The Geometry of Deep Learning. Lecture 4: Geometric Deep Learning

Rita Fioresi

Fabit

January 23, 2026

Graph Neural Networks

Main References:

- Stanford course CS224w by Leskovec:
  http://cs224w.stanford.edu/
- Bronstein et al.
  Geometric deep learning: going beyond Euclidean data,
  https://arxiv.org/abs/1611.08097

Recall: Deep Learning ingredients

- **Score function**: it is a function of the weights w (es. linear classifier)
- **Loss function**: measures error (L_i loss of datum i)

$$L_i = -log \frac{e^{f_{y_i}}}{\sum_j e^{f_j}} = -f_{y_i} + log \sum_j e^{f_j}, \qquad L = \sum_i L_i$$

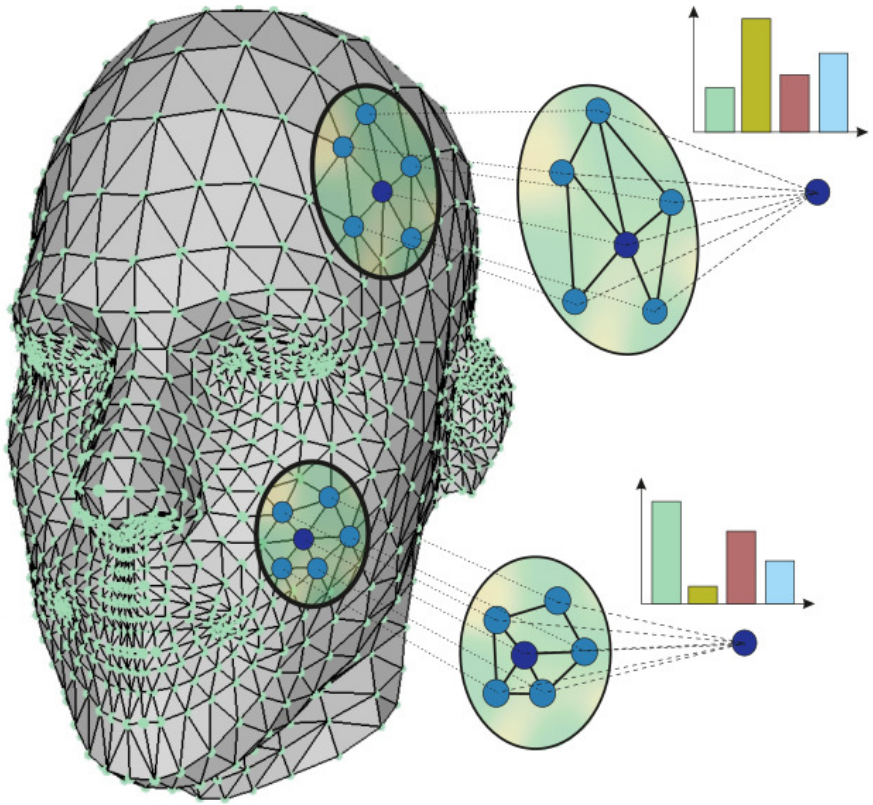
- **Optimizer**: for weights update "minimizes" the Loss (via stochastic gradient).

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \nabla L_{\text{stoc}}, \qquad \nabla L_{\text{stoc}} = \sum_{i=1}^{32} \nabla L_{\text{rand(i)}}$$

