network ?

Neural network is a certain family of functions **parametrized by weights**.

Built upon a biological analogy Rosenblatt 1958



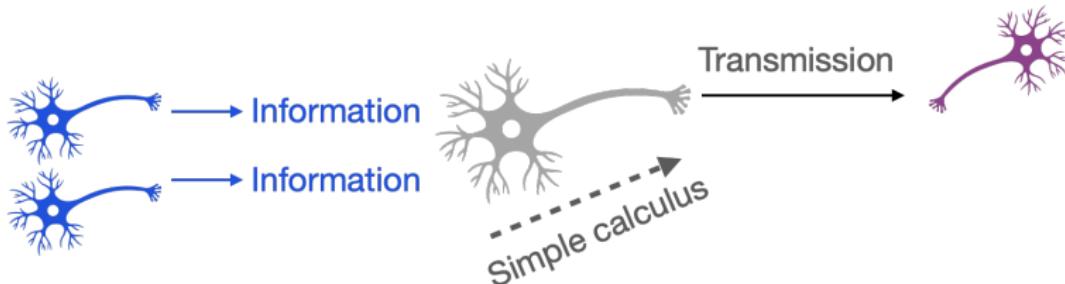
- ▶ First example  $f(\mathbf{x} = (x_1, x_2)) = \text{activation}(\theta_1 x_1 + \theta_2 x_2 + \theta_3)$ :



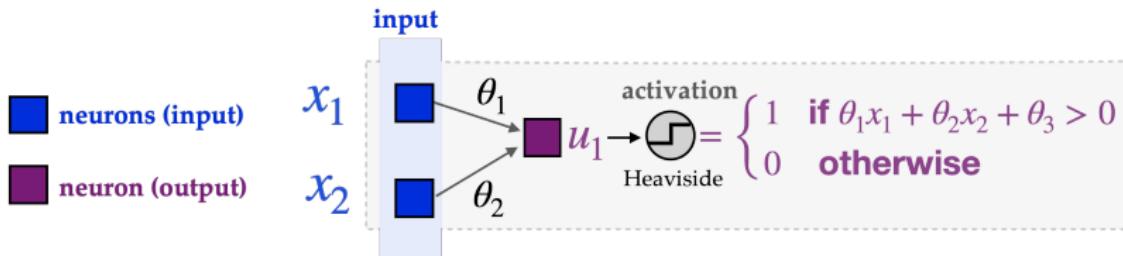
# What is a neural network ?

Neural network is a certain family of functions **parametrized by weights**.

Built upon a biological analogy Rosenblatt 1958



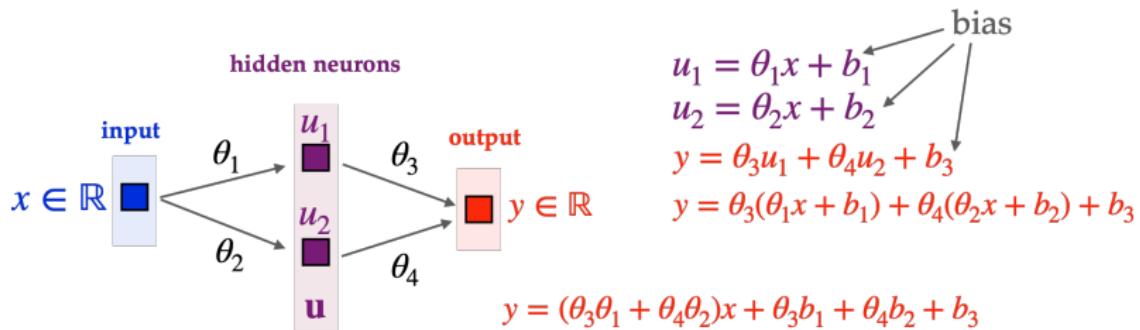
- ▶ Second example  $f(\mathbf{x} = (x_1, x_2)) = \text{activation}(\theta_1 x_1 + \theta_2 x_2 + \theta_3)$ :



# What is a neural network ?

## Feed-forward neural networks

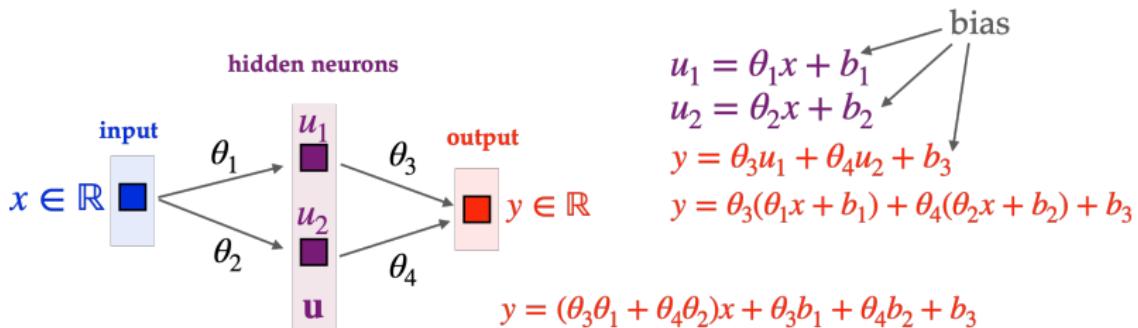
- Linear neural network:



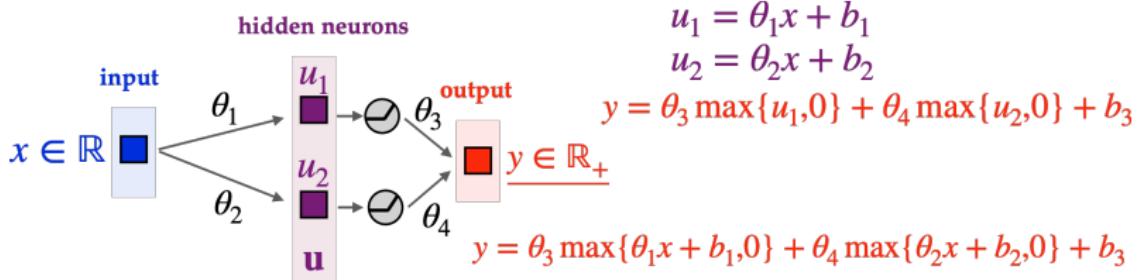
# What is a neural network ?

## Feed-forward neural networks

- ▶ Linear neural network:



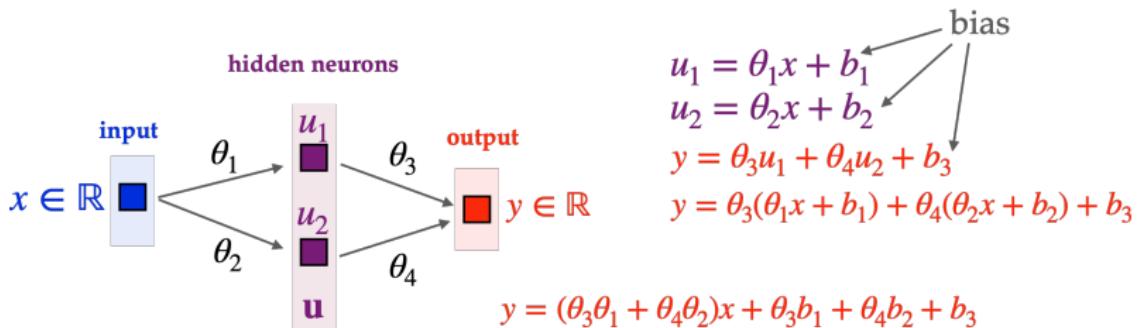
- ▶ Non-linearity:



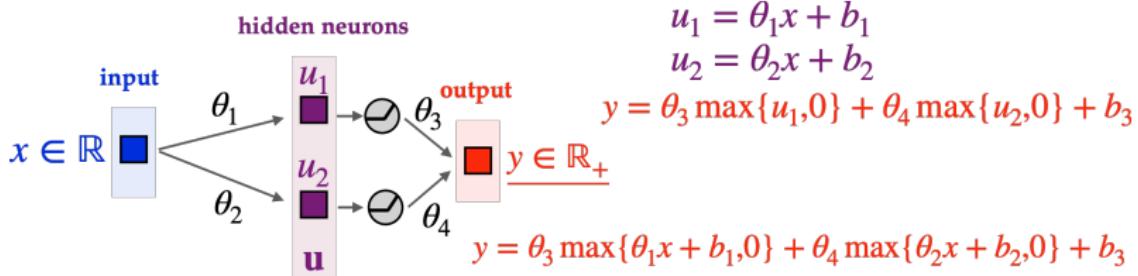
# What is a neural network ?

## Feed-forward neural networks

- ▶ Linear neural network:



- ▶ Non-linearity:

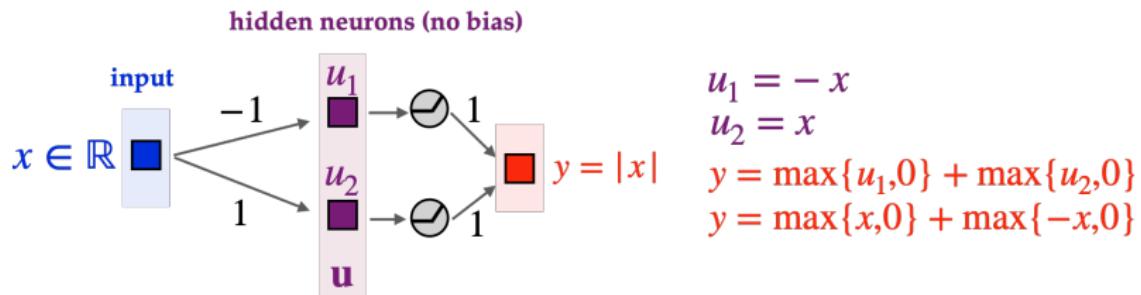


- ▶ Find a neural network that implements the function  $f(x) = |x|$ .

# What is a neural network ?

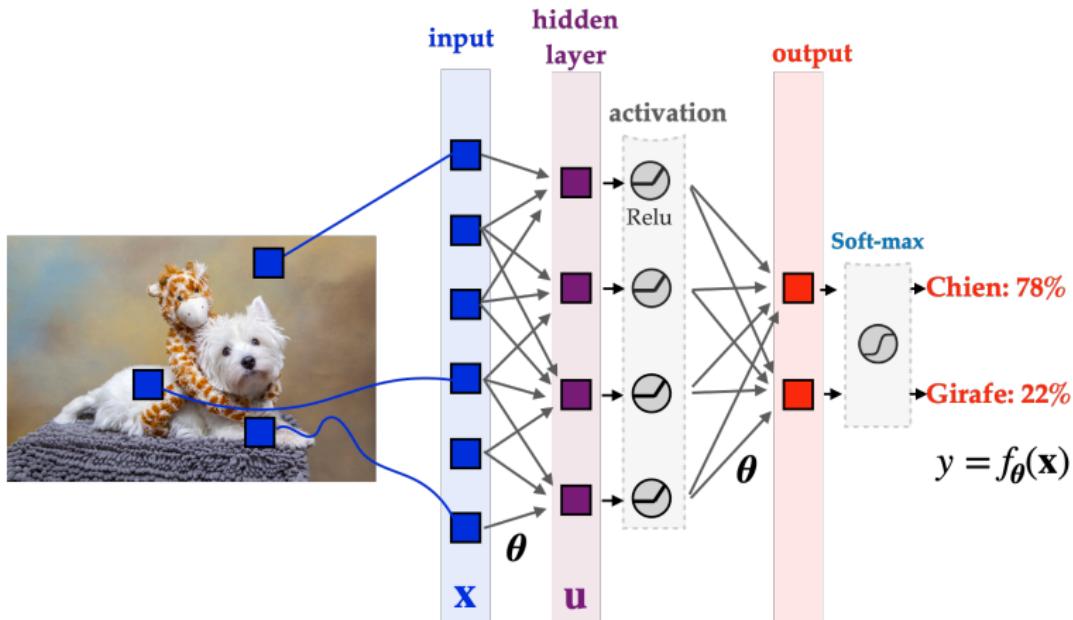
## Feed-forward neural networks

- ▶ Find a neural network that implements the function  $f(x) = |x|$



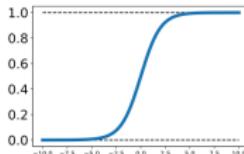
# What is a neural network ?

## Feed-forward neural networks



**Soft-max:**

$$\text{softmax}[u]_i = \frac{\exp(u_i)}{\sum_j \exp(u_j)}$$



# What is a neural network ?

---

## Feed-forward neural networks

- ▶ Feed-forward NN are function of the form

$$f(\mathbf{x}) = T_K \circ \sigma_{K-1} \circ \cdots \circ \sigma_1 \circ T_1(\mathbf{x})$$

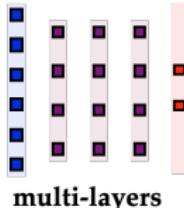
$$\text{where } T_k(\mathbf{z}) = \mathbf{W}^{(k)}\mathbf{z} + \mathbf{b}^{(k)}$$

and  $\sigma_k$  pointwise activation function.

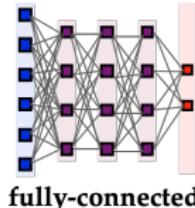
- ▶ All the weights:  $\theta = (\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(K)}, \mathbf{b}^{(1)}, \dots, \mathbf{b}^{(K)})$ .
- ▶ Depending on the task the output of a NN is also transformed  $g(\mathbf{x}) = \text{norm}(f(\mathbf{x}))$ .
- ▶ E.g.  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  and  $g : \mathbb{R}^d \rightarrow (0, 1)$  for binary classification with  $\text{norm}(u) = 1/(1 + \exp(-u))$  (logistic/sigmoid function).

# What is a neural network ?

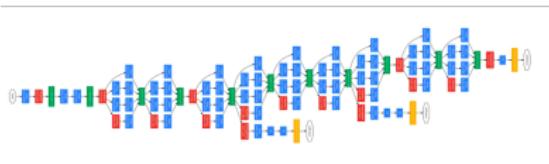
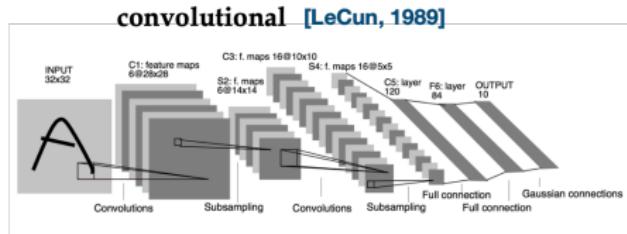
## A zoo of architectures



multi-layers



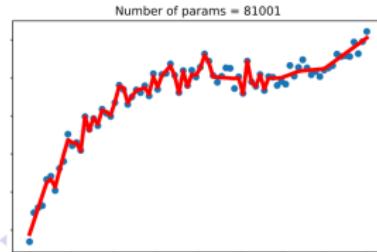
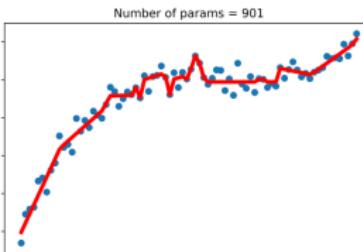
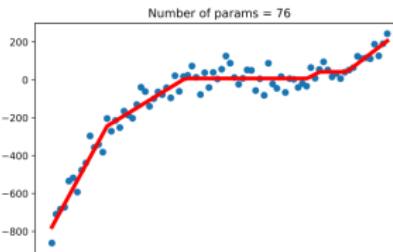
fully-connected



deep-learning

also: generative, recurrent, transformers, attention layer transformers...

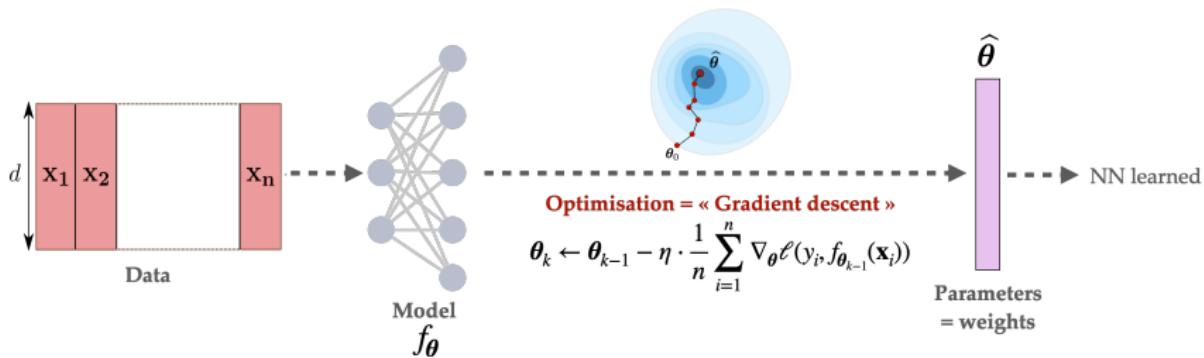
Richness of neural network



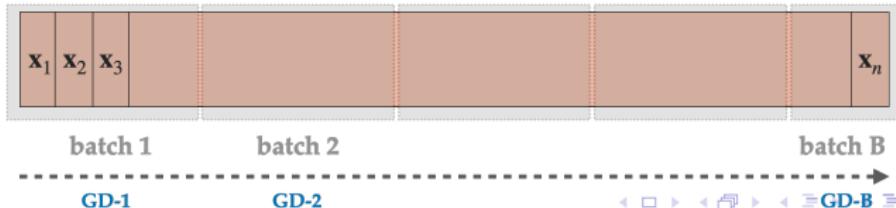
# Neural network in practice

## The (very) big picture

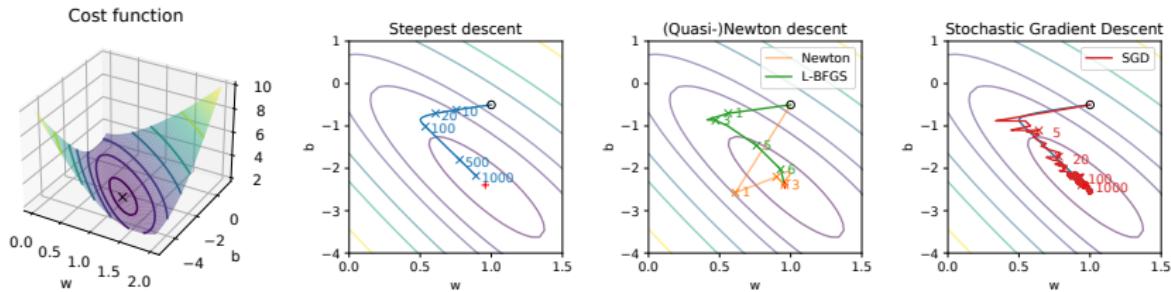
Find the weights that minimizes the empirical minimization loss.



- ▶ In practice gradient descent very slow.
- ▶ We use stochastic gradient descents (and variations) on batches of the data.



# (almost) All optimization in one slide



## Principle

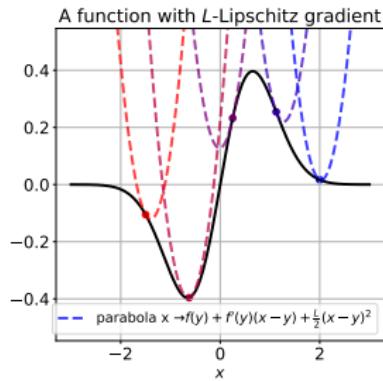
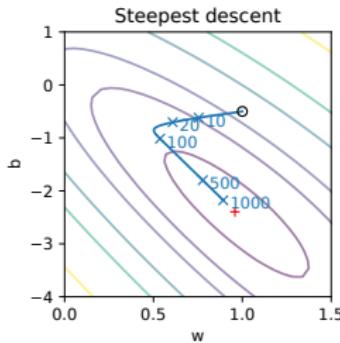
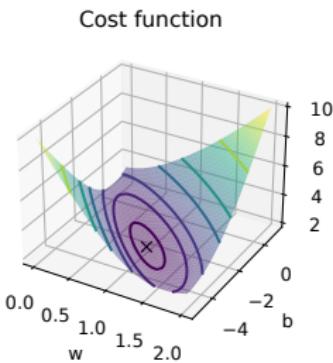
- ▶ Minimize a smooth function  $J(\theta)$  using its gradient (or  $\approx$ ).
- ▶ Initialize a vector  $\theta^{(0)}$  and update it at each iteration  $k$  as:

$$\theta^{(k+1)} = \theta^{(k)} + \mu_k \mathbf{d}_k$$

where  $\mu_k$  is a step and  $\mathbf{d}_k$  is a descent direction  $\mathbf{d}_k^\top \nabla J(\theta^{(k)}) < 0$ .