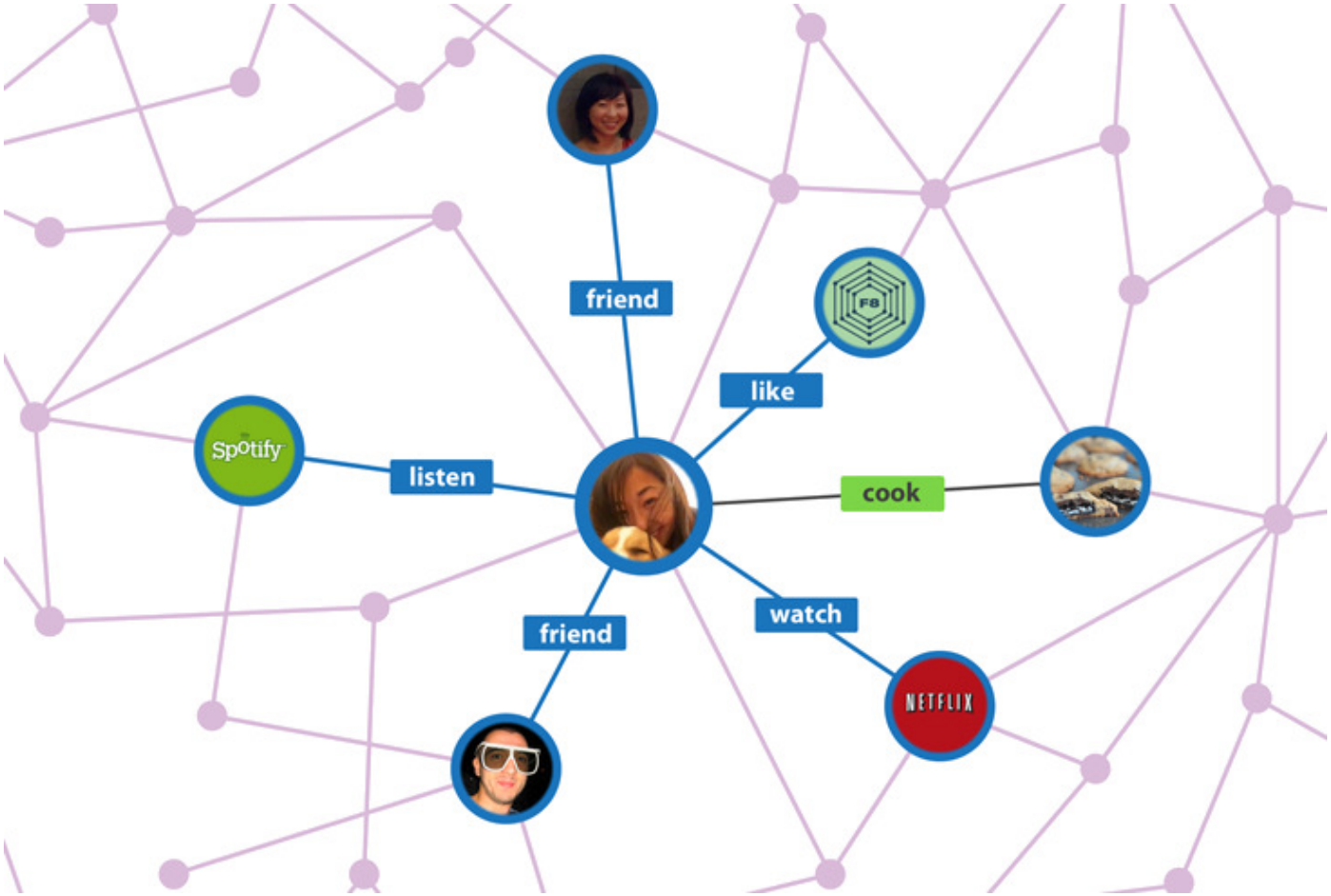
Graph Neural Networks GNN: Overview

Geometric Deep Learning: Bronstein et al, 2016.

- Do convolutions on graphs (called "non euclidean domains")
- Manipulate complex and heterogeneous datasets (beyond image recognition)
- Effectively work on 3D images



Example: Social network as graph



Data organized as Graphs

Image credit: Medium

Social Networks

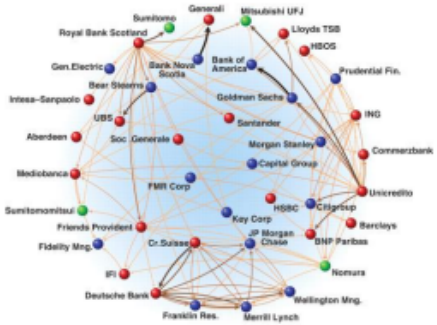


Image credit: Science

Economic Networks

Image credit: Lumen Learning

Communication Networks

Citation Networks

Image credit: Missoula Current News

Internet

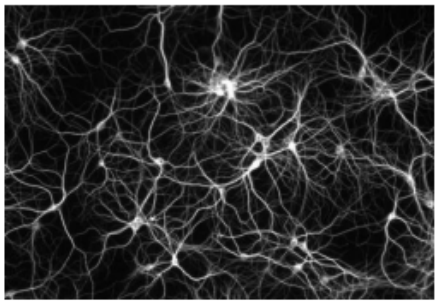


Image credit: The Conversation

Networks of Neurons

- **We use PyG (PyTorch Geometric):**
  - The ultimate library for Graph Neural Networks
- **We further recommend:**
  - **GraphGym**: Platform for designing Graph Neural Networks.
    - Modularized GNN implementation, simple hyperparameter tuning, flexible user customization
  - Both platforms are very helpful for the course project (save your time & provide advanced GNN functionalities)
- **Other network analytics tools**: SNAP.PY, NetworkX

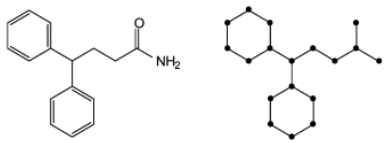
GNN: Supervised Classification problems

GNN are more versatile in classification tasks. Here are some possibilities:

- **Many graphs**. We have homogeneous data coming in the form of graphs each coming with a label.

  Goal: predict the labels of the graphs.

  Example: classification of proteins (label: enzymes or not enzymes). Each datum is a protein represented with a graph.



- **One graph**. We have a unique graph, we want to classify either nodes (which may also be heterogeneous) or graph links

  Goal: predict the labels of nodes (edges).

  Example: a social network. Nodes=people, Link=friendship.

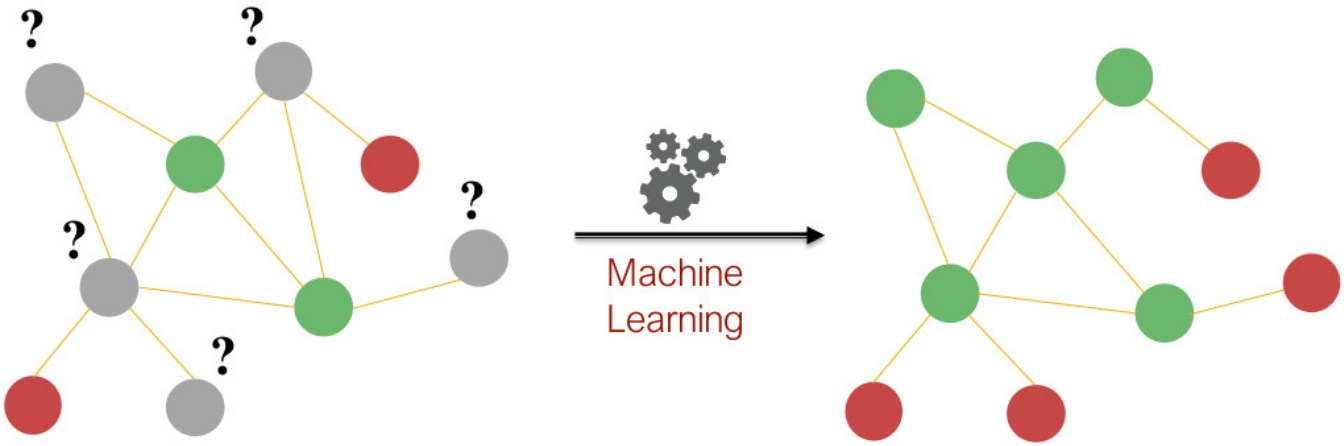
  1) We want to predict the links of a given node.

  2) We want to predict if a node is or is not a celebrity.

  Nodes come with **features** that help to understand the **similarity** between two nodes.

We take into exam the **one graph** case and only the **node classification** problem.

Goal: We have a graph, we know the label of some nodes, we want to understand the labels of all nodes.



Node classification

Steps in GNN learning process

A GNN consists of the following steps:

- **Encoding:** realize a (low) dimensional embedding of the graph. Typically via a set of *learned* convolutional layers.
- **Decoding:** from the embedding we compute a **SCORE** Typically via a *learned* linear layer.
- **Loss function** (same idea as DL)
- **Optimizer** (same idea as DL)

Once score, loss and optimizer are given, the training, validation and step take place in the same way as in Deep Learning algorithm.

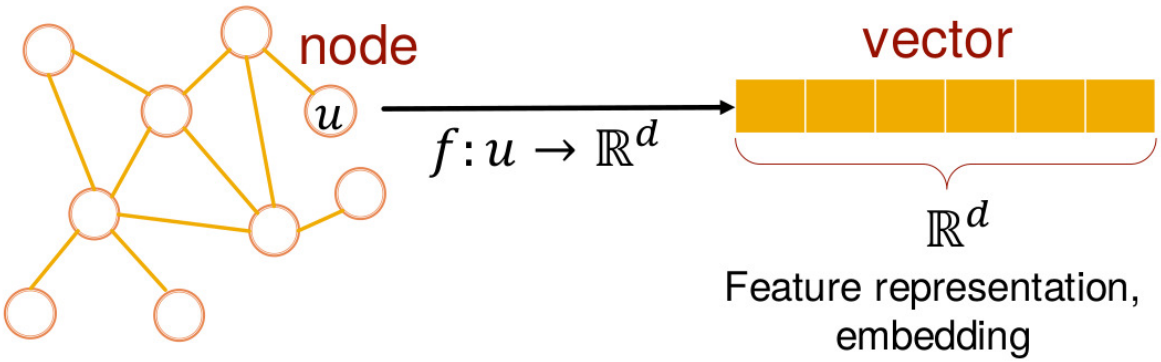
KEY POINT for GDL (graphs) vs DL (images): Encoding!

- Deep Learning: convolutions
- Geometric Deep Learning: message passing

Describes how to learn node and graph embeddings for a given task. (Our example: node classification).