- ▶ Classical descent directions are :
  - ▶ **Steepest descent:**  $\mathbf{d}_k = -\nabla J(\theta^{(k)})$  (a.k.a. Gradient descent).
  - ▶ **(Quasi) Newton:**  $\mathbf{d}_k = -(\nabla^2 J(\theta^{(k)}))^{-1} \nabla J(\theta^{(k)})$ ,  $\nabla^2 J$  is the Hessian.
  - ▶ **Stochastic Gradient Descent :**  $\mathbf{d}_k = -\tilde{\nabla} J(\theta^{(k)})$  with approx. gradient.
- ▶ For NN: gradient computed with **automatic differentiation** (TD).

# (almost) All optimization in two slides...



Why is this a good idea ? (on the board)

Let  $J : \mathbb{R}^D \rightarrow \mathbb{R}$  with  $L$ -Lipschitz gradient<sup>1</sup> and  $J^* := \min_{\theta} J(\theta) > -\infty$ . Then, provided that  $0 < \mu_k < \frac{2}{L}$ , the iterations  $\theta^{(k+1)} = \theta^{(k)} - \mu_k \nabla J(\theta^{(k)})$  satisfy

$J(\theta^{(k+1)}) < J(\theta^{(k)})$  (decrease the objective function)

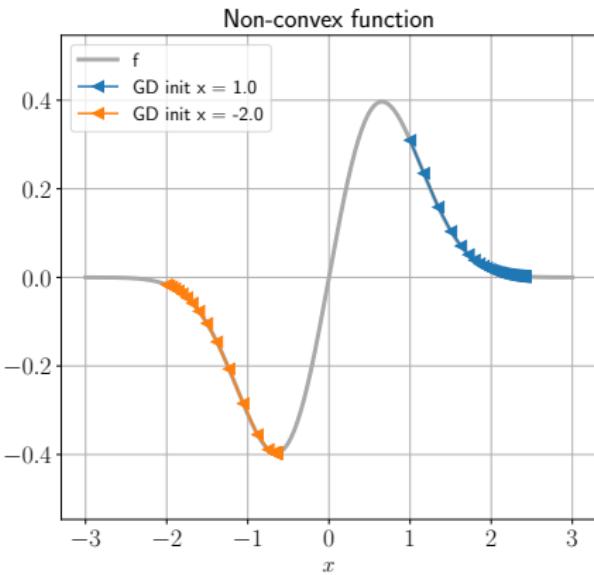
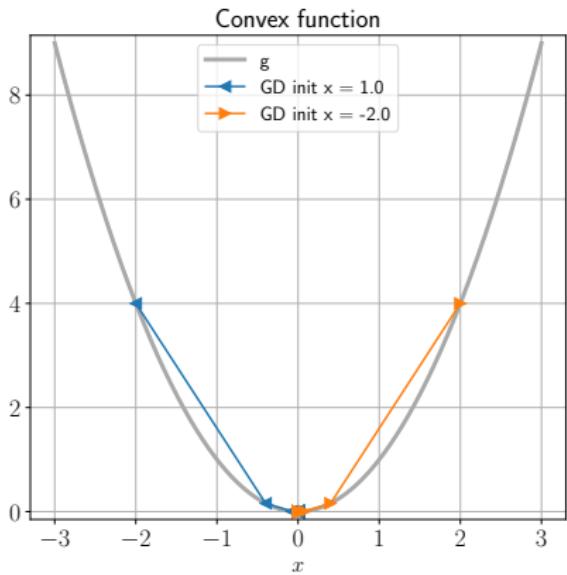
$\lim_{k \rightarrow +\infty} \nabla J(\theta^{(k)}) = \mathbf{0}$  (critical point)

<sup>1</sup>it means that  $\forall \theta_1, \theta_2 \in \mathbb{R}^d$ ,  $\|\nabla J(\theta_1) - \nabla J(\theta_2)\|_2 \leq L\|\theta_1 - \theta_2\|_2$ .

# (almost) All optimization in three slides...

## Be aware of local minima

- ▶ When the functions are not convex, GD and its variants can fall into bad local minima.
- ▶ **Neural networks are not convex w.r.t. the optimized parameters !**



# Table of contents

---

From neural networks...

The basic ideas

**Logistic regression and one layer neural-network**

Convolutional neural networks

... to unsupervised node embeddings techniques...

A chronological start

... to graph neural networks

Learning with graphs

What is a GNN ?

A bit of group theory

Invariance and equivariance

Message-passing neural networks

Examples of GNN

The whole pipeline

Expressivity of GNN

Conclusion

# First simple neural network: logistic regression

---

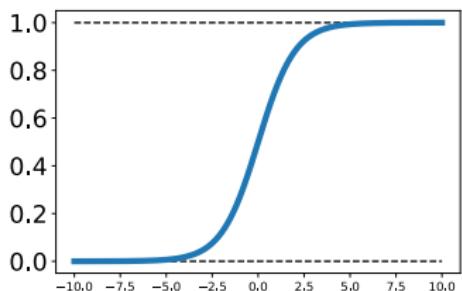
- ▶ It is a **classification method**: input  $(\mathbf{x}_i)_i \in \mathbb{R}^d$  and  $(y_i)_i \in \{+1, -1\}$ .
- ▶ **Probabilistic model**: find a model  $h_\theta$  s.t.  $\mathbb{P}(y = +1|\mathbf{x}) \approx h_\theta(\mathbf{x})$ .
- ▶ Bayes decision:  $f(\mathbf{x}) = \text{sign}(\mathbb{P}(y = +1|\mathbf{x}) - \mathbb{P}(y = -1|\mathbf{x})) \in \{-1, +1\}$ .

# First simple neural network: logistic regression

- ▶ It is a **classification method**: input  $(\mathbf{x}_i)_i \in \mathbb{R}^d$  and  $(y_i)_i \in \{+1, -1\}$ .
- ▶ **Probabilistic model**: find a model  $h_\theta$  s.t.  $\mathbb{P}(y = +1|\mathbf{x}) \approx h_\theta(\mathbf{x})$ .
- ▶ Bayes decision:  $f(\mathbf{x}) = \text{sign}(\mathbb{P}(y = +1|\mathbf{x}) - \mathbb{P}(y = -1|\mathbf{x})) \in \{-1, +1\}$ .

## The sigmoid function

$$\sigma(z) = 1/(1 + \exp(-z)).$$



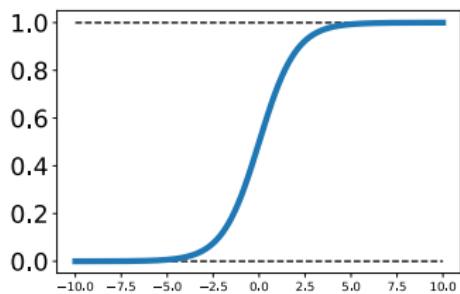
- ▶ Usually used to model probabilities.

# First simple neural network: logistic regression

- ▶ It is a **classification method**: input  $(\mathbf{x}_i)_i \in \mathbb{R}^d$  and  $(y_i)_i \in \{+1, -1\}$ .
- ▶ **Probabilistic model**: find a model  $h_\theta$  s.t.  $\mathbb{P}(y = +1|\mathbf{x}) \approx h_\theta(\mathbf{x})$ .
- ▶ Bayes decision:  $f(\mathbf{x}) = \text{sign}(\mathbb{P}(y = +1|\mathbf{x}) - \mathbb{P}(y = -1|\mathbf{x})) \in \{-1, +1\}$ .

## The sigmoid function

$$\sigma(z) = 1/(1 + \exp(-z)).$$

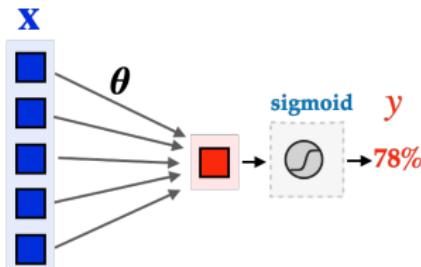


- ▶ Usually used to model probabilities.

## The logistic regression model

The model is  $\mathbb{P}(y = +1|\mathbf{x}) = \sigma(\boldsymbol{\theta}^\top \mathbf{x} + b)$ .

- ▶  $\boldsymbol{\theta} \in \mathbb{R}^d$  are weights,  $b \in \mathbb{R}$  is a bias that are to be optimized.
- ▶ It is a **generalized linear model**.
- ▶ Is also a one layer neural-network (no hidden layer).



# First simple neural network: logistic regression

---

One property

$$\mathbb{P}(y = -1 | \mathbf{x}) = 1 - \mathbb{P}(y = 1 | \mathbf{x}) = 1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x} + b) = \sigma(-( \boldsymbol{\theta}^\top \mathbf{x} + b))$$

# First simple neural network: logistic regression

---

One property

$$\mathbb{P}(y = -1 | \mathbf{x}) = 1 - \mathbb{P}(y = 1 | \mathbf{x}) = 1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x} + b) = \sigma(-( \boldsymbol{\theta}^\top \mathbf{x} + b))$$

Maximum likelihood estimation

Find  $\boldsymbol{\theta} \in \mathbb{R}^d, b \in \mathbb{R}$  that maximize the (conditional) log-likelihood (**board**)

$$\begin{aligned} & \sum_{i:y_i=1} \log \mathbb{P}(y_i = 1 | \mathbf{x}_i) + \sum_{i:y_i=-1} \log \mathbb{P}(y_i = -1 | \mathbf{x}_i) \\ &= \sum_{i:y_i=1} \log \sigma(\boldsymbol{\theta}^\top \mathbf{x}_i + b) + \sum_{i:y_i=-1} \log \sigma(-( \boldsymbol{\theta}^\top \mathbf{x}_i + b)) \\ &= \sum_{i=1}^n \log \sigma(y_i(\boldsymbol{\theta}^\top \mathbf{x}_i + b)). \end{aligned}$$

# First simple neural network: logistic regression

One property

$$\mathbb{P}(y = -1 | \mathbf{x}) = 1 - \mathbb{P}(y = 1 | \mathbf{x}) = 1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x} + b) = \sigma(-(\boldsymbol{\theta}^\top \mathbf{x} + b))$$

Maximum likelihood estimation

Find  $\boldsymbol{\theta} \in \mathbb{R}^d$ ,  $b \in \mathbb{R}$  that maximize the (conditional) log-likelihood (board)

$$\begin{aligned} & \sum_{i:y_i=1} \log \mathbb{P}(y_i = 1 | \mathbf{x}_i) + \sum_{i:y_i=-1} \log \mathbb{P}(y_i = -1 | \mathbf{x}_i) \\ &= \sum_{i:y_i=1} \log \sigma(\boldsymbol{\theta}^\top \mathbf{x}_i + b) + \sum_{i:y_i=-1} \log \sigma(-(\boldsymbol{\theta}^\top \mathbf{x}_i + b)) \\ &= \sum_{i=1}^n \log \sigma(y_i(\boldsymbol{\theta}^\top \mathbf{x}_i + b)). \end{aligned}$$

Minimizing the logistic loss

$$\min_{\boldsymbol{\theta}, b} \sum_{i=1}^n \log \left[ 1 + \exp \left( -y_i(\boldsymbol{\theta}^\top \mathbf{x}_i + b) \right) \right].$$

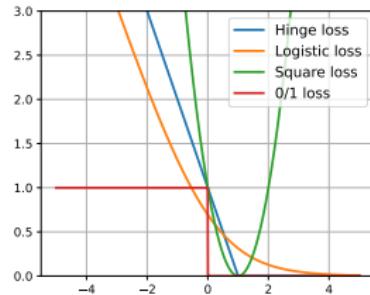
- Convex problem, can be solved with (Quasi) Newton's method.

# First simple neural network: logistic regression

## Remember your losses

With  $f : \mathbb{R}^d \rightarrow \mathbb{R}$ , many losses can be written as  $\ell(y_i, f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i))$  with  $\Phi \downarrow$ .

- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \mathbf{1}_{y_i f(\mathbf{x}_i) \leq 0}$ .
- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \max\{0, 1 - y_i f(\mathbf{x}_i)\}$ .
- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \log(1 + e^{-y_i f(\mathbf{x}_i)})$ .

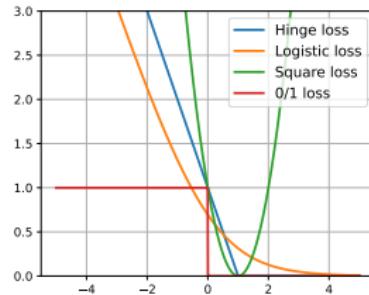


# First simple neural network: logistic regression

## Remember your losses

With  $f : \mathbb{R}^d \rightarrow \mathbb{R}$ , many losses can be written as  $\ell(y_i, f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i))$  with  $\Phi \downarrow$ .

- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \mathbf{1}_{y_i f(\mathbf{x}_i) \leq 0}$ .
- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \max\{0, 1 - y_i f(\mathbf{x}_i)\}$ .
- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \log(1 + e^{-y_i f(\mathbf{x}_i)})$ .



And so ?

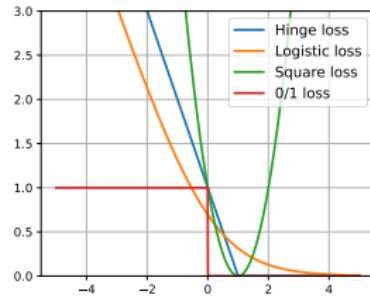
- ▶ Logistic regression = fitting  $f(\mathbf{x}) = \theta^\top \mathbf{x} + b$  with the logistic loss.
- ▶ The decision/prediction of the label is  $\text{sign}(f(\mathbf{x}))$ .
- ▶ So it is a **linear decision boundary** (linear classification).

# First simple neural network: logistic regression

## Remember your losses

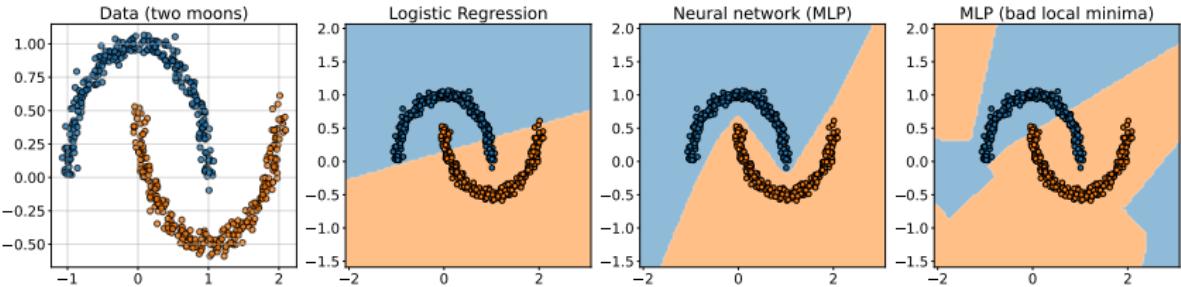
With  $f : \mathbb{R}^d \rightarrow \mathbb{R}$ , many losses can be written as  $\ell(y_i, f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i))$  with  $\Phi \downarrow$ .

- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \mathbf{1}_{y_i f(\mathbf{x}_i) \leq 0}$ .
- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \max\{0, 1 - y_i f(\mathbf{x}_i)\}$ .
- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \log(1 + e^{-y_i f(\mathbf{x}_i)})$ .



And so ?

- ▶ Logistic regression = fitting  $f(\mathbf{x}) = \theta^\top \mathbf{x} + b$  with the logistic loss.
- ▶ The decision/prediction of the label is  $\text{sign}(f(\mathbf{x}))$ .
- ▶ So it is a **linear decision boundary** (linear classification).



# Table of contents

---

From neural networks...

The basic ideas

Logistic regression and one layer neural-network

**Convolutional neural networks**

... to unsupervised node embeddings techniques...

A chronological start

... to graph neural networks

Learning with graphs

What is a GNN ?

A bit of group theory

Invariance and equivariance

Message-passing neural networks

Examples of GNN

The whole pipeline

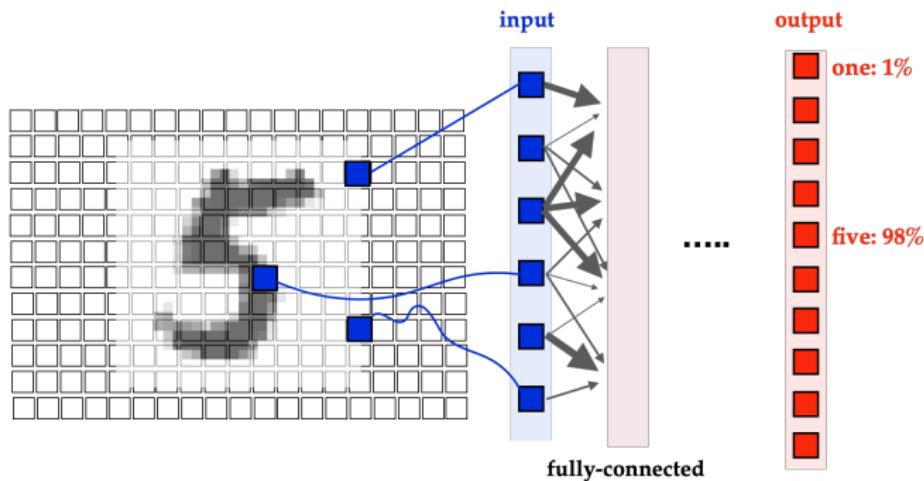
Expressivity of GNN

Conclusion

# Convolutional neural networks

- ▶ The core block for deep learning on images.
- ▶ Induces an **implicit bias** on the architecture.

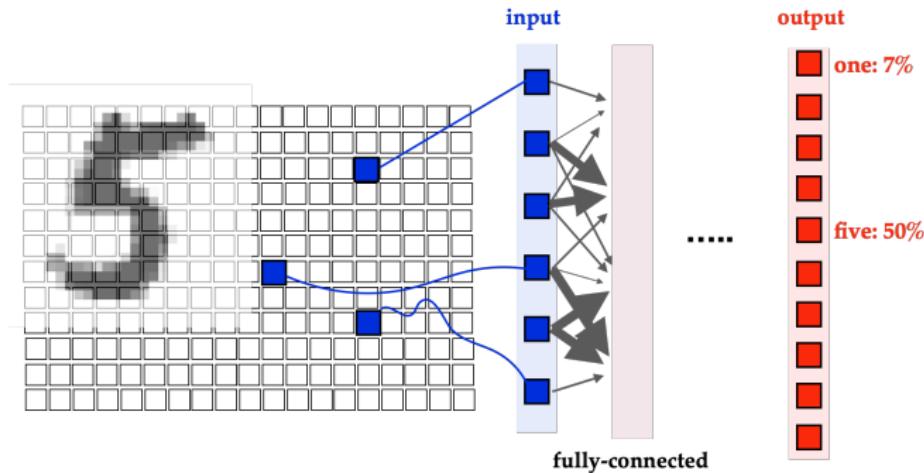
What could happen with a fully-connected architecture?



# Convolutional neural networks

- ▶ The core block for deep learning on images.
- ▶ Induces an **implicit bias** on the architecture.

What could happen with a fully-connected architecture?



- ▶ We want a function that doesn't change if we only translate the image.  
We want a **translation invariant function**.
- ▶ Convolution: particular structure on the weights that induce  
**translation equivariance**.

# Convolutional neural networks

---

## Convolution/correlation of functions

Let  $f, h \in L_2(\mathbb{R})$ . The convolution  $f * h \in L_2(\mathbb{R})$  is defined as

$$f * h(x) = \int_{-\infty}^{+\infty} f(t)h(x-t)dt \text{ and } f \star h(x) = \int_{-\infty}^{+\infty} f(t)h(t+x)dt$$

- ▶ **Translate a filter  $h$**  and then take the inner product with<sup>2</sup>  $f$ :

$$f \star h(x) = \langle \tau_{-x} h, f \rangle_{L_2(\mathbb{R})}.$$

- ▶ It weights the local contributions of  $f$  by a filter.