Encoder-decoder framework:

- Encoder: produces an embedding of a graph in a low dimensional space (keeping track of features!)
- Decoder: predicts the score based on embedding to match node similarity



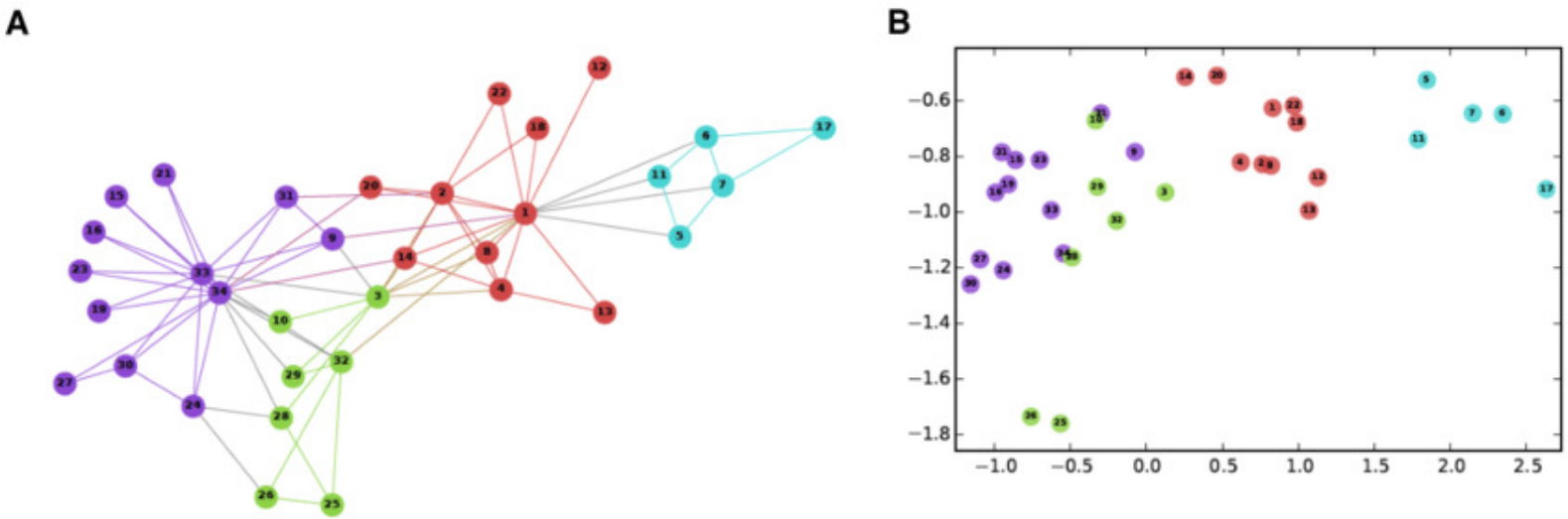
Main reference:

- Kipf, Thomas N; Welling, Max (2016) *Semi-supervised classification with graph convolutional networks". International Conference on Learning Representations. 5 (1): 61–80. arXiv:1609.02907*

Encoding of the Zachary Karate club:

Setting: 1 graph, each node has a label in $\{0, 1, 2, 3\}$
Features for each node: vector of 34 coordinates.

We map the graph into the plane each node corresponds to a point:



Important: nodes that are "near" in the graph correspond to points which are "near" in the plane. The encoding function:

$$f : \text{Zachary graph} \longrightarrow \mathbb{R}^2$$

Implementation via Colab: GNN structure

GCN(
(conv1): GCNConv(34, 4)
(conv2): GCNConv(4, 4) } ENCODER
(conv3): GCNConv(4, 2)
(classifier): Linear(2, 4)) } DECODER

34: dimension of features vector
2: dimension of encoding space, $f : \mathbb{R}^{34} \longrightarrow \mathbb{R}^2$ **encoding function**
4: predicting labels, $s : \mathbb{R}^2 \longrightarrow \mathbb{R}^4$ **score function**, correct label is assigned to the highest value (e.g. $s(v) = (34, 5, 76, 1)$ gives label 2 in $\{0, 1, 2, 3\}$).

Main References:

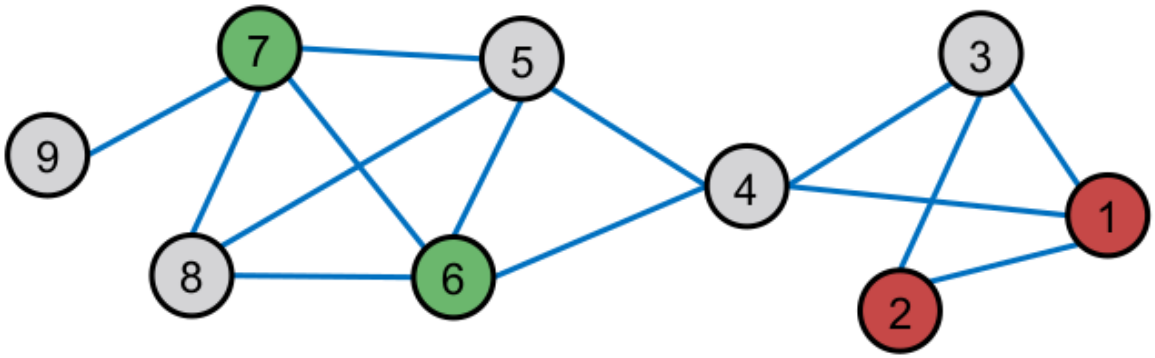
- Stanford course CS224w by Leskovec: http://cs224w.stanford.edu/
- Kipf, Thomas N; Welling, Max (2016) *Semi-supervised classification with graph convolutional networks". International Conference on Learning Representations. 5 (1): 61–80. arXiv:1609.02907*

Convolutions and node classification

Classification label of a node v in graph neural network G may depend on:

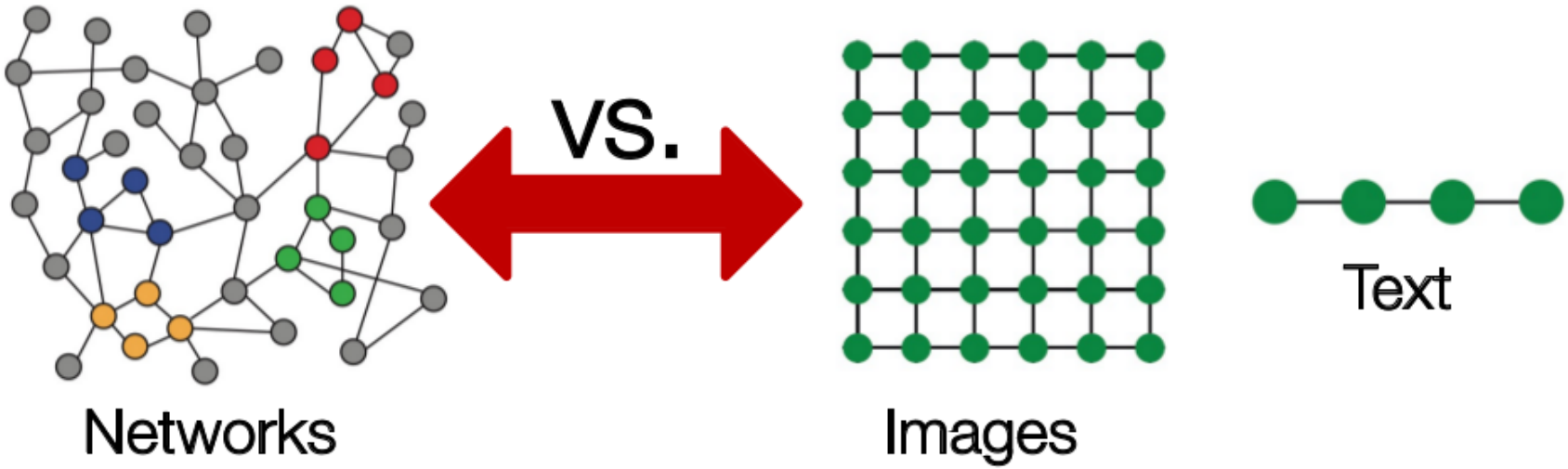
- Features of v
- Labels of the nodes in v's neighborhood (topology of G)
- Features of the nodes in v's neighborhood (topology of G)

Assumption: topology matters! i.e. link occurs between nodes that have common/similar features. If not: impossible to classify!



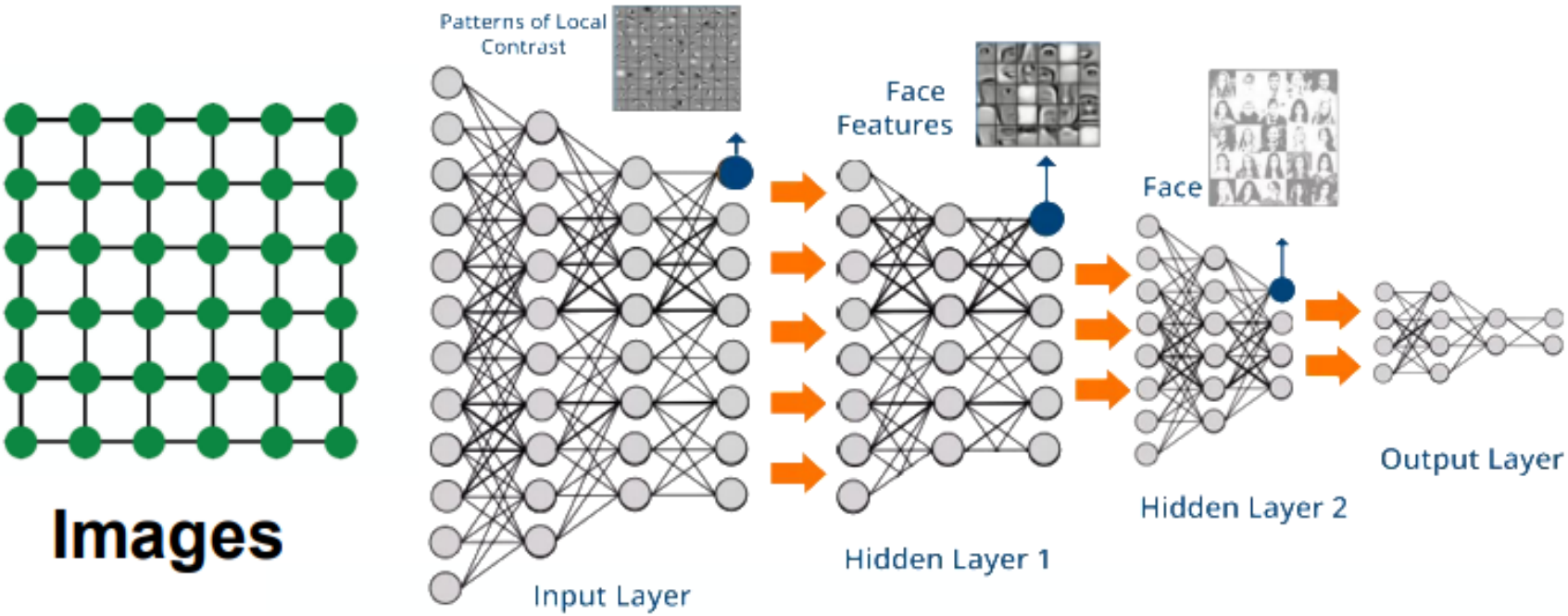
red: label 0
green: label 1
Classification Problem: give a label to gray vertices.