---

<sup>2</sup> $\tau_x f = t \rightarrow f(t-x)$

# Convolutional neural networks

---

## Convolution/correlation of functions

Let  $f, h \in L_2(\mathbb{R})$ . The convolution  $f * h \in L_2(\mathbb{R})$  is defined as

$$f * h(x) = \int_{-\infty}^{+\infty} f(t)h(x-t)dt \text{ and } f \star h(x) = \int_{-\infty}^{+\infty} f(t)h(t+x)dt$$

- ▶ **Translate a filter  $h$**  and then take the inner product with<sup>2</sup>  $f$ :

$$f \star h(x) = \langle \tau_{-x}h, f \rangle_{L_2(\mathbb{R})}.$$

- ▶ It weights the local contributions of  $f$  by a filter.
- ▶ It is **translation equivariant**.

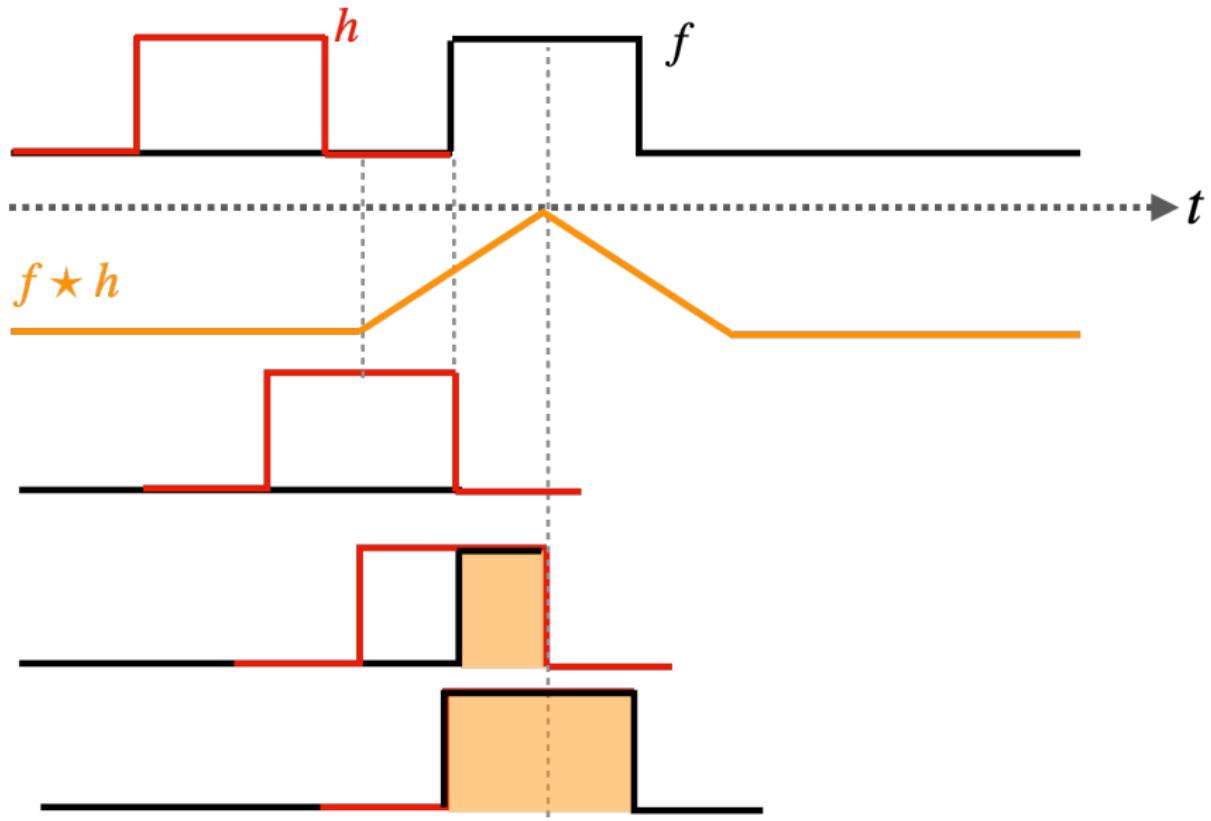
$$(\tau_x f) * h = \tau_x(f * h)$$

- ▶ If we translate the input, the output will be equally translated.

---

<sup>2</sup> $\tau_x f = t \rightarrow f(t-x)$

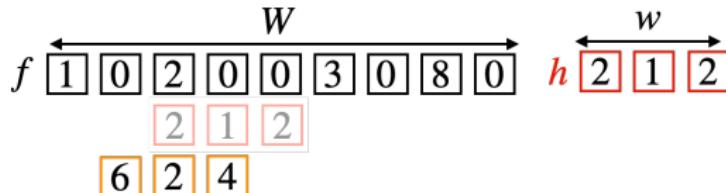
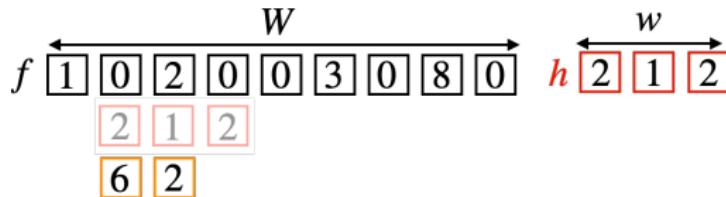
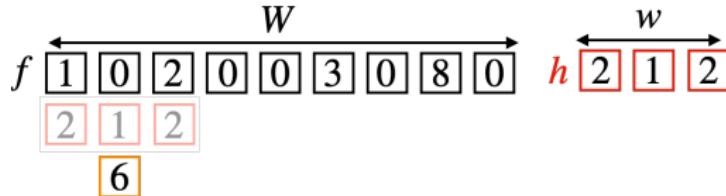
# Convolutional neural networks



# Convolutional neural networks

In practice convolutions are applied on discrete signals.

## Discrete convolutions in 1D

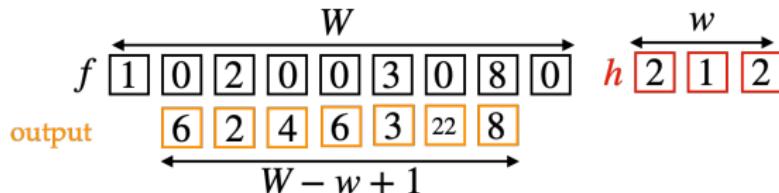


Question: size of the output ?

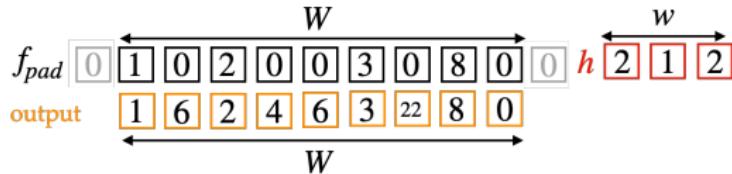
# Convolutional neural networks

In practice convolutions are applied on discrete signals.

## Discrete convolutions in 1D



- ▶ Padding strategies can be used to have output of the same size.



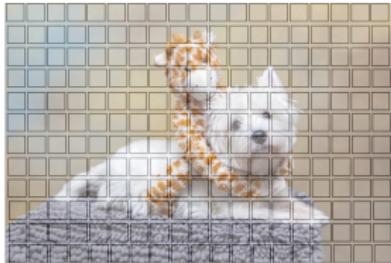
- ▶ Also stride can be used to move the filter from more than one pixel.

# Convolutional neural networks

Discrete convolutions **not** in 1D

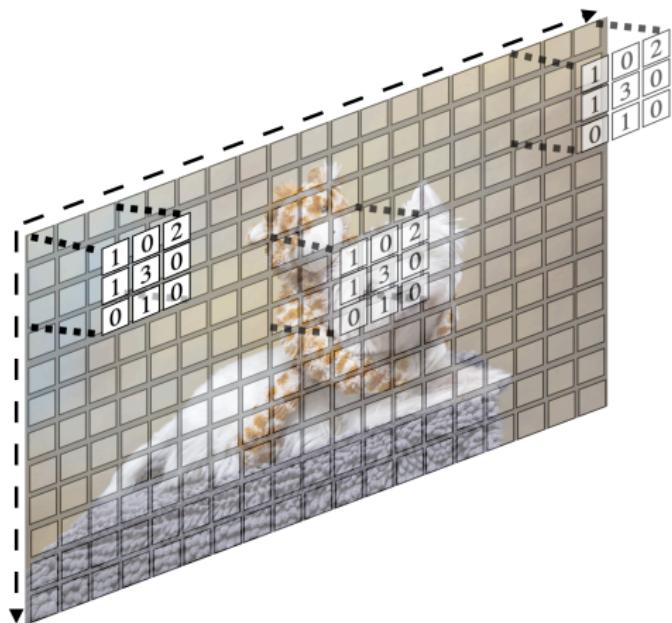
See also [https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic).

Image



Filter

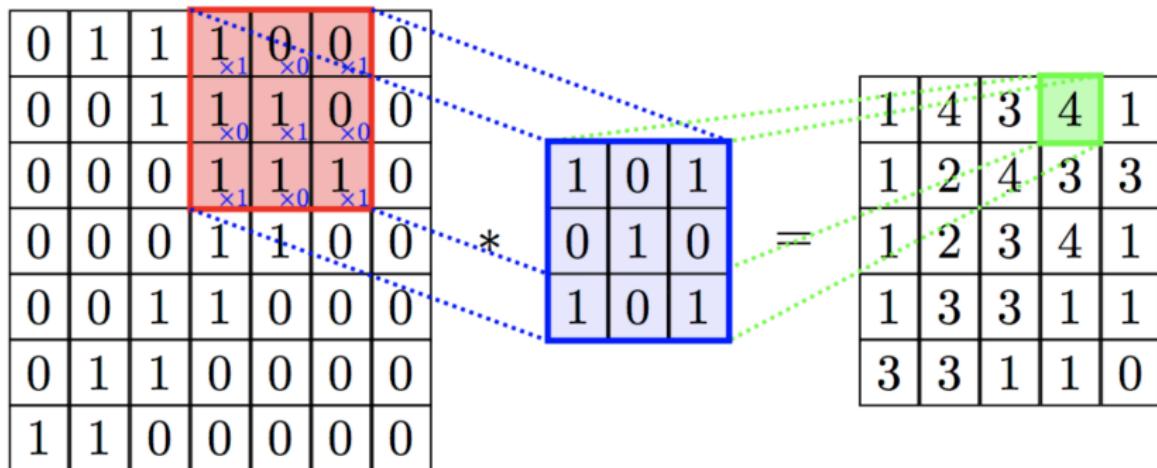
1	0	2
1	3	0
0	1	0



# Convolutional neural networks

## Discrete convolutions **not** in 1D

See also [https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic).



# Convolutional neural networks

Discrete convolutions **not** in 1D

See also [https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic).

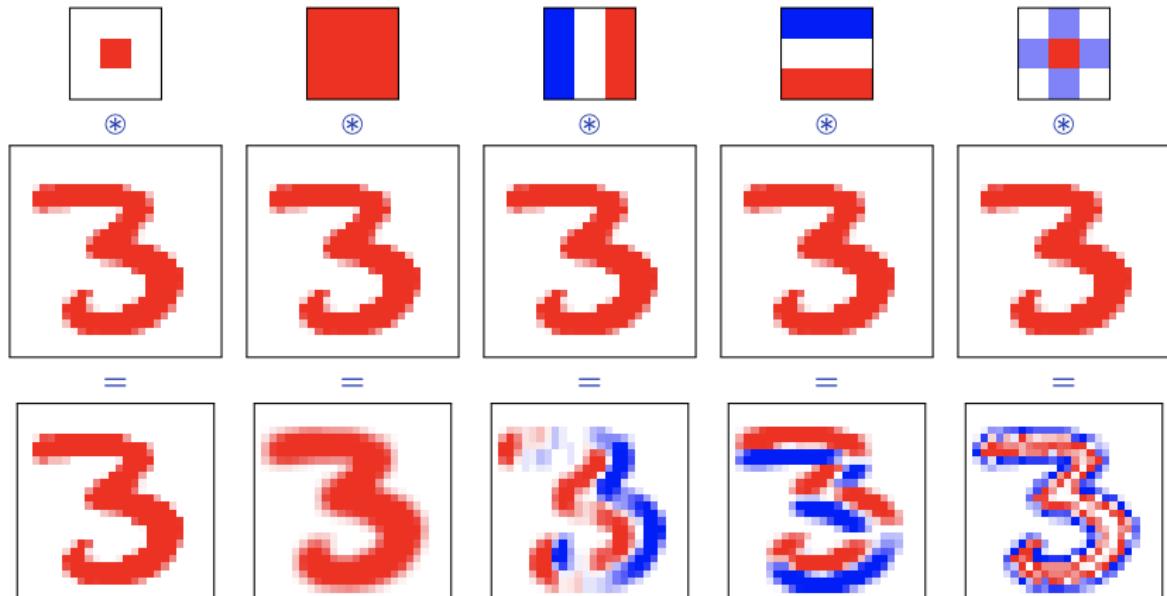


Figure: From Francois Fleuret <https://fleuret.org/dlc/>

# Convolutional neural networks

Discrete convolutions **not** in 1D

See also [https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic).

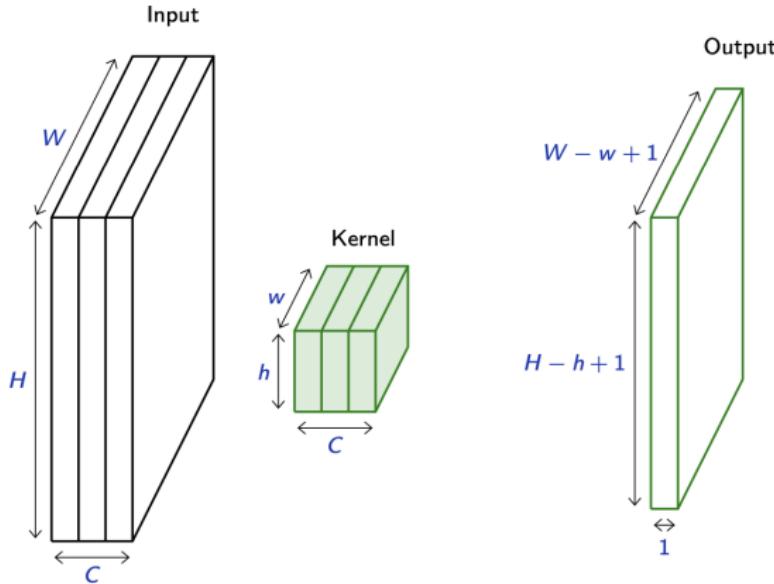


Figure: From Fran ois Fleuret <https://fleuret.org/dlc/>

# Convolutional neural networks

Discrete convolutions **not** in 1D

See also [https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic).

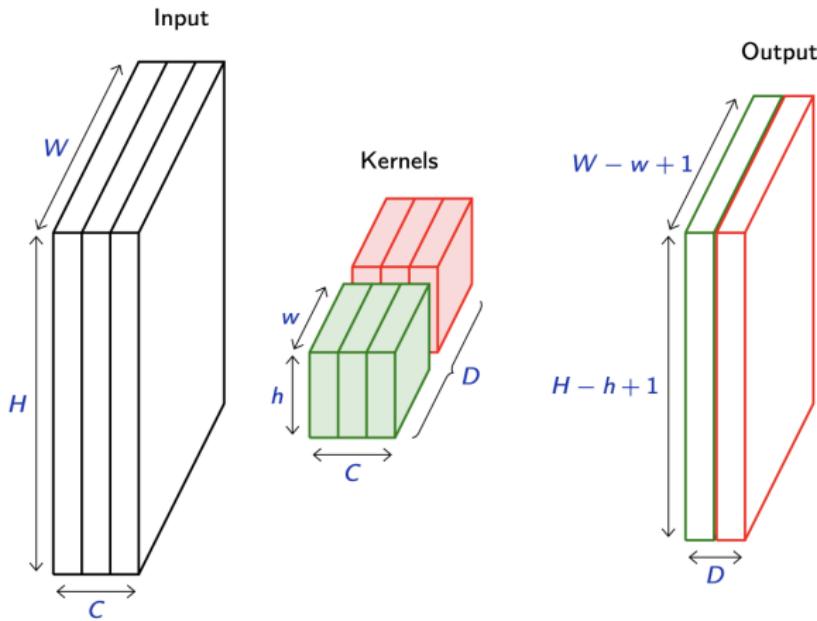


Figure: From Fran<sup>c</sup>cois Fleuret <https://fleuret.org/dlc/>

# Convolutional neural networks

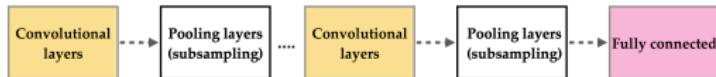
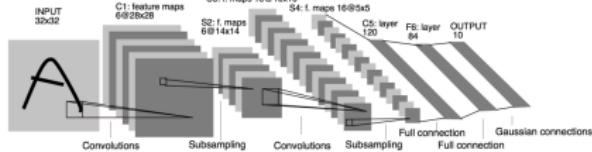


Figure: Schematic view

Figure: LeNet from LeCun et al. 1998

## Principle and intuition (Zeiler and Fergus 2014)

- ▶ Define multiple convolutions, **learn the corresponding filter weights**.
- ▶ Recognize local patterns in images.
- ▶ Find intermediate features that are “general” and “adaptive” due to the translation equivariance bias  
<https://fabianfuchsml.github.io/equivariance1of2/>.
- ▶ Revealing local features that are shared across the data domain.

# Conclusion

---

- ▶ Deep learning: in almost everything when there are images.
- ▶ Very versatile: learn complex functions.
- ▶ Prior also helps ! (translation equivariance).
- ▶ Side note: still struggles on tabular data ([Grinsztajn, Oyallon, and Varoquaux 2022](#)).

# Conclusion

---

- ▶ Deep learning: in almost everything when there are images.
- ▶ Very versatile: learn complex functions.
- ▶ Prior also helps ! (translation equivariance).
- ▶ Side note: still struggles on tabular data ([Grinsztajn, Oyallon, and Varoquaux 2022](#)).

## Graph neural networks ?

- ▶ How do we extend neural networks to graphs?
- ▶ Careful to node ordering: must be invariant to relabelling of the nodes (graph isomorphism).

# Table of contents

---

From neural networks...

The basic ideas

Logistic regression and one layer neural-network

Convolutional neural networks

... to unsupervised node embeddings techniques...

A chronological start

... to graph neural networks

Learning with graphs

What is a GNN ?

A bit of group theory

Invariance and equivariance

Message-passing neural networks

Examples of GNN

The whole pipeline

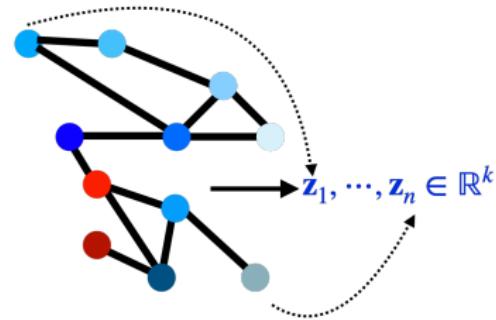
Expressivity of GNN

Conclusion

# Objective

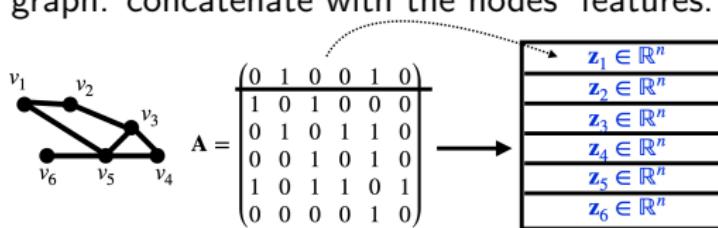
## A chronological start

- ▶ Idea: to learn on a graph: nodes → vector → standard ML pipeline.
- ▶ The embedding **must take into account the structure** of the graph.
- ▶ Also useful for visualization.



## One naive approach

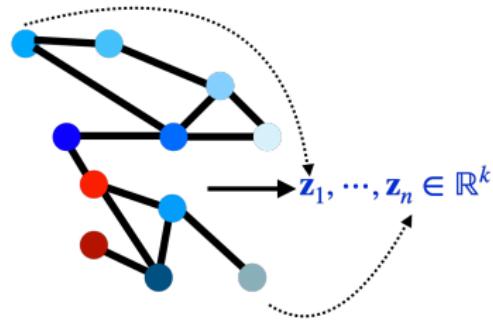
- ▶ Consider each row of the adjacency matrix as an embedding vector.
- ▶ If labelled graph: concatenate with the nodes' features.



# Objective

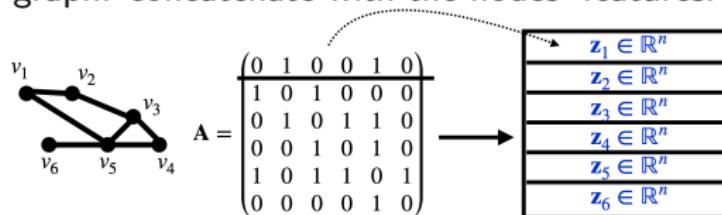
## A chronological start

- ▶ Idea: to learn on a graph: nodes → vector → standard ML pipeline.
- ▶ The embedding **must take into account the structure** of the graph.
- ▶ Also useful for visualization.



## One naive approach

- ▶ Consider each row of the adjacency matrix as an embedding vector.
- ▶ If labelled graph: concatenate with the nodes' features.



- ▶ Sensitive to the node ordering ! Also, expensive  $O(|V|)$  !
- ▶ Not applicable to graph with different sizes !

# An encoder-decoder perspective

---

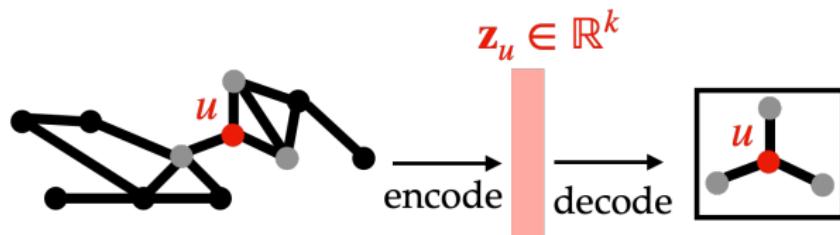
## Notations

- ▶ We suppose we have one graph  $G = (V, E)$ , without features (so far).
- ▶ For each  $u \in V$  we look for an embedding  $\mathbf{z}_u \in \mathbb{R}^k$ .

## Principle

We look for a “good” encoder  $E : V \rightarrow \mathbb{R}^k$  such that  $E(u) = \mathbf{z}_u$ .

- ▶ Ideally the embedding  $\mathbf{z}_u$  contains the neighbourhood informations of  $u$ .

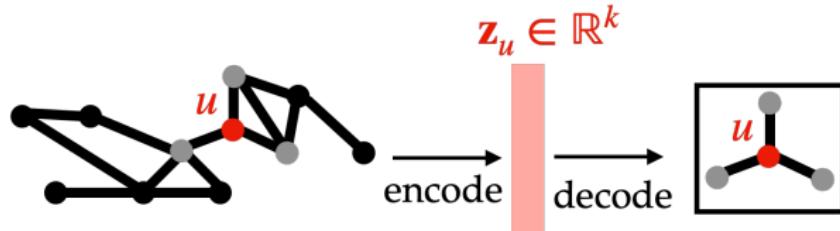


# An encoder-decoder perspective

## Principle

We look for a “good” encoder  $E : V \rightarrow \mathbb{R}^k$  such that  $E(u) = \mathbf{z}_u$ .

- ▶ Ideally the embedding  $\mathbf{z}_u$  contains the neighbourhood informations of  $u$ .



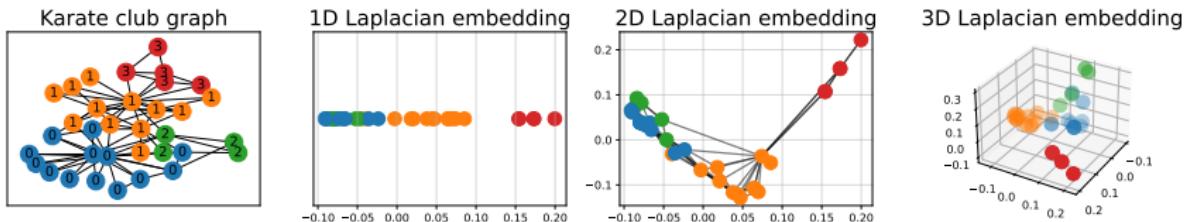
## Encoding/decoding scheme

A lot of methods attempt to minimize

$$\mathcal{L} = \sum_{(u,v) \in \mathcal{D}} \ell(\text{similarity}(\mathbf{z}_u, \mathbf{z}_v), S[u, v])$$

- ▶  $\text{similarity}(\mathbf{z}_u, \mathbf{z}_v)$  how close are the embeddings.
- ▶  $S[u, v]$  how close are the nodes in the graph.
- ▶  $\ell : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$  is a loss: how similar are the similarities.

# Unsupervised node embeddings techniques



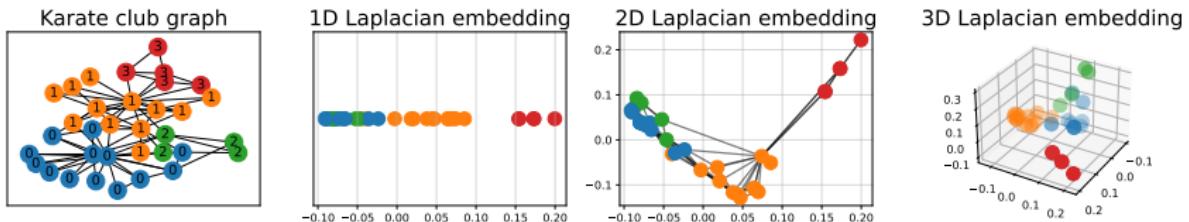
Inspiration from Laplacian eigenmaps Belkin and Niyogi 2003

- ▶ In the embedding space similarity( $\mathbf{z}_u, \mathbf{z}_v$ ) =  $\frac{1}{2} \|\mathbf{z}_u - \mathbf{z}_v\|_2^2$ .
- ▶ When similiary is  $\mathbf{S}[u_i, v_j] = A_{ij} / \sqrt{\text{degree}(u_i)} \sqrt{\text{degree}(u_j)}$ , loss to minimize:

$$\frac{1}{2} \sum_{ij} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 \frac{A_{ij}}{\sqrt{\text{degree}(u_i)} \sqrt{\text{degree}(u_j)}} = \text{tr}(\mathbf{Z}^\top \tilde{\mathbf{L}} \mathbf{Z}).$$

- ▶ Normalized Laplacian  $\tilde{\mathbf{L}} = \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2}$ .
- ▶ Interpretation + permutation equivariance of the cost (on the board).

# Unsupervised node embeddings techniques



Inspiration from Laplacian eigenmaps Belkin and Niyogi 2003

- ▶ In the embedding space similarity( $\mathbf{z}_u, \mathbf{z}_v$ ) =  $\frac{1}{2} \|\mathbf{z}_u - \mathbf{z}_v\|_2^2$ .
- ▶ When similiary is  $\mathbf{S}[u_i, v_j] = A_{ij} / \sqrt{\text{degree}(u_i)} \sqrt{\text{degree}(u_j)}$ , loss to minimize:

$$\frac{1}{2} \sum_{ij} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 \frac{A_{ij}}{\sqrt{\text{degree}(u_i)} \sqrt{\text{degree}(u_j)}} = \text{tr}(\mathbf{Z}^\top \tilde{\mathbf{L}} \mathbf{Z}).$$

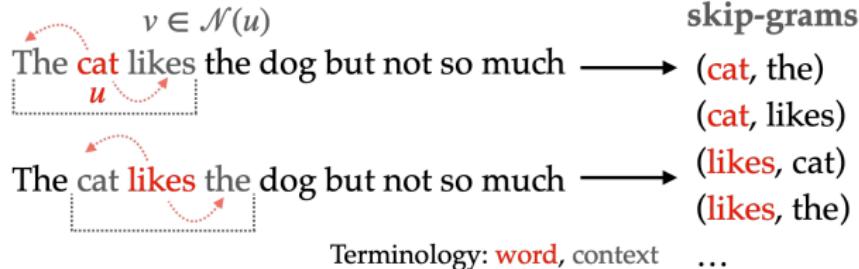
- ▶ Normalized Laplacian  $\tilde{\mathbf{L}} = \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2}$ .
- ▶ Interpretation + permutation equivariance of the cost (on the board).
- ▶ With the constraint  $\mathbf{Z}^\top \mathbf{Z} = \mathbf{I}_d$  it recovers Laplacian eigenmaps.
- ▶ Sol. is the  $d$  eigenvectors associated to the  $d$  smallest eigenvalues of  $\tilde{\mathbf{L}}$ .

# Unsupervised node embeddings techniques

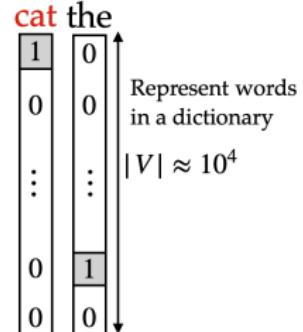
## Skip-Gram and the Word2vec model (Mikolov et al. 2013)

The meaning of a word is its use in language (Wittgenstein).

- ▶ Objective: “similar” words are embedded into “similar” vectors.
- ▶ Goal: predict context words from each **input word**.
- ▶ We want to maximize  $\mathbb{P}(\text{context}|\text{input word})$ .



### One hot encoding

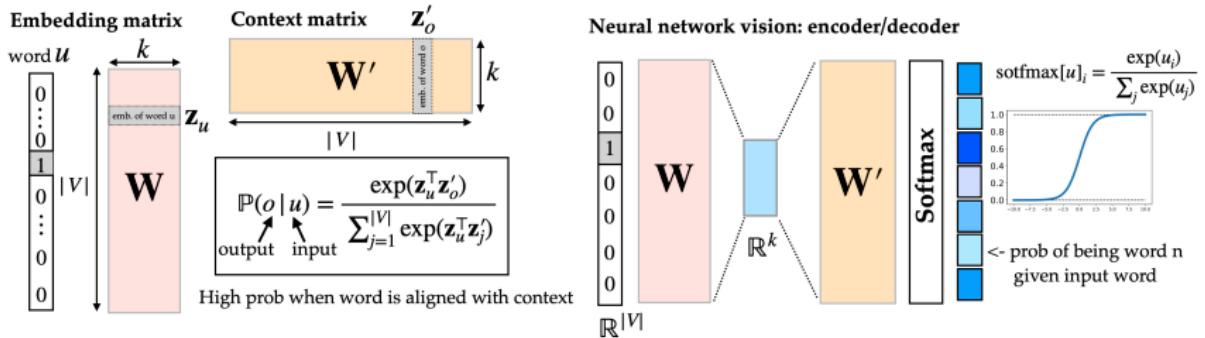


# Unsupervised node embeddings techniques

## Skip-Gram and the Word2vec model (Mikolov et al. 2013)

The meaning of a word is its use in language (Wittgenstein).

- ▶ Objective: “similar” words are embedded into “similar” vectors.
- ▶ Goal: predict context words from each **input word**.
- ▶ We want to maximize  $\mathbb{P}(\text{context}|\text{input word})$ .

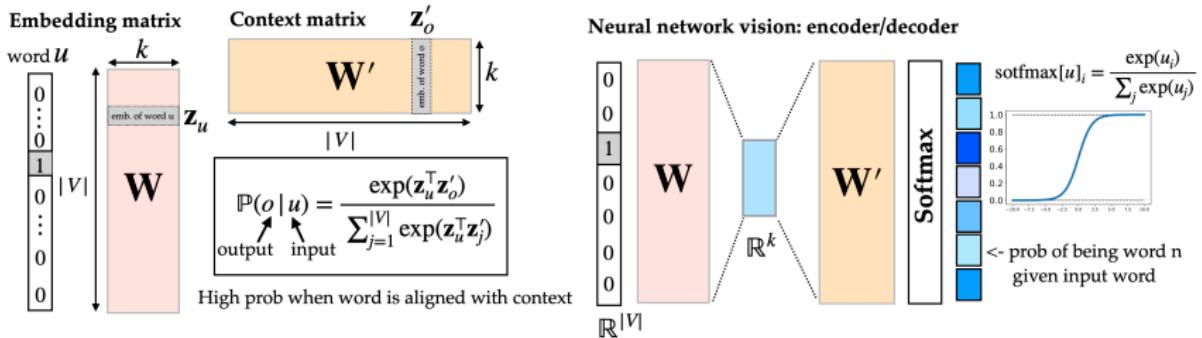


# Unsupervised node embeddings techniques

## Skip-Gram and the Word2vec model (Mikolov et al. 2013)

The meaning of a word is its use in language (Wittgenstein).

- ▶ Objective: “similar” words are embedded into “similar” vectors.
- ▶ Goal: predict context words from each **input word**.
- ▶ We want to maximize  $\mathbb{P}(\text{context}|\text{input word})$ .



- ▶ Dataset  $\mathcal{D}$  of input/output words (surrounding). Loss to minimize is:

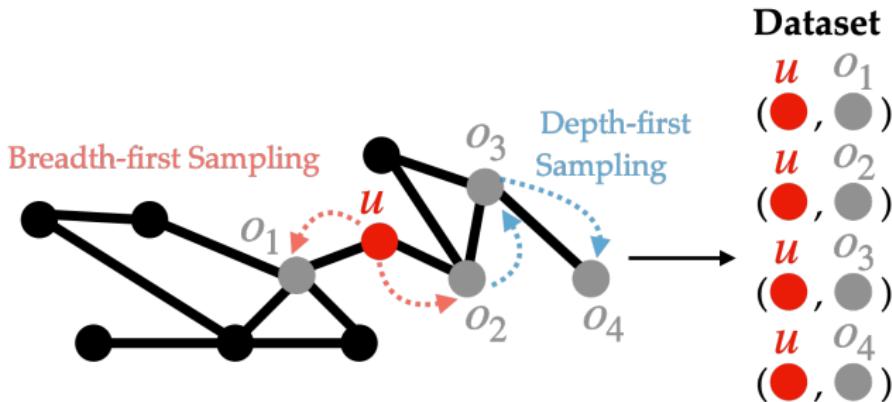
$$-\sum_{(u,o) \in \mathcal{D}} \log \mathbb{P}(o|u).$$

- ▶ But computing it in  $\mathcal{O}(|V| \times |\{\text{words to embed}\}|)$ : negative sampling.

# Unsupervised node embeddings techniques

## The node2vec model (Grover and Leskovec 2016)

- ▶ Similar as before: each node  $u \in V$  is embedded as  $\mathbf{z}_u \in \mathbb{R}^k$ .
- ▶ Goal of the embedding: reflect the neighboring nodes of  $u$ .
- ▶ Sampling strategies based on random walks (BFS/DFS).



- ▶ With a dataset  $\mathcal{D}$  of input/output nodes. Loss to minimize:

$$\mathcal{L} = - \sum_{(u,o) \in \mathcal{D}} \log \frac{\exp(\mathbf{z}_u^\top \mathbf{z}_o)}{\sum_{w \in V} \exp(\mathbf{z}_u^\top \mathbf{z}_w)}$$

# Unsupervised node embeddings techniques

## The node2vec model (Grover and Leskovec 2016)

- ▶ Similar as before: each node  $u \in V$  is embedded as  $\mathbf{z}_u \in \mathbb{R}^k$ .
- ▶ Goal of the embedding: reflect the neighboring nodes of  $u$ .
- ▶ Sampling strategies based on random walks (BFS/DFS).

## Negative sampling (NS)

- ▶ Loss is too expensive to compute  $\mathcal{O}(|V|^2)$ .
- ▶ NS: introduce negative data samples.
- ▶ Goal: distinguish between neighboring points of a target node  $u$  and random nodes drawn from a noise distribution using logistic regression.
- ▶ New loss (explanations on the board) (Goldberg and Levy 2014):

$$\mathcal{L} = - \left( \sum_{(u_+, o_+) \in \mathcal{D}_+} \log \sigma(\mathbf{z}_u^\top \mathbf{z}_o) + \sum_{(u_-, o_-) \in \mathcal{D}_-} \log \sigma(-\mathbf{z}_u^\top \mathbf{z}_o) \right)$$

with sigmoid function  $\sigma(x) = \frac{1}{1 + \exp(-x)}$ .

# Unsupervised node embeddings techniques

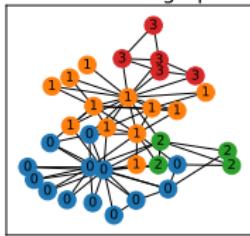
## The node2vec model (Grover and Leskovec 2016)

- ▶ Similar as before: each node  $u \in V$  is embedded as  $\mathbf{z}_u \in \mathbb{R}^k$ .
- ▶ Goal of the embedding: reflect the neighboring nodes of  $u$ .
- ▶ Sampling strategies based on random walks (BFS/DFS).

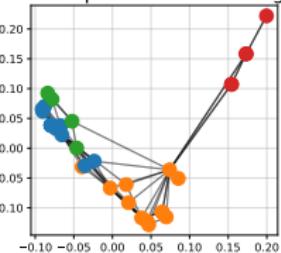
## Negative sampling (NS)

- ▶ Goal: distinguish between neighboring points of a target node  $u$  and random draws from a noise distribution using logistic regression.

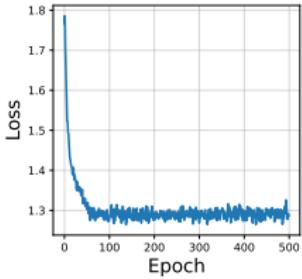
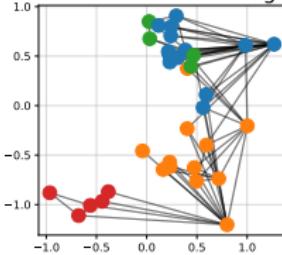
Karate club graph



2D Laplacian embedding



node2vec embedding



# Unsupervised node embeddings techniques

---

## Limitations of previous embeddings techniques

- ▶ The previous embeddings are called **shallow**: encoder function  $E : V \rightarrow \mathbb{R}^k$  is simply an embedding lookup based on the node ID.

$$E(u) = \mathbf{Z}[:, u] = \mathbf{z}_u .$$

# Unsupervised node embeddings techniques

---

## Limitations of previous embeddings techniques

- ▶ The previous embeddings are called **shallow**: encoder function  $E : V \rightarrow \mathbb{R}^k$  is simply an embedding lookup based on the node ID.

$$E(u) = \mathbf{Z}[:, u] = \mathbf{z}_u .$$

- ▶ Lack of parameter sharing between nodes in the encoder.
- ▶ Do not leverage node features !
- ▶ Inherently **transductive**: these methods can only generate embeddings for nodes that were present during the training phase.
- ▶ **If new nodes must retrain everything.**

# Table of contents

---

From neural networks...

The basic ideas

Logistic regression and one layer neural-network

Convolutional neural networks

... to unsupervised node embeddings techniques...

A chronological start

... to graph neural networks

Learning with graphs

What is a GNN ?

A bit of group theory

Invariance and equivariance

Message-passing neural networks

Examples of GNN

The whole pipeline

Expressivity of GNN

Conclusion

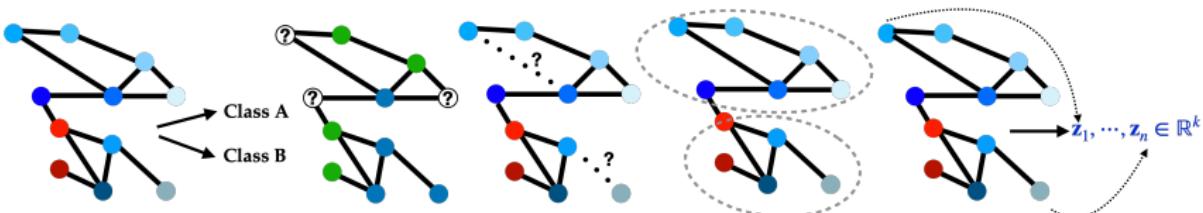
# Frameworks considered here

Supervised:

- ▶ Graph classification: labelled graphs → label new graph (molecule classification, drug efficiency prediction).
- ▶ Node (or edge) classification: labelled nodes → label other nodes (advertisement, protein interface prediction).