Convolutions on Graphs are more complicated with respect to images or text (or DNA analysis)

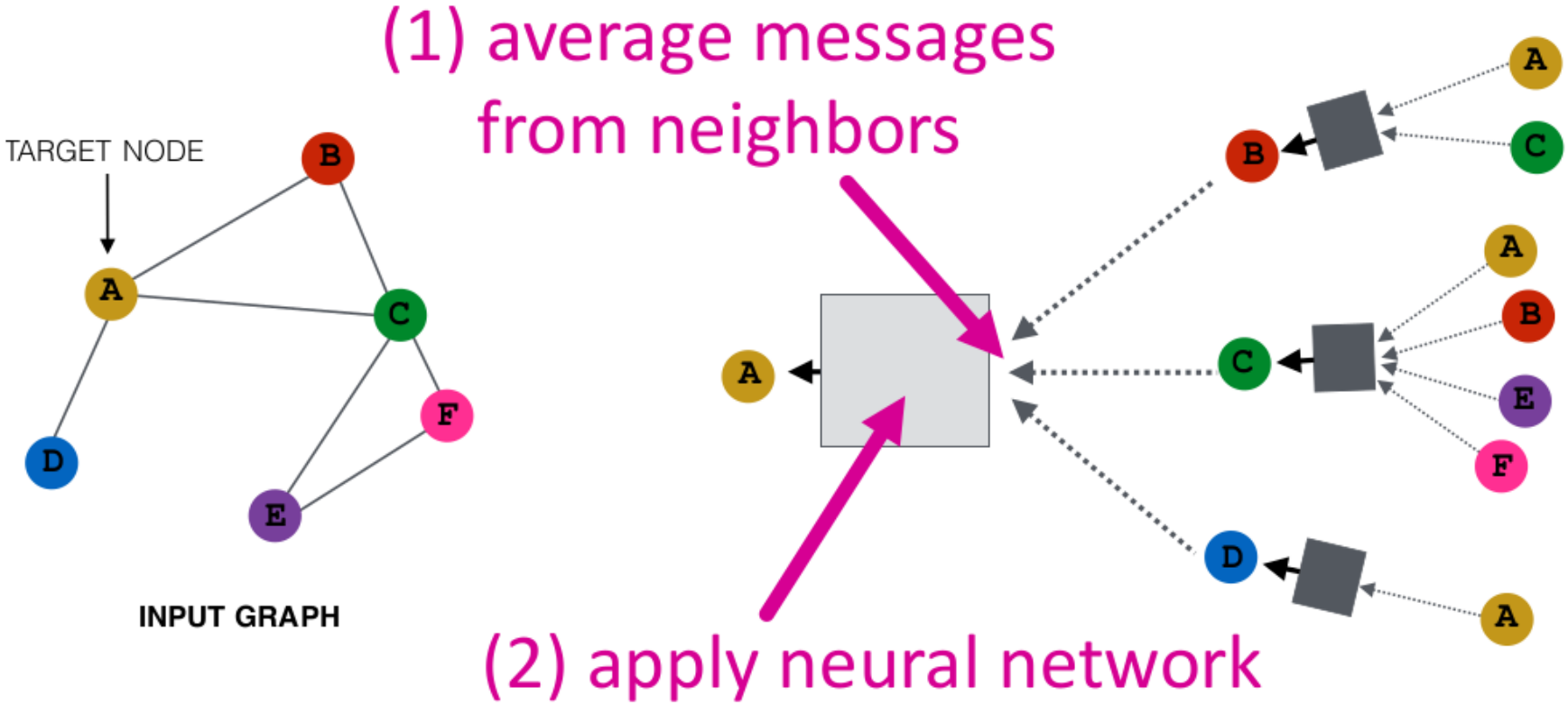


Topology matters! The neighbourhood of different points are different.
It is not possible to perform a convolution using kernels=filters as in the images setting.