Unsupervised (semi-supervised):

- ▶ Community detection: one graph → group nodes (social network analysis).
- ▶ Link prediction: one graph → potential new edge.
- ▶ Unsupervised node embeddings.



# Some limitations

---

## Tip of the iceberg

- ▶ Approx. 100 GNN papers a month on arXiv.
- ▶ Despite 1000s of papers, same ideas coming round: be critical, learn to spot incremental changes!
- ▶ We will only see the most well-known architectures (according to me).
- ▶ Be aware that it might already be out-of-date.
- ▶ Some surveys [Wu et al. 2021](#); [Zhang, Cui, and Zhu 2020](#); [William L Hamilton 2020](#).
- ▶ See also <https://github.com/houchengbin/awesome-GNN-papers>.

# Table of contents

---

From neural networks...

The basic ideas

Logistic regression and one layer neural-network

Convolutional neural networks

... to unsupervised node embeddings techniques...

A chronological start

... to graph neural networks

Learning with graphs

What is a GNN ?

A bit of group theory

Invariance and equivariance

Message-passing neural networks

Examples of GNN

The whole pipeline

Expressivity of GNN

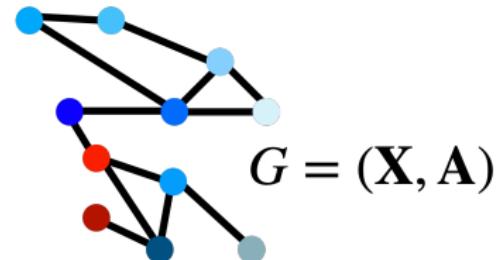
Conclusion

# What is a graph neural network ?

---

## Framework

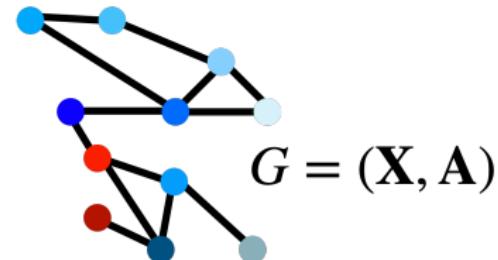
- ▶ Graphs considered here:
- ▶  $G = (V, E)$  with  $|V| = n$ , features on the nodes.
- ▶ Adjacency matrix  $\mathbf{A} \in \mathbb{R}^{n \times n}$ .
- ▶ Feature matrix  $\mathbf{X} \in \mathbb{R}^{n \times d}$ , feature  $\mathbf{x}_i \in \mathbb{R}^d$ .



# What is a graph neural network ?

## Framework

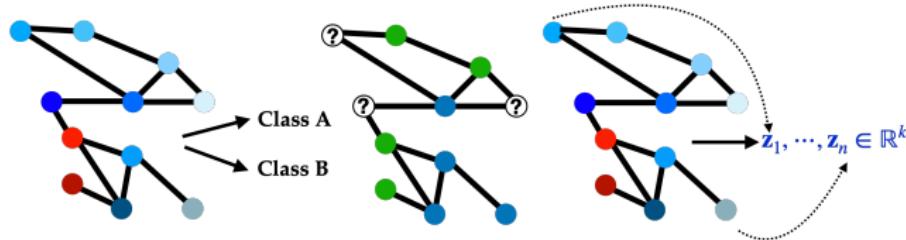
- ▶ Graphs considered here:
- ▶  $G = (V, E)$  with  $|V| = n$ , features on the nodes.
- ▶ Adjacency matrix  $\mathbf{A} \in \mathbb{R}^{n \times n}$ .
- ▶ Feature matrix  $\mathbf{X} \in \mathbb{R}^{n \times d}$ , feature  $\mathbf{x}_i \in \mathbb{R}^d$ .



## GNN general definition

A GNN is a **specific parametrized function** that takes a input a graph  $G = (\mathbf{X}, \mathbf{A})$  and outputs “something” (depends on the application).

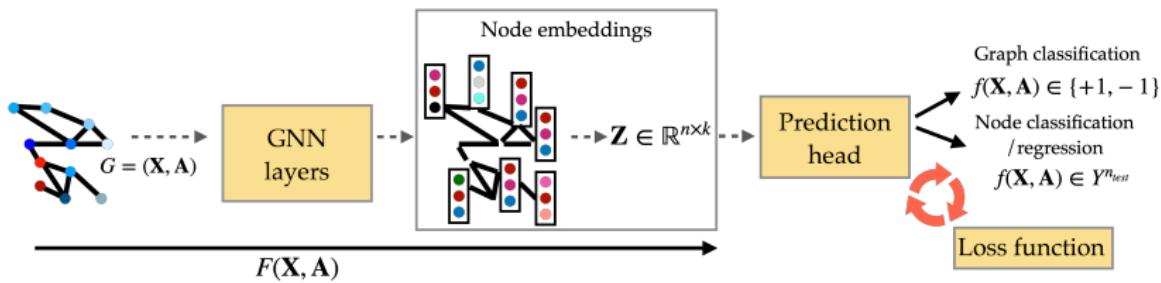
- ▶ It is made of a **combination of different layers**.
- ▶ Graph classification, node classification/regression, node embedding



- ▶ Notations: vector output  $f(\mathbf{X}, \mathbf{A})$ , matrix output  $F(\mathbf{X}, \mathbf{A})$ .

# What properties to ensure ?

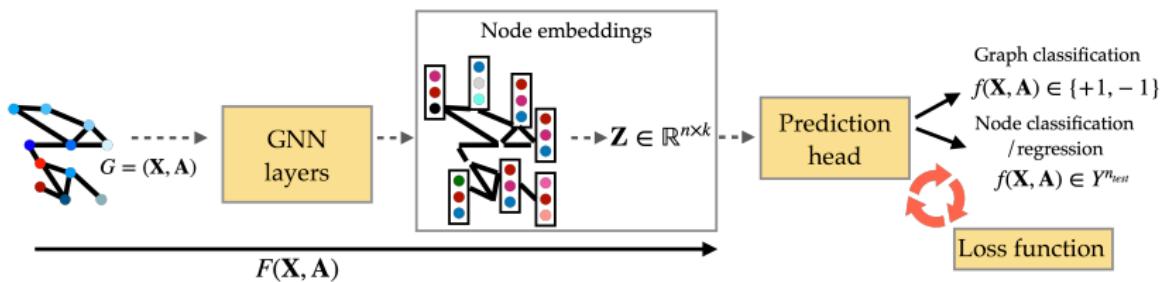
## The training pipeline



- ▶ Overall the same procedure: find an embedding of the nodes  $F(\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{n \times k}$  (supervised or unsupervised) and then do stuff.

# What properties to ensure ?

## The training pipeline



- ▶ Overall the same procedure: find an embedding of the nodes  $F(\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{n \times k}$  (supervised or unsupervised) and then do stuff.

## Properties to ensure

- ▶ If graph classification then  $f(\mathbf{X}, \mathbf{A}) \in \pm 1$ : the function must be **invariant to permutations of the graph**.
- ▶ Prediction on the node level: we want to let the permutation of the graph produce a different result but while **making this phenomena predictable**.
- ▶ It will be formalized with the notion of invariance/equivariance.

# Table of contents

---

From neural networks...

The basic ideas

Logistic regression and one layer neural-network

Convolutional neural networks

... to unsupervised node embeddings techniques...

A chronological start

... to graph neural networks

Learning with graphs

What is a GNN ?

**A bit of group theory**

Invariance and equivariance

Message-passing neural networks

Examples of GNN

The whole pipeline

Expressivity of GNN

Conclusion

# On invariance and equivariance

The right symmetries can facilitate learning

- ▶ Fit a polynomial  $\hat{f}(x) = \sum_{n=0}^N \theta_n x^n$  on

symmetric:  $f(-x) = f(x)$

antisymmetric:  $f(-x) = -f(x)$

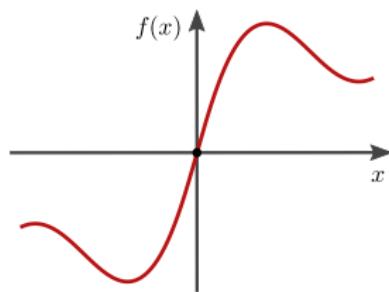
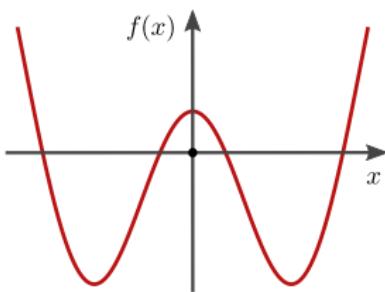


Figure: From Weiler et al. 2023

- ▶ Ignore prior knowledge about the function.
- ▶ Better: fit  $\sum_{n \text{ even}}^N \theta_n x^n$  (invariant) or  $\sum_{n \text{ odd}}^N \theta_n x^n$  (equivariant).
- ▶ Need half of the parameters + generalize well.

# On invariance and equivariance

On the previous episodes

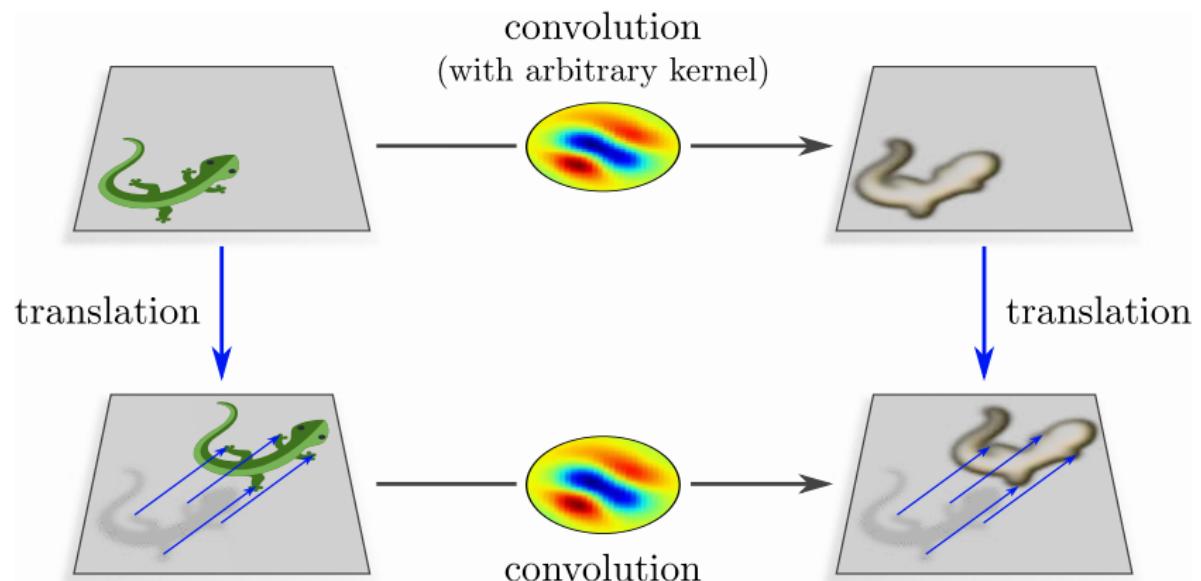


Figure: From Weiler et al. 2023

# On invariance and equivariance

---

## A little bit of group theory

A group  $\mathfrak{G}$  is a set along with a binary operation  $\circ : \mathfrak{G} \times \mathfrak{G} \rightarrow \mathfrak{G}$  satisfying

- ▶ *Associativity*:  $\forall g, h, i \in \mathfrak{G}, (g \circ h) \circ i = g \circ (h \circ i)$ .
- ▶ *Identity*: there exists  $e \in \mathfrak{G}$  such that  $\forall g \in \mathfrak{G}, g \circ e = e \circ g = g$ .
- ▶ *Inverse*: For each  $g \in \mathfrak{G}$  there exists  $g^{-1} \in \mathfrak{G}$  such that  
 $g \circ g^{-1} = g^{-1} \circ g = e$ .
- ▶ *Closure*:  $\forall g, h \in \mathfrak{G}, g \circ h \in \mathfrak{G}$ .

Commutativity is not part of this definition ( $g \circ h \neq h \circ g$ ).

# On invariance and equivariance

---

## A little bit of group theory

A group  $\mathfrak{G}$  is a set along with a binary operation  $\circ : \mathfrak{G} \times \mathfrak{G} \rightarrow \mathfrak{G}$  satisfying

- ▶ *Associativity*:  $\forall g, h, i \in \mathfrak{G}, (g \circ h) \circ i = g \circ (h \circ i)$ .
- ▶ *Identity*: there exists  $e \in \mathfrak{G}$  such that  $\forall g \in \mathfrak{G}, g \circ e = e \circ g = g$ .
- ▶ *Inverse*: For each  $g \in \mathfrak{G}$  there exists  $g^{-1} \in \mathfrak{G}$  such that  $g \circ g^{-1} = g^{-1} \circ g = e$ .
- ▶ *Closure*:  $\forall g, h \in \mathfrak{G}, g \circ h \in \mathfrak{G}$ .

Commutativity is not part of this definition ( $g \circ h \neq h \circ g$ ).

## Some examples

- ▶ Translation group on  $\mathbb{Z}^2$  is an Abelian group:

$$(m, n) \circ (p, q) = (n + p, m + q).$$

- ▶ Translation + rotations, mirror reflections.
- ▶ Permutation group  $S_n = \{\sigma : \llbracket n \rrbracket \rightarrow \llbracket n \rrbracket, \sigma \text{ is a bijection}\}$  with the composition of functions.

# On invariance and equivariance

---

## Group action

Given a set  $\Omega$  and a group  $\mathfrak{G}$ , a (left) **group action** of  $\mathfrak{G}$  on  $\Omega$  is a function

$$\begin{aligned}\mathfrak{G} \times \Omega &\rightarrow \Omega \\ (\mathfrak{g}, x) &\rightarrow \mathfrak{g}x\end{aligned}$$

satisfying

- ▶  $\forall x \in \Omega, \mathfrak{e}x = x$
- ▶ *Compatibility:*  $\forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{G}, \forall x \in \Omega, \mathfrak{g}(\mathfrak{h}x) = (\mathfrak{g} \circ \mathfrak{h})x.$
- ▶ It acts on the element of the sets via the group.
- ▶ **A set endowed with an action of  $\mathfrak{G}$  on it is called a  $\mathfrak{G}$ -set.**

# On invariance and equivariance

---

## Group action

Given a set  $\Omega$  and a group  $\mathfrak{G}$ , a (left) **group action** of  $\mathfrak{G}$  on  $\Omega$  is a function

$$\begin{aligned}\mathfrak{G} \times \Omega &\rightarrow \Omega \\ (\mathfrak{g}, x) &\mapsto \mathfrak{g}x\end{aligned}$$

satisfying

- ▶  $\forall x \in \Omega, \mathfrak{e}x = x$
- ▶ *Compatibility:*  $\forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{G}, \forall x \in \Omega, \mathfrak{g}(\mathfrak{h}x) = (\mathfrak{g} \circ \mathfrak{h})x.$
- ▶ It acts on the element of the sets via the group.
- ▶ **A set endowed with an action of  $\mathfrak{G}$  on it is called a  $\mathfrak{G}$ -set.**

## Translation of functions

- ▶ Group of translations  $\mathfrak{G} = \{\tau_x, x \in \mathbb{R}\}$  with  $\tau_x \circ \tau_y = \tau_{x+y}$ . Identity element  $\tau_0$ .
- ▶ For a function  $f$  and  $\tau_x$  the group action

$$\tau_x f := t \rightarrow f(t - x).$$

# On invariance and equivariance

---

## Group action

Given a set  $\Omega$  and a group  $\mathfrak{G}$ , a (left) **group action** of  $\mathfrak{G}$  on  $\Omega$  is a function

$$\mathfrak{G} \times \Omega \rightarrow \Omega$$

$$(\mathfrak{g}, x) \mapsto \mathfrak{g}x$$

satisfying

- ▶  $\forall x \in \Omega, \mathfrak{e}x = x$
- ▶ *Compatibility:*  $\forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{G}, \forall x \in \Omega, \mathfrak{g}(\mathfrak{h}x) = (\mathfrak{g} \circ \mathfrak{h})x.$
- ▶ It acts on the element of the sets via the group.
- ▶ **A set endowed with an action of  $\mathfrak{G}$  on it is called a  $\mathfrak{G}$ -set.**

## Permutation of vectors

- ▶ Group of permutations  $S_n$  with composition  $\circ$ . Identity element id.
- ▶ For  $\mathbf{x} \in \mathbb{R}^n$  a group action is  $\sigma\mathbf{x} = (x_{\sigma(1)}, x_{\sigma(2)}, \dots, x_{\sigma(n)}).$
- ▶ Is it a left group action ?

# On invariance and equivariance

---

## Group action

Given a set  $\Omega$  and a group  $\mathfrak{G}$ , a (left) **group action** of  $\mathfrak{G}$  on  $\Omega$  is a function

$$\mathfrak{G} \times \Omega \rightarrow \Omega$$

$$(\mathfrak{g}, x) \rightarrow \mathfrak{g}x$$

satisfying

- ▶  $\forall x \in \Omega, \mathfrak{e}x = x$
- ▶ *Compatibility:*  $\forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{G}, \forall x \in \Omega, \mathfrak{g}(\mathfrak{h}x) = (\mathfrak{g} \circ \mathfrak{h})x.$
- ▶ It acts on the element of the sets via the group.
- ▶ **A set endowed with an action of  $\mathfrak{G}$  on it is called a  $\mathfrak{G}$ -set.**

## Permutation of vectors

- ▶ For  $\mathbf{x} \in \mathbb{R}^n$  a group action is  $\sigma\mathbf{x} = (x_{\sigma(1)}, x_{\sigma(2)}, \dots, x_{\sigma(n)}).$
- ▶ Def  $(\sigma_1\mathbf{x})_i = x_{\sigma_1(i)}$ . So  $(\sigma_2(\sigma_1\mathbf{x}))_i = (\sigma_1\mathbf{x})_{\sigma_2(i)} = x_{\sigma_1(\sigma_2(i))} = x_{\sigma_1 \circ \sigma_2(i)}.$
- ▶ Thus  $\sigma_2(\sigma_1\mathbf{x}) = (\sigma_1 \circ \sigma_2)\mathbf{x} \neq (\sigma_2 \circ \sigma_1)\mathbf{x}.$

# On invariance and equivariance

---

## Group action

Given a set  $\Omega$  and a group  $\mathfrak{G}$ , a (left) **group action** of  $\mathfrak{G}$  on  $\Omega$  is a function

$$\begin{aligned}\mathfrak{G} \times \Omega &\rightarrow \Omega \\ (\mathfrak{g}, x) &\rightarrow \mathfrak{g}x\end{aligned}$$

satisfying

- ▶  $\forall x \in \Omega, \mathfrak{e}x = x$
- ▶ *Compatibility:*  $\forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{G}, \forall x \in \Omega, \mathfrak{g}(\mathfrak{h}x) = (\mathfrak{g} \circ \mathfrak{h})x.$
- ▶ It acts on the element of the sets via the group.
- ▶ **A set endowed with an action of  $\mathfrak{G}$  on it is called a  $\mathfrak{G}$ -set.**

## Permutation of vectors

- ▶ For  $\mathbf{x} \in \mathbb{R}^n$  a **left** group action is  $\sigma\mathbf{x} = (x_{\sigma^{-1}(1)}, x_{\sigma^{-1}(2)}, \dots, x_{\sigma^{-1}(n)}).$
- ▶ Def  $(\sigma_1\mathbf{x})_i = x_{\sigma_1^{-1}(i)}$ . So  
 $(\sigma_2(\sigma_1\mathbf{x}))_i = (\sigma_1\mathbf{x})_{\sigma_2^{-1}(i)} = x_{\sigma_1^{-1}(\sigma_2^{-1}(i))} = x_{(\sigma_2 \circ \sigma_1)^{-1}(i)}.$
- ▶ Thus  $\sigma_2(\sigma_1\mathbf{x}) = (\sigma_2 \circ \sigma_1)\mathbf{x}.$

# Table of contents

---

From neural networks...

The basic ideas

Logistic regression and one layer neural-network

Convolutional neural networks

... to unsupervised node embeddings techniques...

A chronological start

... to graph neural networks

Learning with graphs

What is a GNN ?

A bit of group theory

**Invariance and equivariance**

Message-passing neural networks

Examples of GNN

The whole pipeline

Expressivity of GNN

Conclusion

# A formal definition of invariance

---

## Invariance

Let  $\Omega$  be a  $\mathfrak{G}$ -set. A function  $f : \Omega \rightarrow Y$  is  **$\mathfrak{G}$ -invariant** if

$$\forall x \in \Omega, \forall g \in \mathfrak{G}, f(gx) = f(x).$$

- ▶  $f$  is  $\mathfrak{G}$ -invariant if its output is unaffected by the group action.

# A formal definition of invariance

---

## Invariance

Let  $\Omega$  be a  $\mathfrak{G}$ -set. A function  $f : \Omega \rightarrow Y$  is  **$\mathfrak{G}$ -invariant** if

$$\forall x \in \Omega, \forall g \in \mathfrak{G}, f(gx) = f(x).$$

- ▶  $f$  is  $\mathfrak{G}$ -invariant if its output is unaffected by the group action.

## Permutation invariant functions

Find three functions  $f, g, h : \mathbb{R}^n \rightarrow \mathbb{R}$  that are  $S_n$ -invariant.

# A formal definition of invariance

---

## Invariance

Let  $\Omega$  be a  $\mathfrak{G}$ -set. A function  $f : \Omega \rightarrow Y$  is  **$\mathfrak{G}$ -invariant** if

$$\forall x \in \Omega, \forall g \in \mathfrak{G}, f(gx) = f(x).$$

- ▶  $f$  is  $\mathfrak{G}$ -invariant if its output is unaffected by the group action.

## Permutation invariant functions

- ▶  $f(\mathbf{x}) = \sum_{i=1}^n x_i, g(\mathbf{x}) = \max_{i \in [n]} x_i, h(\mathbf{x}) = \text{sort}(\mathbf{x})$  (to  $\mathbb{R}^n$ ).
- ▶ Characterization of all linear permutation invariant functions  $L : \mathbb{R}^{n^k} \rightarrow \mathbb{R}$  ([Maron et al. 2018](#)).

# A formal definition of invariance

---

## Invariance

Let  $\Omega$  be a  $\mathfrak{G}$ -set. A function  $f : \Omega \rightarrow Y$  is  **$\mathfrak{G}$ -invariant** if

$$\forall x \in \Omega, \forall g \in \mathfrak{G}, f(gx) = f(x).$$

- ▶  $f$  is  $\mathfrak{G}$ -invariant if its output is unaffected by the group action.

## Permutation invariant functions

Let  $\mathbf{X} \in \mathbb{R}^{n \times d}$ . The action of  $\sigma$  on  $\mathbf{X}$  is  $\sigma\mathbf{X} = (X_{\sigma^{-1}(i)j})_{ij}$ . Find a permutation invariant function  $F : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}$ .

# A formal definition of invariance

---

## Invariance

Let  $\Omega$  be a  $\mathfrak{G}$ -set. A function  $f : \Omega \rightarrow Y$  is  **$\mathfrak{G}$ -invariant** if

$$\forall x \in \Omega, \forall g \in \mathfrak{G}, f(gx) = f(x).$$

- ▶  $f$  is  $\mathfrak{G}$ -invariant if its output is unaffected by the group action.

## Permutation invariant functions

Let  $\mathbf{X} \in \mathbb{R}^{n \times d}$ . The action of  $\sigma$  on  $\mathbf{X}$  is  $\sigma\mathbf{X} = (X_{\sigma^{-1}(i)j})_{ij}$ . Find a permutation invariant function  $F : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}$ .

- ▶ With  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top$  and  $F(\mathbf{X}) = \phi(\sum_{i=1}^n \psi(\mathbf{x}_i))$  with any  $\psi : \mathbb{R}^d \rightarrow Z, \phi : Z \rightarrow Y$ .
- ▶  $F(\mathbf{X}) = \text{rank}(\mathbf{X})$ .

# A formal definition of invariance

---

## Function operating on sets/multisets

Let  $\mathcal{X}$  be a **countable set**. By construction, any function acting on sets  $f : 2^{\mathcal{X}} \rightarrow Y$  for some  $Y$  is **permutation invariant**. That is

$$\forall \{x_1, \dots, x_n\} \in 2^{\mathcal{X}}, \forall \sigma \in S_n, f(\{x_1, \dots, x_n\}) = f(\{x_{\sigma^{-1}(1)}, \dots, x_{\sigma^{-1}(n)}\}).$$

Simply because  $\{x_1, \dots, x_n\} = \{x_{\sigma^{-1}(1)}, \dots, x_{\sigma^{-1}(n)}\}$ .

# A formal definition of invariance

---

## Function operating on sets/multisets

Let  $\mathcal{X}$  be a **countable set**. By construction, any function acting on sets  $f : 2^{\mathcal{X}} \rightarrow Y$  for some  $Y$  is **permutation invariant**. That is

$$\forall \{x_1, \dots, x_n\} \in 2^{\mathcal{X}}, \forall \sigma \in S_n, f(\{x_1, \dots, x_n\}) = f(\{x_{\sigma^{-1}(1)}, \dots, x_{\sigma^{-1}(n)}\}).$$

Simply because  $\{x_1, \dots, x_n\} = \{x_{\sigma^{-1}(1)}, \dots, x_{\sigma^{-1}(n)}\}$ .

- ▶ Any function  $f : 2^{\mathcal{X}} \rightarrow \mathbb{R}$  has the form ([Zaheer et al. 2018](#))

$$f(X) = \phi\left(\sum_{x \in X} \psi(x)\right) \text{ for some } \psi : \mathcal{X} \rightarrow \mathbb{R}, \phi : \mathbb{R} \rightarrow \mathbb{R}.$$

- ▶ See prev. course: a multiset is a “set” where element can be repeated several times e.g.  $\{\{a, a, b\}\}$ .
- ▶ Same representation result holds for functions on multisets ([Wagstaff et al. 2019](#)).

# A formal definition of equivariance

---

## Equivariance

Let  $\Omega_1, \Omega_2$  be two  $\mathfrak{G}$ -sets (of the same group). A function  $h : \Omega_1 \rightarrow \Omega_2$  is  **$\mathfrak{G}$ -equivariant** if

$$\forall x \in \Omega_1, \forall g \in \mathfrak{G}, h(gx) = gh(x).$$

- ▶ Pay attention to the input/output spaces and the compatibility.
- ▶ Transform the input + apply  $h =$  apply  $h$  and transform the result.

# A formal definition of equivariance

---

## Equivariance

Let  $\Omega_1, \Omega_2$  be two  $\mathfrak{G}$ -sets (of the same group). A function  $h : \Omega_1 \rightarrow \Omega_2$  is  **$\mathfrak{G}$ -equivariant** if

$$\forall x \in \Omega_1, \forall g \in \mathfrak{G}, h(gx) = gh(x).$$

- ▶ Pay attention to the input/output spaces and the compatibility.
- ▶ Transform the input + apply  $h$  = apply  $h$  and transform the result.

## Convolutions

Prove that the convolution with a filter  $h \in L_2(\mathbb{R})$  is translation equivariant.

# A formal definition of equivariance

---

## Equivariance

Let  $\Omega_1, \Omega_2$  be two  $\mathfrak{G}$ -sets (of the same group). A function  $h : \Omega_1 \rightarrow \Omega_2$  is  **$\mathfrak{G}$ -equivariant** if

$$\forall x \in \Omega_1, \forall g \in \mathfrak{G}, h(gx) = gh(x).$$

- ▶ Pay attention to the input/output spaces and the compatibility.
- ▶ Transform the input + apply  $h$  = apply  $h$  and transform the result.

## Convolutions

Consider a filter  $h \in L_2(\mathbb{R})$ .

- ▶ The convolution with a filter is  $H : \Omega = L_2(\mathbb{R}) \rightarrow L_2(\mathbb{R})$  such that  $H(g) := g * h = h * g$ .
- ▶ For any translation  $\tau_x$

$$\forall g \in L_2(\mathbb{R}), H(\tau_x g) = (\tau_x g) * h = \tau_x(g * h) = \tau_x H(g).$$

- ▶ Translate then convolve = convolve then translate.

# A formal definition of equivariance

---

## Equivariance

Let  $\Omega_1, \Omega_2$  be two  $\mathfrak{G}$ -sets (of the same group). A function  $h : \Omega_1 \rightarrow \Omega_2$  is  **$\mathfrak{G}$ -equivariant** if

$$\forall x \in \Omega_1, \forall g \in \mathfrak{G}, h(gx) = gh(x).$$

- ▶ Pay attention to the input/output spaces and the compatibility.
- ▶ Transform the input + apply  $h$  = apply  $h$  and transform the result.

## Permutation equivariant functions

- ▶ Find two permutation equivariant functions  $F : \mathbb{R}^{n \times d_1} \rightarrow \mathbb{R}^{n \times d_2}$ .

# A formal definition of equivariance

## Equivariance

Let  $\Omega_1, \Omega_2$  be two  $\mathfrak{G}$ -sets (of the same group). A function  $h : \Omega_1 \rightarrow \Omega_2$  is  **$\mathfrak{G}$ -equivariant** if

$$\forall x \in \Omega_1, \forall g \in \mathfrak{G}, h(gx) = gh(x).$$

- ▶ Pay attention to the input/output spaces and the compatibility.
- ▶ Transform the input + apply  $h$  = apply  $h$  and transform the result.

## Permutation equivariant functions

- ▶ Let  $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$  and  $F(\mathbf{X}) = \mathbf{XW}$ .

$$\text{▶ Let } \mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} \text{ previous example } F(\mathbf{X}) = \begin{pmatrix} (\mathbf{W}^\top \mathbf{x}_1)^\top \\ \vdots \\ (\mathbf{W}^\top \mathbf{x}_n)^\top \end{pmatrix}.$$

$$\text{▶ More generally } \Psi(\mathbf{X}) = \begin{pmatrix} \psi(\mathbf{x}_1)^\top \\ \vdots \\ \psi(\mathbf{x}_n)^\top \end{pmatrix} \text{ where } \psi : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}.$$

# A formal definition of equivariance

---

## Equivariance

Let  $\Omega_1, \Omega_2$  be two  $\mathfrak{G}$ -sets (of the same group). A function  $h : \Omega_1 \rightarrow \Omega_2$  is  **$\mathfrak{G}$ -equivariant** if

$$\forall x \in \Omega_1, \forall g \in \mathfrak{G}, h(gx) = gh(x).$$

- ▶ Pay attention to the input/output spaces and the compatibility.
- ▶ Transform the input + apply  $h$  = apply  $h$  and transform the result.

## Laplacian matrix

- ▶ An action of  $S_n$  on  $\mathbb{R}^{n \times n}$  is defined as

$$\sigma \mathbf{A} = (A_{\sigma^{-1}(i), \sigma^{-1}(j)})_{ij}$$

- ▶  $\mathcal{L} : \text{sym}_n(\mathbb{R}) \rightarrow \text{sym}_n(\mathbb{R})$  which takes a symmetric matrix  $\mathbf{A}$  and outputs the Laplacian matrix  $\mathcal{L}(\mathbf{A}) = \text{diag}(\mathbf{A}\mathbf{1}) - \mathbf{A}$
- ▶ **Show that  $\mathcal{L}$  is  $S_n$ -permutation equivariant.**

# Combining them together

---

## Composition of invariant/equivariant functions

Let  $\Omega_1, \Omega_2$  be  $\mathfrak{G}$ -sets.

- ▶ Let  $f : \Omega_1 \rightarrow \Omega_2$  be a  **$\mathfrak{G}$ -equivariant** function.
- ▶ Let  $g : \Omega_2 \rightarrow Y$  be a  **$\mathfrak{G}$ -invariant** function.

Then  $h = g \circ f$  is  **$\mathfrak{G}$ -invariant**.

# Combining them together

## Composition of invariant/equivariant functions

Let  $\Omega_1, \Omega_2$  be  $\mathfrak{G}$ -sets.

- ▶ Let  $f : \Omega_1 \rightarrow \Omega_2$  be a  **$\mathfrak{G}$ -equivariant** function.
- ▶ Let  $g : \Omega_2 \rightarrow Y$  be a  **$\mathfrak{G}$ -invariant** function.

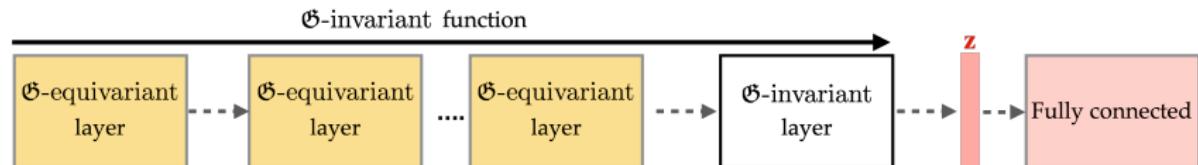
Then  $h = g \circ f$  is  **$\mathfrak{G}$ -invariant**.

## Proof

Indeed with  $x \in \Omega_1, g \in \mathfrak{G}$

$$h(gx) = g(f(gx)) = g(gf(x)) = g(f(x)) = (g \circ f)(x) = h(x).$$

Simple but powerful: one of the reason CNNs work so well



# Combining them together

## Composition of invariant/equivariant functions

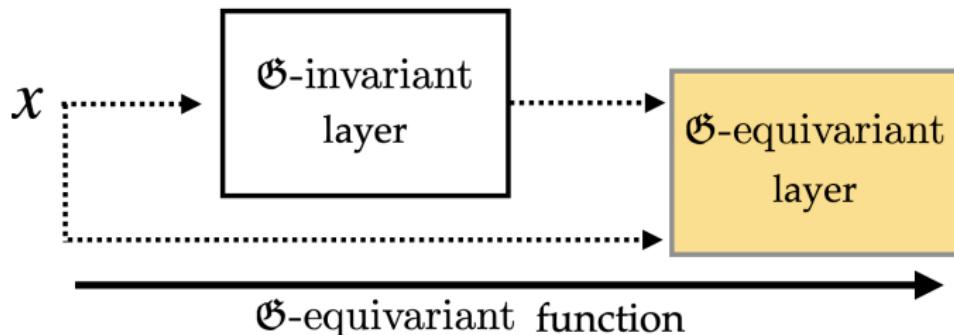
Let  $\Omega_1, \Omega_2$  be  $\mathfrak{G}$ -sets.

- ▶ Let  $f : \Omega_1 \times Y \rightarrow \Omega_2$  be a  **$\mathfrak{G}$ -equivariant** function **with respect to its first variable** i.e.  $\forall y \in \Omega_1, \forall g \in \mathfrak{G}, \forall y \in Y, f(gx, y) = gf(gx, y)$ .
- ▶ Let  $g : \Omega_1 \rightarrow Y$  be a  **$\mathfrak{G}$ -invariant** function.

Then the function  $h$  defined by  $h(x) = f(x, g(x))$  is  **$\mathfrak{G}$ -equivariant**.

## Proof

$$h(gx) = f(gx, g(gx)) = f(gx, g(x)) = gf(x, g(x)) = gh(x).$$



# Focus on permutation invariance/equivariance

---

## Permutations as matrices

- ▶  $\sigma \in S_n$  can be described as  $\mathbf{P}_\sigma = \begin{pmatrix} \mathbf{e}_{\sigma(1)}^\top \\ \vdots \\ \mathbf{e}_{\sigma(n)}^\top \end{pmatrix} \in \{0, 1\}^{n \times n}$ .  $\mathbf{P}_{\sigma^{-1}} = \mathbf{P}_\sigma^\top$ .
- ▶ For  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , the previous action is  $\sigma \mathbf{A} = (A_{\sigma^{-1}(i)\sigma^{-1}(j)})_{ij} = \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma$ .
- ▶ An action of  $S_n$  on  $\mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n}$

$$\sigma(\mathbf{X}, \mathbf{A}) = (\mathbf{P}_\sigma^\top \mathbf{X}, \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma)$$

# Focus on permutation invariance/equivariance

Permutations as matrices

- ▶  $\sigma \in S_n$  can be described as  $\mathbf{P}_\sigma = \begin{pmatrix} \mathbf{e}_{\sigma(1)}^\top \\ \vdots \\ \mathbf{e}_{\sigma(n)}^\top \end{pmatrix} \in \{0, 1\}^{n \times n}$ .  $\mathbf{P}_{\sigma^{-1}} = \mathbf{P}_\sigma^\top$ .
- ▶ For  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , the previous action is  $\sigma \mathbf{A} = (A_{\sigma^{-1}(i)\sigma^{-1}(j)})_{ij} = \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma$ .
- ▶ An action of  $S_n$  on  $\mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n}$

$$\sigma(\mathbf{X}, \mathbf{A}) = (\mathbf{P}_\sigma^\top \mathbf{X}, \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma)$$

Interpretation

- ▶  $\sigma(\mathbf{X}, \mathbf{A})$  permutes the nodes of the graph **and** the features **in the same manner**.

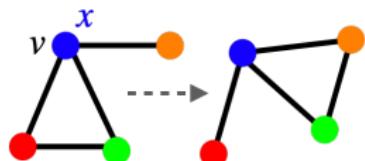


Figure: Is it a valid action of  $\sigma$ ?

# Focus on permutation invariance/equivariance

---

Permutations as matrices

- ▶  $\sigma \in S_n$  can be described as  $\mathbf{P}_\sigma = \begin{pmatrix} \mathbf{e}_{\sigma(1)}^\top \\ \vdots \\ \mathbf{e}_{\sigma(n)}^\top \end{pmatrix} \in \{0, 1\}^{n \times n}$ .  $\mathbf{P}_{\sigma^{-1}} = \mathbf{P}_\sigma^\top$ .
- ▶ For  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , the previous action is  $\sigma \mathbf{A} = (A_{\sigma^{-1}(i)\sigma^{-1}(j)})_{ij} = \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma$ .
- ▶ An action of  $S_n$  on  $\mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n}$

$$\sigma(\mathbf{X}, \mathbf{A}) = (\mathbf{P}_\sigma^\top \mathbf{X}, \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma)$$

Back to the GNN context

- ▶ In classification/regression  $f : G = (\mathbf{X}, \mathbf{A}) \rightarrow y \in Y$  (e.g.  $(\{+1, -1\})$ ).
- ▶ For node embeddings  $F : G = (\mathbf{X}, \mathbf{A}) \rightarrow \mathbf{Z} \in \mathbb{R}^{n \times k}$

# Focus on permutation invariance/equivariance

---

Permutations as matrices

- ▶  $\sigma \in S_n$  can be described as  $\mathbf{P}_\sigma = \begin{pmatrix} \mathbf{e}_{\sigma(1)}^\top \\ \vdots \\ \mathbf{e}_{\sigma(n)}^\top \end{pmatrix} \in \{0, 1\}^{n \times n}$ .  $\mathbf{P}_{\sigma^{-1}} = \mathbf{P}_\sigma^\top$ .
- ▶ For  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , the previous action is  $\sigma \mathbf{A} = (A_{\sigma^{-1}(i)\sigma^{-1}(j)})_{ij} = \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma$ .
- ▶ An action of  $S_n$  on  $\mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n}$

$$\sigma(\mathbf{X}, \mathbf{A}) = (\mathbf{P}_\sigma^\top \mathbf{X}, \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma)$$

Back to the GNN context

- ▶ In classification/regression  $f : G = (\mathbf{X}, \mathbf{A}) \rightarrow y \in Y$  (e.g.  $(\{+1, -1\})$ ).
- ▶ For node embeddings  $F : G = (\mathbf{X}, \mathbf{A}) \rightarrow \mathbf{Z} \in \mathbb{R}^{n \times k}$

Ensuring invariance/equivariance is key when learning on graphs

Find  $f$  that are  $S_n$ -**invariant**,  $F$  that are  $S_n$ -**equivariant**.

- ▶  $f(\mathbf{P}_\sigma^\top \mathbf{X}, \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma) = f(\mathbf{X}, \mathbf{A})$  and  $F(\mathbf{P}_\sigma^\top \mathbf{X}, \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma) = \mathbf{P}_\sigma^\top F(\mathbf{X}, \mathbf{A})$ .