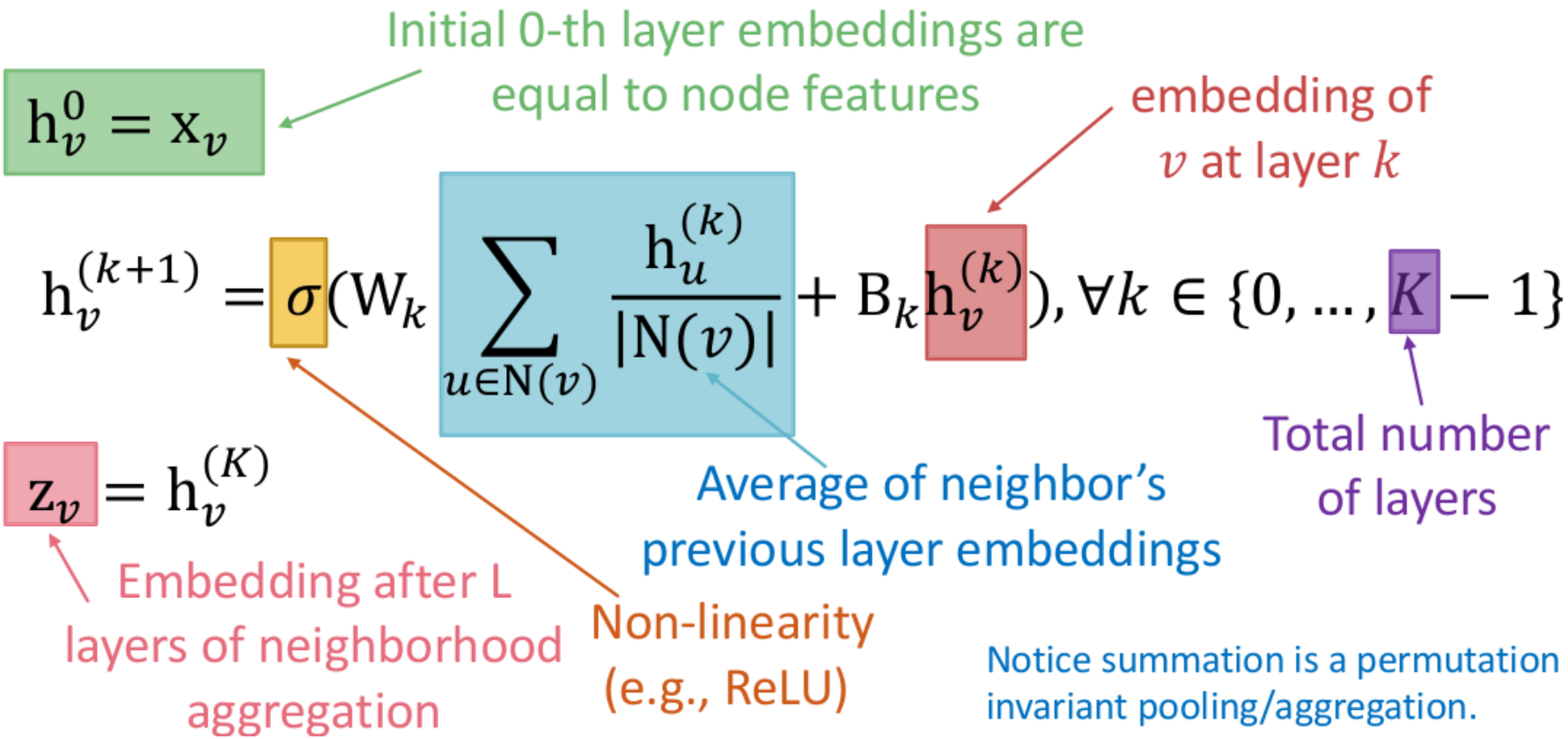
Convolutions on Grids



The mechanism of **message passing** is an approximation of the mathematical operation of convolutions on graph (Spectral methods):



The operation of message passing in practice



$h_v^{(k)}$: the hidden representation of node v at layer k

W_k: weight matrix for neighborhood aggregation (B_k: bias).

Important: Message passing and neighbor aggregation in graph convolution networks is **permutation equivariant**.

Zachary Karate club: encoding mechanism via message passing

$$\begin{aligned} (conv1) : GCNConv(34, 4) \quad & h_v^{(0)} = z^0 \mapsto h_v^{(1)} \\ (conv2) : GCNConv(4, 4) \quad & h_v^{(1)} \mapsto h_v^{(2)} \\ (conv3) : GCNConv(4, 2) \quad & h_v^{(2)} \mapsto h_v^{(3)} = z_v \end{aligned}$$

W^1 is a 34×4 weight matrix (same for ALL nodes)
W^2 is a 4×4 weight matrix (same for ALL nodes)
W^3 is a 4×2 weight matrix (same for ALL nodes)
Here $K = 3$: number of layers.

Zachary Karate club: Training, validation, test

- **Training+Validation**: 4 randomly chosen nodes
- **Test**: 30 nodes
- **Accuracy**: 71% on 4 classes

GAT, case study: Cora dataset

Main References

- Stanford course CS224w by Leskovec:
  http://cs224w.stanford.edu/
- Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio
  Graph Attention Networks, 2018
  https://arxiv.org/abs/1710.10903

Goal: Specify arbitrary importance to different neighbors of each node in the graph.

Idea: Compute embedding of each node (these are the embedded features $h^{(\ell)}$ in the graph following the **attention strategy** $a : \mathbb{R}^F \times \mathbb{R}^F \longrightarrow \mathbb{R}$:

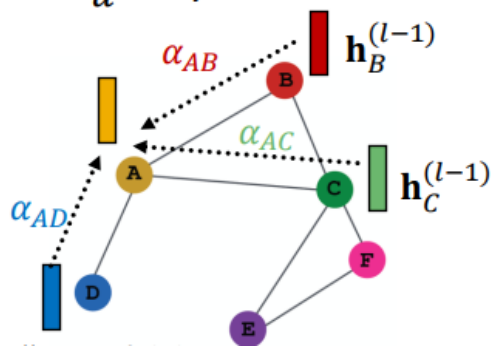
- Nodes get their neighborhoods message via the function a.
- This implicitly specifies different weights to different nodes in a neighborhood.

- **Weighted sum** based on the **final attention weight** α_{vu}:

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \alpha_{vu} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right)$$

Weighted sum using α_{AB}, α_{AC}, α_{AD}:

$$\mathbf{h}_A^{(l)} = \sigma\left(\alpha_{AB}\mathbf{W}^{(l)}\mathbf{h}_B^{(l-1)} + \alpha_{AC}\mathbf{W}^{(l)}\mathbf{h}_C^{(l-1)} + \alpha_{AD}\mathbf{W}^{(l)}\mathbf{h}_D^{(l-1)}\right)$$



We modify the message passing into the Encoder part creating a new "convolution" method.

Attentional Layer

$$a : \mathbb{R}^F \times \mathbb{R}^F \longrightarrow \mathbb{R}, \qquad e_{ij} := a(Wh_i, Wh_j) \quad \text{attention coefficients}$$

e_{ij}: measure the importance of node j's features to node i
a: scalar product or another **learned function**
We normalize the e_{ij} with softmax:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathbf{N}_i} exp(e_{ik})}$$

$\mathbf{N}_i$: neighbourhood of vertex i.

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \alpha_{vu} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right)$$

Attention weights

Not all node's neighbors are equally important

- **Attention** is inspired by cognitive attention.
- The **attention** α_{vu} focuses on the important parts of the input data and fades out the rest.
  - **Idea:** the NN should devote more computing power on that small but important part of the data.
  - Which part of the data is more important depends on the context and is learned through training.

Message passing vs Attention

The feature at step ℓ:

$$(1)\ h_v^{(\ell)} = \sigma\Big(\sum_{u\in\mathcal{N}} W^{(\ell)} \frac{h_u^{(\ell-1)}}{\mathcal{N}(u)}\Big) \quad \text{versus} \quad (2)\ h_v^{(\ell)} = \sigma\Big(\sum_{u\in\mathcal{N}} \alpha_{vu} W^{(\ell)} h_u^{(\ell-1)}\Big)$$

1. **Message passing:** the information is coming **uniformely** from the neighbouring nodes.
2. **Attention mechanism:** the information focuses on the importance of the information (features) of neighbouring nodes, encoded in the function α_{ij}.

Example: The Cora Dataset

The Cora Dataset consists of:

- Nodes: 2708 scientific papers
- Features: each node is equipped with 1433 features. $h^{(0)} \in \mathbb{R}^{1433}$ is a vector with entries 0 or 1. Each coordinate is assigned 1 if the corresponding key word appears in the paper, 0 if not.
- Edges: two nodes (papers) are connected if one of the two cites the other (the graph is nevertheless undirected).
  There are 10556 edges.
- Node labels: $\{0, \ldots, 5\}$ correspond to a paper belonging to a certain category or another (i.e. ML, MAT, PHYS, etc).

Training nodes: 140 chosen randomly.
Test accuracy: 95% (test size: 140)