# Focus on permutation invariance/equivariance

---

Ensuring invariance/equivariance is key when learning on graphs

Find  $f$  that are  $S_n$ -**invariant**,  $F$  that are  $S_n$ -**equivariant**.

- ▶  $f(\mathbf{P}_\sigma^\top \mathbf{X}, \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma) = f(\mathbf{X}, \mathbf{A})$  and  $F(\mathbf{P}_\sigma^\top \mathbf{X}, \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma) = \mathbf{P}_\sigma^\top F(\mathbf{X}, \mathbf{A})$ .

Examples: equivariance (1/2)

- ▶ Take  $\mathbf{X} \in \mathbb{R}^{n \times d_1}$ ,  $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$  and a function  $\Psi$  that applies **independently on each row** of a matrix.
- ▶  $F(\mathbf{X}, \mathbf{A}) = \Psi(\mathbf{A} \mathbf{X} \mathbf{W})$  is  $S_n$ -**equivariant**.

# Focus on permutation invariance/equivariance

---

Ensuring invariance/equivariance is key when learning on graphs

Find  $f$  that are  $S_n$ -**invariant**,  $F$  that are  $S_n$ -**equivariant**.

- ▶  $f(\mathbf{P}_\sigma^\top \mathbf{X}, \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma) = f(\mathbf{X}, \mathbf{A})$  and  $F(\mathbf{P}_\sigma^\top \mathbf{X}, \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma) = \mathbf{P}_\sigma^\top F(\mathbf{X}, \mathbf{A})$ .

Examples: equivariance (1/2)

- ▶ Take  $\mathbf{X} \in \mathbb{R}^{n \times d_1}$ ,  $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$  and a function  $\Psi$  that applies **independently on each row** of a matrix.
- ▶  $F(\mathbf{X}, \mathbf{A}) = \Psi(\mathbf{A}\mathbf{X}\mathbf{W})$  is  $S_n$ -**equivariant**.
- ▶ In particular when  $\Psi$  is **element-wise**.

# Focus on permutation invariance/equivariance

---

Ensuring invariance/equivariance is key when learning on graphs

Find  $f$  that are  $S_n$ -**invariant**,  $F$  that are  $S_n$ -**equivariant**.

- ▶  $f(\mathbf{P}_\sigma^\top \mathbf{X}, \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma) = f(\mathbf{X}, \mathbf{A})$  and  $F(\mathbf{P}_\sigma^\top \mathbf{X}, \mathbf{P}_\sigma^\top \mathbf{A} \mathbf{P}_\sigma) = \mathbf{P}_\sigma^\top F(\mathbf{X}, \mathbf{A})$ .

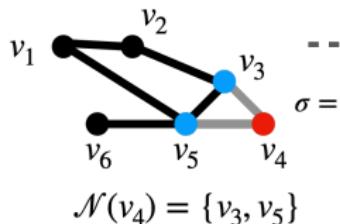
Examples: equivariance (1/2)

- ▶ Take  $\mathbf{X} \in \mathbb{R}^{n \times d_1}$ ,  $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$  and a function  $\Psi$  that applies **independently on each row** of a matrix.
- ▶  $F(\mathbf{X}, \mathbf{A}) = \Psi(\mathbf{A}\mathbf{X}\mathbf{W})$  is  $S_n$ -**equivariant**.
- ▶ In particular when  $\Psi$  is **element-wise**.
- ▶ But also  $F(\mathbf{X}, \mathbf{A}) = \Psi(G(\mathbf{A})\mathbf{X}\mathbf{W})$  where  $G$  is  $S_n$ -**equivariant**.
- ▶ E.g.  $F(\mathbf{X}, \mathbf{A}) = \Psi(\mathcal{L}(\mathbf{A})\mathbf{X}\mathbf{W})$  where  $\mathcal{L}$  computes the Laplacian.
- ▶ E.g.  $F(\mathbf{X}, \mathbf{A}) = \Psi(P[\mathcal{L}](\mathbf{A})\mathbf{X}\mathbf{W})$  where  $P$  is a polynomial  $P[\mathcal{L}] = \sum_m c_m \mathcal{L}^m$ .

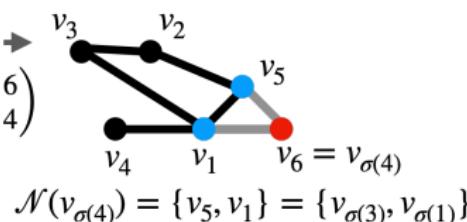
# Focus on permutation invariance/equivariance

Examples: equivariance (2/2)

- Take  $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix}$  and define the multiset  $X_i := \{\{\mathbf{x}_j : j \in \mathcal{N}(i)\}\}$ .
- Then  $X_{\sigma(i)} = \{\{\mathbf{x}_{\sigma(j)} : j \in \mathcal{N}(i)\}\}$ :



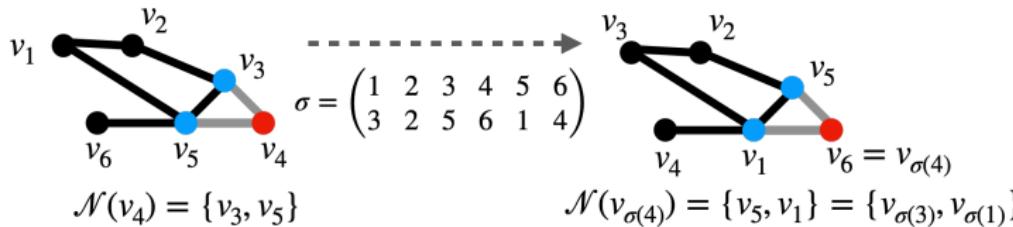
$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 2 & 5 & 6 & 1 & 4 \end{pmatrix}$$



# Focus on permutation invariance/equivariance

Examples: equivariance (2/2)

- Take  $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix}$  and define the multiset  $X_i := \{\{\mathbf{x}_j : j \in \mathcal{N}(i)\}\}$ .
- Then  $X_{\sigma(i)} = \{\{\mathbf{x}_{\sigma(j)} : j \in \mathcal{N}(i)\}\}$ :



- A function AGGREGATE operating on **multisets of vectors**.
- Then the following function is permutation equivariant.

$$F(\mathbf{X}, \mathbf{A}) = \begin{pmatrix} \psi(\mathbf{x}_1, \text{AGGREGATE}(X_1)) \\ \vdots \\ \psi(\mathbf{x}_n, \text{AGGREGATE}(X_n)) \end{pmatrix}$$

# Table of contents

---

From neural networks...

The basic ideas

Logistic regression and one layer neural-network

Convolutional neural networks

... to unsupervised node embeddings techniques...

A chronological start

... to graph neural networks

Learning with graphs

What is a GNN ?

A bit of group theory

Invariance and equivariance

**Message-passing neural networks**

Examples of GNN

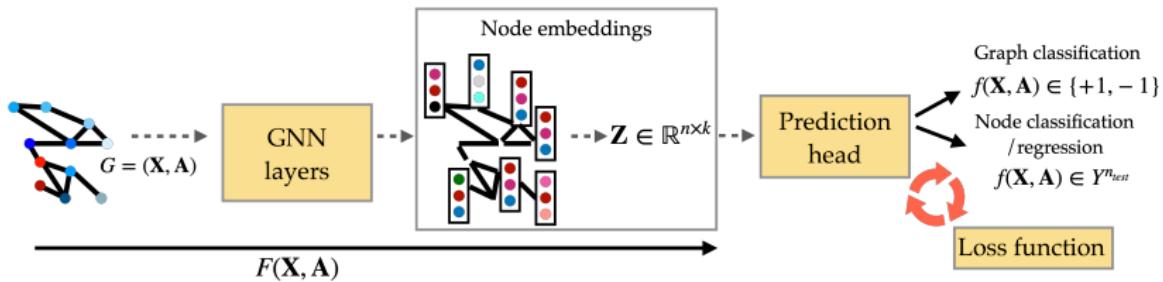
The whole pipeline

Expressivity of GNN

Conclusion

# Remember

## The training pipeline

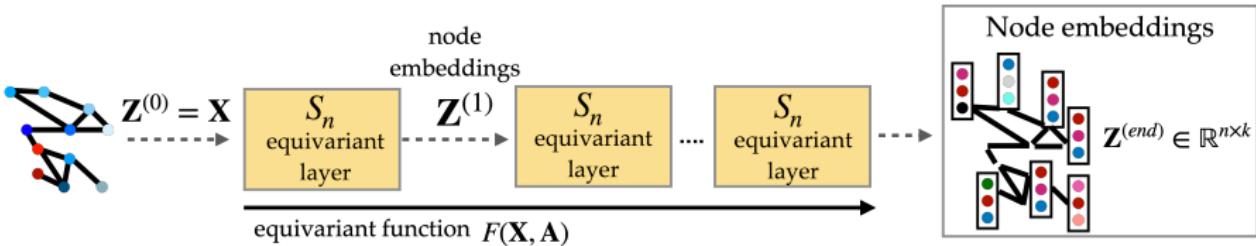


- ▶ Overall the same procedure: find an embedding of the nodes  $F(\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{n \times k}$  (supervised or unsupervised) and then do stuff.

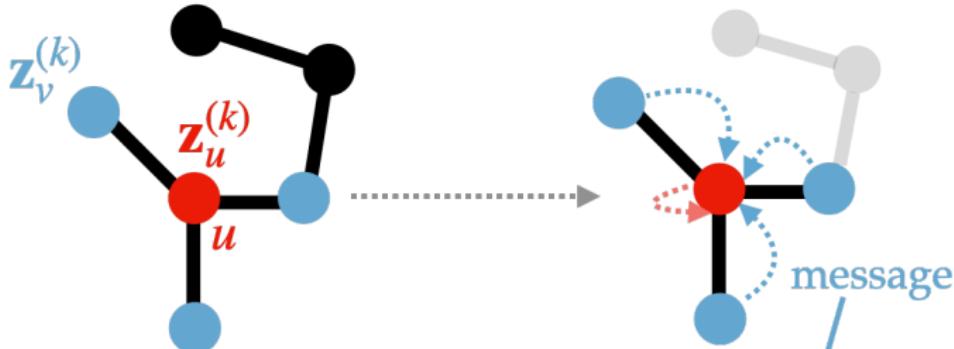
# Message-passing for node embeddings

Goal of the message passing framework

- ▶ Defines specific  $S_n$ -equivariant layers/functions.
- ▶ Can be used for node embeddings.
- ▶ Usually  $\mathbf{Z}^{(0)} = \mathbf{X}$  but when no node features are available several options (e.g. node statistics).
- ▶ Notation:  $\mathbf{z}_u^{(k)}$  is the embedding of the node  $u \in V$  at the  $k$ -layer.



# The message passing framework



$$\mathbf{z}_u^{(k+1)} = \text{COMBINE}^{(k)} \left( \mathbf{z}_u^{(k)}, \text{AGGREGATE}^{(k)} \left( \{\{\mathbf{z}_v^{(k)} : v \in \mathcal{N}(u)\}\} \right) \right)$$

One of the most used GNN framework in practice

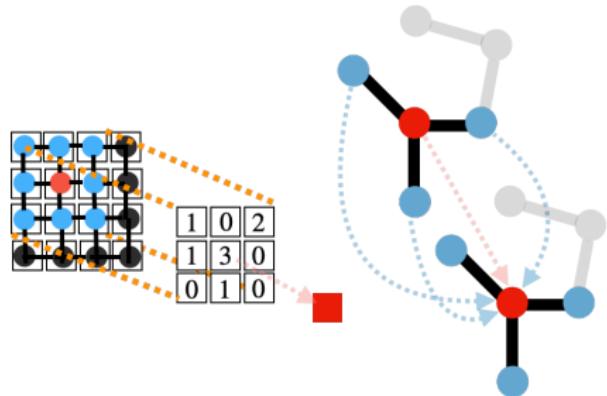
- ▶ At each iteration, **every node aggregates information from its local neighborhood.**
- ▶ A zoo of methods for different COMBINE, AGGREGATE functions.
- ▶ Why is this defining a permutation equivariant layer ?

# The message passing framework

## Similarities with CNN

- ▶ One layer of message-passing GNN shares similarities to convolutional layers.
- ▶ Usually it takes the form

$$\mathbf{z}_u^{(k+1)} = \phi \left( \sum_{v \in \mathcal{N}(u) \cup \{u\}} \alpha_{uv} \mathbf{z}_v^{(k)} \right).$$

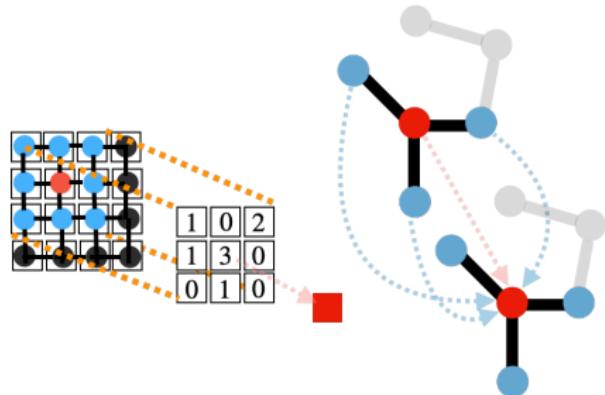


# The message passing framework

## Similarities with CNN

- ▶ One layer of message-passing GNN shares similarities to convolutional layers.
- ▶ Usually it takes the form

$$\mathbf{z}_u^{(k+1)} = \phi \left( \sum_{v \in \mathcal{N}(u) \cup \{u\}} \alpha_{uv} \mathbf{z}_v^{(k)} \right).$$



## k-hop neighbourhood

After  $k$ -steps each node has received the informations from its  $k$ -hop neighbourhood.

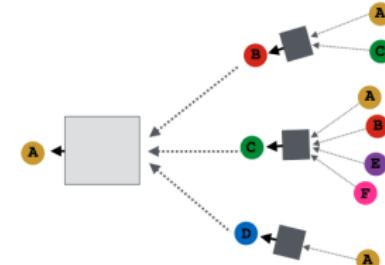
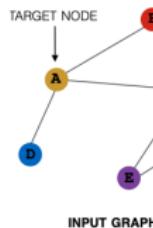


Figure: From Jure Leskovec course *Machine Learning with Graphs*.

# Table of contents

---

From neural networks...

The basic ideas

Logistic regression and one layer neural-network

Convolutional neural networks

... to unsupervised node embeddings techniques...

A chronological start

... to graph neural networks

Learning with graphs

What is a GNN ?

A bit of group theory

Invariance and equivariance

Message-passing neural networks

**Examples of GNN**

The whole pipeline

Expressivity of GNN

Conclusion

# A first GNN with message passing

---

Sum/mean aggregation (Scarselli et al. 2008)

A first idea would be

$$\mathbf{z}_u^{(k+1)} = \phi(\mathbf{W}_{\text{self}}^{(k)} \mathbf{z}_u^{(k)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{v \in \mathcal{N}(u)} \mathbf{z}_v^{(k)} + \mathbf{b}^{(k)})$$

- ▶  $\mathbf{W}_{\text{self}}^{(k)}, \mathbf{W}_{\text{neigh}}^{(k)} \in \mathbb{R}^{d_{k+1} \times d_k}$  are matrices of **learnable parameters**.
- ▶ Do not depend on the number of nodes ! .
- ▶ Complexity of computing it for all nodes is  $O(|E|)$ .
- ▶  $\mathbf{b}^{(k)} \in \mathbb{R}^{d_{k+1}}$  is a bias term (often omitted to simplify notations).
- ▶  $\phi$  is a pointwise non-linearity such as ReLu.

## Questions

- ▶ What is COMBINE, AGGREGATE ?
- ▶ Write this in matrix form.

# A first GNN with message passing

Sum/mean aggregation (Scarselli et al. 2008)

A first idea would be

$$\mathbf{z}_u^{(k+1)} = \phi(\mathbf{W}_{\text{self}}^{(k)} \mathbf{z}_u^{(k)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{v \in \mathcal{N}(u)} \mathbf{z}_v^{(k)} + \mathbf{b}^{(k)})$$

- ▶  $\mathbf{W}_{\text{self}}^{(k)}, \mathbf{W}_{\text{neigh}}^{(k)} \in \mathbb{R}^{d_{k+1} \times d_k}$  are matrices of **learnable parameters**.
- ▶ Do not depend on the number of nodes ! .
- ▶ Complexity of computing it for all nodes is  $O(|E|)$ .
- ▶  $\mathbf{b}^{(k)} \in \mathbb{R}^{d_{k+1}}$  is a bias term (often omitted to simplify notations).
- ▶  $\phi$  is a pointwise non-linearity such as ReLu.

## Answers

- ▶ What is COMBINE, AGGREGATE ?
- ▶  $\forall k, \text{AGGREGATE}^{(k)}(\{\{\mathbf{z}_v : v \in \mathcal{N}(u)\}\}) = \sum_{v \in \mathcal{N}(u)} \mathbf{z}_v$ .
- ▶  $\text{COMBINE}^{(k)}(\mathbf{z}_1, \mathbf{z}_2) = \mathbf{W}_{\text{self}}^{(k)} \mathbf{z}_1 + \mathbf{W}_{\text{neigh}}^{(k)} \mathbf{z}_2 + \mathbf{b}^{(k)}$ .

# A first GNN with message passing

Sum/mean aggregation (Scarselli et al. 2008)

A first idea would be

$$\mathbf{z}_u^{(k+1)} = \phi(\mathbf{W}_{\text{self}}^{(k)} \mathbf{z}_u^{(k)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{v \in \mathcal{N}(u)} \mathbf{z}_v^{(k)} + \mathbf{b}^{(k)})$$

- ▶  $\mathbf{W}_{\text{self}}^{(k)}, \mathbf{W}_{\text{neigh}}^{(k)} \in \mathbb{R}^{d_{k+1} \times d_k}$  are matrices of **learnable parameters**.
- ▶ Do not depend on the number of nodes ! .
- ▶ Complexity of computing it for all nodes is  $O(|E|)$ .
- ▶  $\mathbf{b}^{(k)} \in \mathbb{R}^{d_{k+1}}$  is a bias term (often omitted to simplify notations).
- ▶  $\phi$  is a pointwise non-linearity such as ReLu.

## Answers

- ▶ Write this in matrix form.

$$\mathbf{Z}^{(k+1)} = \phi \left( \mathbf{A} \mathbf{Z}^{(k)} \mathbf{W}_{\text{neigh}}^{(k)} + \mathbf{Z}^{(k)} \mathbf{W}_{\text{self}}^{(k)} + \begin{pmatrix} \mathbf{b}^{(k)} \\ \vdots \\ \mathbf{b}^{(k)} \end{pmatrix} \right).$$

# Graph convolutional neural networks

## Most popular baseline model

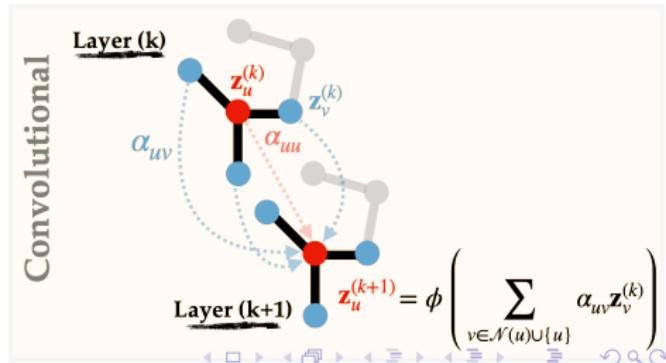
Introduced by Kipf and Welling 2016 for semi-supervised node classification.

$$\mathbf{z}_u^{(k+1)} = \text{Relu}(\mathbf{W}_{\text{self}}^{(k)} \mathbf{z}_u^{(k)} + \mathbf{W}_{\text{neigh}}^{(k)} \frac{1}{\sqrt{|\mathcal{N}(u)|}} \sum_{v \in \mathcal{N}(u)} \frac{\mathbf{z}_v^{(k)}}{\sqrt{|\mathcal{N}(v)|}})$$

- ▶ Also GraphSage framework (William L. Hamilton, R. Ying, and Leskovec 2018).
- ▶ What is COMBINE, AGGREGATE ?

## In matrix form

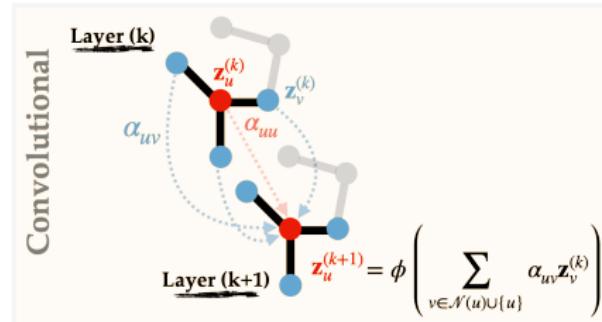
- ▶ With  $\mathbf{W}_{\text{self}} = \mathbf{W}_{\text{neigh}}$ ,  $\mathbf{Z}^{(k+1)} = \text{Relu} \left( (\mathbf{I} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}) \mathbf{Z}^{(k)} \mathbf{W}^{(k)} \right)$ .
- ▶ First-order approximation of localized spectral filters on graphs.



# Graph Attention Networks

## Motivations

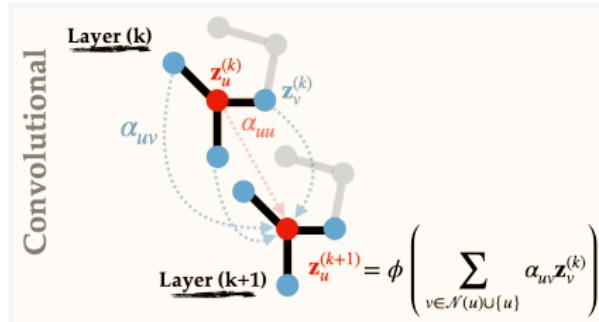
- ▶ In many MP-GNN layers weights of the convolutions **are fixed**.
- ▶ What if we also learn them ?
- ▶ Learn the importance of the neighbours contributions.



# Graph Attention Networks

## Motivations

- ▶ In many MP-GNN layers weights of the convolutions **are fixed**.
- ▶ What if we also learn them ?
- ▶ Learn the importance of the neighbours contributions.



## GAT networks (Veličković et al. 2017)

$$\mathbf{z}_u^{(k+1)} = \text{Relu}(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \alpha_{uv} \mathbf{z}_v^{(k)})$$

- ▶ Here  $\alpha_{uv}$  are **learnable weights**.
- ▶  $e_{uv} = \text{NN}(\Theta_1 \mathbf{z}_u, \Theta_2 \mathbf{z}_u)$  with learnable matrices  $\Theta_1, \Theta_2$  and

$$\alpha_{uv} = \text{softmax}_v(e_{uv}) = \frac{\exp(e_{uv})}{\sum_{v' \in \mathcal{N}(u)} e_{uv'}}$$

- ▶ It is based on attention mechanisms (Vaswani et al. 2023).

# Graph Isomorphism Networks (GIN)

---

## The problem of injectivity

Xu et al. 2019 provide a detailed discussion of the relative power of GNN.

- ▶ One interesting property is **injectivity** of COMBINE, AGGREGATE.
- ▶ They propose

$$\mathbf{z}_u^{(k+1)} = \text{MLP}^{(k)} \left( (1 + \theta^{(k)}) \mathbf{z}_u^{(k)} + \sum_{v \in \mathcal{N}(u)} \mathbf{z}_v^{(k)} \right)$$

- ▶ MLP :  $\mathbb{R}^{d_k} \rightarrow \mathbb{R}^{d_{k+1}}$  is a fully connected neural-network.

# Spectral GNN

---

## Learning filters

Originally introduced by Bruna et al. 2013. The idea is

$$\mathbf{Z}^{(k+1)} = \text{Relu}(P[\mathcal{L}](\mathbf{A})\mathbf{Z}^{(k)}\mathbf{W}^{(k)})$$

- ▶  $\mathcal{L}(\mathbf{A}) = \text{diag}(\mathbf{A}\mathbf{1}) - \mathbf{A}$  is the Laplacian (or normalized version).
- ▶  $P[\mathcal{L}] = \sum_{m=0}^M c_m \mathcal{L}^m$  is a **learnable** polynomial of the Laplacian.
- ▶ As  $\mathcal{L}(\mathbf{A}) = \mathbf{U}\Lambda\mathbf{U}^\top$ ,  $P[\mathcal{L}](\mathbf{A}) = \mathbf{U}P[\Lambda]\mathbf{U}^\top$ .
- ▶ Connections with the Fourier transform on graphs:  $P[\mathcal{L}]$  acts as a filter.

# Spectral GNN

---

## Learning filters

Originally introduced by Bruna et al. 2013. The idea is

$$\mathbf{Z}^{(k+1)} = \text{Relu}(P[\mathcal{L}](\mathbf{A})\mathbf{Z}^{(k)}\mathbf{W}^{(k)})$$

- ▶  $\mathcal{L}(\mathbf{A}) = \text{diag}(\mathbf{A}\mathbf{1}) - \mathbf{A}$  is the Laplacian (or normalized version).
- ▶  $P[\mathcal{L}] = \sum_{m=0}^M c_m \mathcal{L}^m$  is a **learnable** polynomial of the Laplacian.
- ▶ As  $\mathcal{L}(\mathbf{A}) = \mathbf{U}\Lambda\mathbf{U}^\top$ ,  $P[\mathcal{L}](\mathbf{A}) = \mathbf{U}P[\Lambda]\mathbf{U}^\top$ .
- ▶ Connections with the Fourier transform on graphs:  $P[\mathcal{L}]$  acts as a filter.

## Limitations

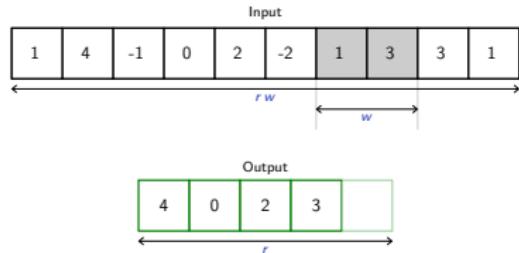
- ▶ Naive complexity in  $O(|V|^3)$  (eigen-decomposition).
- ▶ Any perturbation to a graph results in a change of eigenbasis  $\mathbf{U}$ .
- ▶ Learned filters are domain dependent.
- ▶ Alternative ChebNet [Defferrard, Bresson, and Vandergheynst 2017](#) relies on Chebyshev polynomials with  $O(|E|M)$  complexity.

# Graph pooling

## Pooling layers in neural networks

At the core of many NN architectures.

- ▶ Most standard type is **max-pooling**.
- ▶ ↓ the number of parameters to learn.
- ▶ Improves robustness.

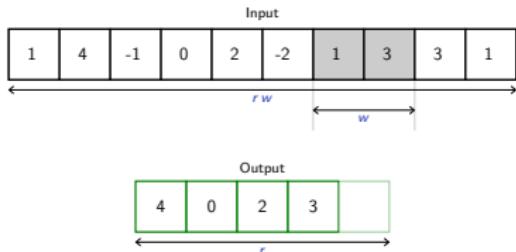


# Graph pooling

## Pooling layers in neural networks

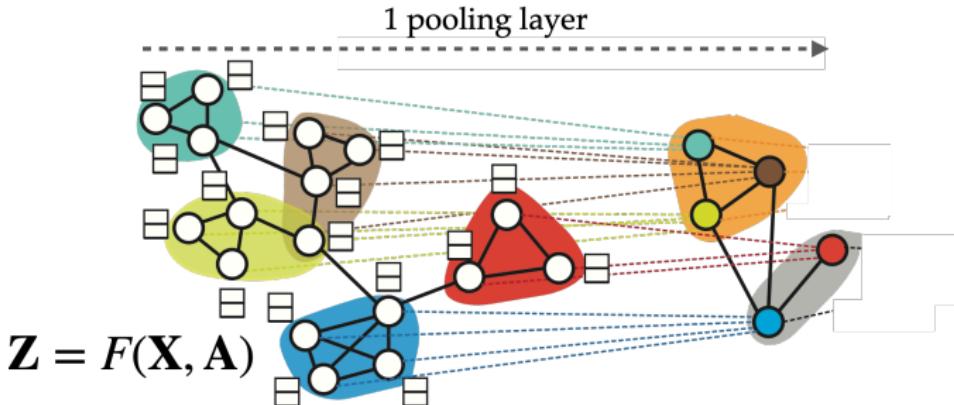
At the core of many NN architectures.

- ▶ Most standard type is **max-pooling**.
- ▶ ↓ the number of parameters to learn.
- ▶ Improves robustness.



## Pooling in GNN

Equivalent to down-sampling = reducing the number of nodes.



## Learning at the graph level

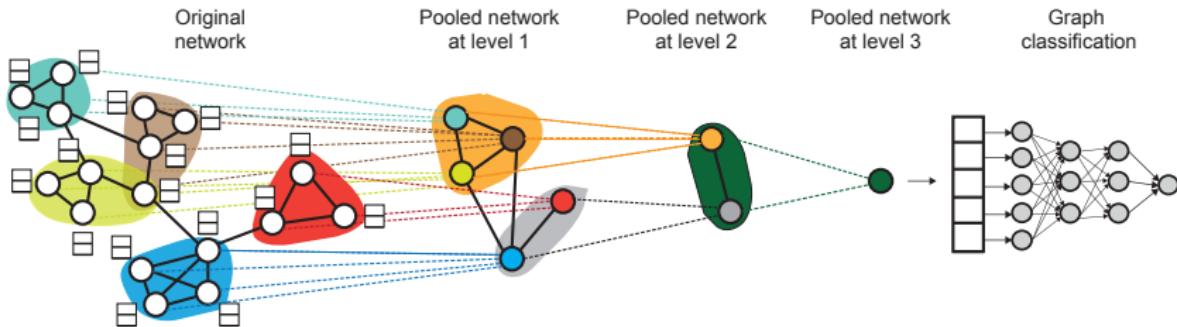
- ▶ The neural message passing approach produces a set of node embeddings  $F(\mathbf{X}, \mathbf{A}) = \mathbf{Z} \in \mathbb{R}^{n \times k}$ .
- ▶ What about predictions at the graph level ? E.g. in graph classification.
- ▶ **We want one embedding for the entire graph  $\mathbf{z}_G$ .**
- ▶ It should be a permutation invariant function  $f(\mathbf{X}, \mathbf{A})$ .
- ▶ E.g. global average pooling  $\mathbf{z}_G = f(\mathbf{X}, \mathbf{A}) = \frac{1}{|V|} \sum_{u \in V} \mathbf{z}_u \in \mathbb{R}^k$ .

# Diffpool

## Learning at the graph level

- ▶ The neural message passing approach produces a set of node embeddings  $F(\mathbf{X}, \mathbf{A}) = \mathbf{Z} \in \mathbb{R}^{n \times k}$ .
- ▶ What about predictions at the graph level ? E.g. in graph classification.
- ▶ **We want one embedding for the entire graph  $\mathbf{z}_G$ .**
- ▶ It should be a permutation invariant function  $f(\mathbf{X}, \mathbf{A})$ .
- ▶ E.g. global average pooling  $\mathbf{z}_G = f(\mathbf{X}, \mathbf{A}) = \frac{1}{|V|} \sum_{u \in V} \mathbf{z}_u \in \mathbb{R}^k$ .

Better idea: hierarchical pooling (Z. Ying et al. 2018)



# Table of contents

---

From neural networks...

The basic ideas

Logistic regression and one layer neural-network

Convolutional neural networks

... to unsupervised node embeddings techniques...

A chronological start

... to graph neural networks

Learning with graphs

What is a GNN ?

A bit of group theory

Invariance and equivariance

Message-passing neural networks

Examples of GNN

**The whole pipeline**

Expressivity of GNN

Conclusion

# Applications

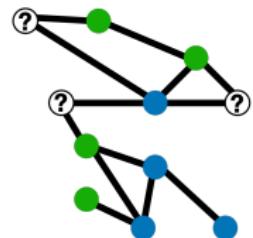
---

## Node classification

- ▶ **One graph**  $G$  where each node has a class.

Train GNNs in a fully-supervised manner by minimizing

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



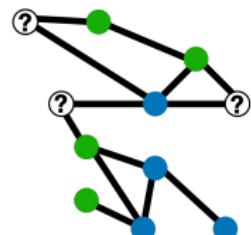
# Applications

## Node classification

- **One graph**  $G$  where each node has a class.

Train GNNs in a fully-supervised manner by minimizing

$$\mathcal{L} = \sum_{u \in V_{train}} -\log(\text{softmax}(\mathbf{z}_u, y_u))$$

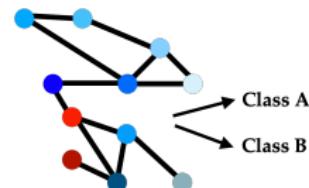


## Graph classification

- **Many graphs**  $G_1, \dots, G_n$  associated with classes  $(y_{G_i})_i$ .

Train GNNs in a fully-supervised manner by minimizing

$$\mathcal{L} = \sum_{G \in T_{train}} \ell(\text{MLP}(\mathbf{z}_G), y_G)$$



# Table of contents

---

From neural networks...

The basic ideas

Logistic regression and one layer neural-network

Convolutional neural networks

... to unsupervised node embeddings techniques...

A chronological start

**... to graph neural networks**

Learning with graphs

What is a GNN ?

A bit of group theory

Invariance and equivariance

Message-passing neural networks

Examples of GNN

The whole pipeline

**Expressivity of GNN**

Conclusion

# Connection with the WL test

---

## WL algorithm and MP-GNN

- ▶ WL algorithm and the message passing GNN approach are very similar.
- ▶ Iteratively aggregate information from local node neighborhoods.

# Connection with the WL test

---

## WL algorithm and MP-GNN

- ▶ WL algorithm and the message passing GNN approach are very similar.
- ▶ Iteratively aggregate information from local node neighborhoods.

Message passing neural networks are not that powerful ?

- ▶ Consider a MP-GNN with  $K$  layers

$$\mathbf{z}_u^{(k+1)} = \text{COMBINE}^{(k)} \left( \mathbf{z}_u^{(k)}, \text{AGGREGATE}^{(k)} \left( \{\{\mathbf{z}_v^{(k)} : v \in \mathcal{N}(u)\}\} \right) \right)$$

- ▶ Suppose that discrete node labels  $\mathbf{Z}^{(0)} = \mathbf{X} \in \mathbb{Z}^{n \times d}$ .
- ▶ Then Xu et al. 2019 show that

$\mathbf{z}_u^{(K)} \neq \mathbf{z}_v^{(K)} \iff$  labels of  $u$  and  $v$  are  $\neq$  after  $K$  iter. of the WL algorithm.

- ▶ If the WL test cannot distinguish between  $G_1, G_2$ , then MP-GNN also incapable of doing it.
- ▶ Ability of solving isomorphism = good measure of “expressivity” ?

# Other limitations

- ▶ **The oversmoothing problem:** if too many layers of MP-GNN, the node features tend to converge to a non-informative limit.

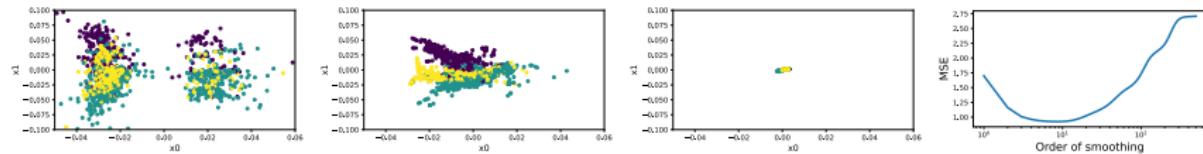


Figure: From Keriven 2022

- ▶ **Heterophily vs homophilie:** neighbours should have similar embeddings ? ([Luan et al. 2022](#)).

# Table of contents

---

From neural networks...

The basic ideas

Logistic regression and one layer neural-network

Convolutional neural networks

... to unsupervised node embeddings techniques...

A chronological start

... to graph neural networks

Learning with graphs

What is a GNN ?

A bit of group theory

Invariance and equivariance

Message-passing neural networks

Examples of GNN

The whole pipeline

Expressivity of GNN

Conclusion

# Conclusion

---

- ▶ Flexible: graph/node/edge classification, semi-supervised learning, link prediction...
- ▶ Generally state-of-the-art, but...
- ▶ ... sometimes do not work “that well” (compared to other DL)
- ▶ Simple methods may perform better but might be “forgotten” in benchmarks
- ▶ Room for improvement (many interesting challenges), but conventional DL wisdom might not hold
- ▶ Arguably, no real “ImageNet moment” yet for GNNs -*i* several recent initiatives for bigger datasets and more complex tasks (eg Open Graph Benchmark)

# References I

---

-  Belkin, Mikhail and Partha Niyogi (2003). "Laplacian eigenmaps for dimensionality reduction and data representation". In: *Neural computation* 15.6, pp. 1373–1396.
-  Bruna, Joan et al. (2013). "Spectral networks and locally connected networks on graphs". In: *arXiv preprint arXiv:1312.6203*.
-  Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst (2017). *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*. *arXiv: 1606.09375 [cs.LG]*.
-  Goldberg, Yoav and Omer Levy (2014). "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method". In: *arXiv preprint arXiv:1402.3722*.
-  Grinsztajn, Léo, Edouard Oyallon, and Gaël Varoquaux (2022). "Why do tree-based models still outperform deep learning on typical tabular data?" In: *Advances in Neural Information Processing Systems* 35, pp. 507–520.
-  Grover, Aditya and Jure Leskovec (2016). "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864.

## References II

---

-  Hamilton, William L (2020). *Graph representation learning*. Morgan & Claypool Publishers.
-  Hamilton, William L., Rex Ying, and Jure Leskovec (2018). *Inductive Representation Learning on Large Graphs*. arXiv: 1706.02216 [cs.SI].
-  Keriven, Nicolas (2022). "Not too little, not too much: a theoretical analysis of graph (over) smoothing". In: *Advances in Neural Information Processing Systems 35*, pp. 2268–2281.
-  Kipf, Thomas N and Max Welling (2016). "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907*.
-  LeCun, Yann et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE 86.11*, pp. 2278–2324.
-  Luan, Sitao et al. (2022). "Revisiting heterophily for graph neural networks". In: *Advances in neural information processing systems 35*, pp. 1362–1375.
-  Maron, Haggai et al. (2018). "Invariant and equivariant graph networks". In: *arXiv preprint arXiv:1812.09902*.
-  Mikolov, Tomas et al. (2013). "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems 26*.

## References III

---

-  Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain.". In: *Psychological review* 65.6, p. 386.
-  Scarselli, Franco et al. (2008). "The graph neural network model". In: *IEEE transactions on neural networks* 20.1, pp. 61–80.
-  Vaswani, Ashish et al. (2023). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL].
-  Velivcković, Petar et al. (2017). "Graph attention networks". In: *arXiv preprint arXiv:1710.10903*.
-  Wagstaff, Edward et al. (2019). "On the limitations of representing functions on sets". In: *International Conference on Machine Learning*. PMLR, pp. 6487–6494.
-  Weiler, Maurice et al. (2023). *Equivariant and Coordinate Independent Convolutional Networks. A Gauge Field Theory of Neural Networks*. URL: [https://maurice-weiler.gitlab.io/cnn\\_book/EquivariantAndCoordinateIndependentCNNs.pdf](https://maurice-weiler.gitlab.io/cnn_book/EquivariantAndCoordinateIndependentCNNs.pdf).

## References IV

---

-  Wu, Zonghan et al. (Jan. 2021). "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1, pp. 4–24. DOI: 10.1109/tnnls.2020.2978386. URL: <https://doi.org/10.1109%2Ftnnls.2020.2978386>.
-  Xu, Keyulu et al. (2019). *How Powerful are Graph Neural Networks?* arXiv: 1810.00826 [cs.LG].
-  Ying, Zhitao et al. (2018). "Hierarchical graph representation learning with differentiable pooling". In: *Advances in neural information processing systems* 31.
-  Zaheer, Manzil et al. (2018). *Deep Sets*. arXiv: 1703.06114 [cs.LG].
-  Zeiler, Matthew D and Rob Fergus (2014). "Visualizing and understanding convolutional networks". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part I* 13. Springer, pp. 818–833.
-  Zhang, Ziwei, Peng Cui, and Wenwu Zhu (2020). *Deep Learning on Graphs: A Survey*. arXiv: 1812.04202 [cs.LG].