

Graph Neural Networks in Big Data Analytics: Introduction III

Alexander Schönhuth



Bielefeld University
November 3, 2022

CONTENTS TODAY

- ▶ Reminder: Simple Graph Neural Networks
- ▶ Reminder: Convolution
- ▶ Message Passing

Graph Neural Networks: Definition

GRAPH NEURAL NETWORKS: DEFINITION

DEFINITION [GRAPH NEURAL NETWORK]:

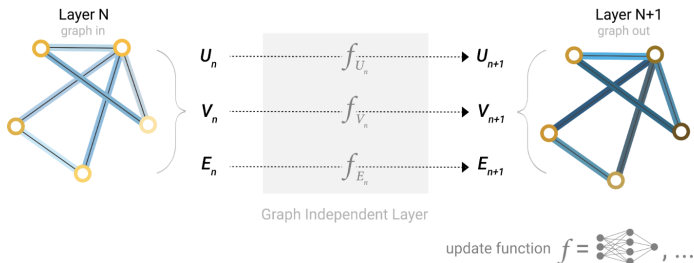
A *graph neural network (GNN)* is an

- ▶ optimizable transformation on
- ▶ all attributes of the graph (nodes, edges, global) that
- ▶ preserves graph symmetries (permutation invariances)

- ▶ GNN's adopt a "graph-in, graph-out" architecture:
 - ▶ Graph loaded with information accepted as input
 - ▶ Embeddings are progressively transformed
 - ▶ Connectivity of input graph never changed

Simple Graph Neural Networks

SIMPLE GNN I

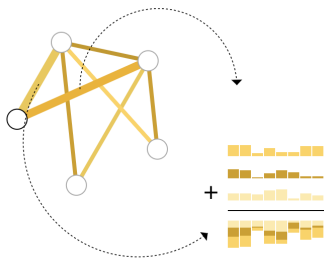


U_n, V_n, E_n reflect global, vertex, edge information.

From <https://distill.pub/2021/gnn-intro/>

- ▶ Initial embeddings: U_0, V_0, E_0
- ▶ $U_n, V_n, E_n, n \geq 0$ iteratively updated to $U_{n+1}, V_{n+1}, E_{n+1} \dots$
- ▶ ... using multilayer perceptions (MLP's) $f_{U_n}, f_{V_n}, f_{E_n}$ until ...
- ▶ ... final layer is reached, where final embeddings are computed.

PREDICTIONS BY POOLING

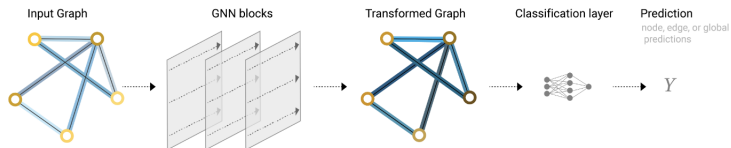


From <https://distill.pub/2021/gnn-intro/>

- ▶ May not always be so simple. For example:
 - ▶ Would like to raise predictions about nodes
 - ▶ But only edge embeddings available
- ▶ *Solution:* Aggregate (adjacent) edge embeddings using pooling function

$$\rho_{E_n \rightarrow V_n}$$

PREDICTIONS BY POOLING II



GNN: End-to-end prediction task

From <https://distill.pub/2021/gnn-intro/>

- ▶ Classification layer comprises pooling as well, if necessary
- ▶ *Remark:* Classification model can be any differentiable model
 - ▶ Models other than MLP's conceivable

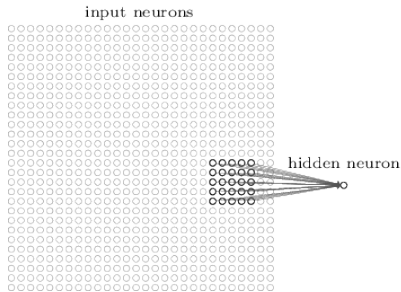
Reminder: Convolutional Neural Networks (CNNs)

CONVOLUTIONAL NEURAL NETWORKS

LOCAL RECEPTIVE FIELDS

In a convolutional NN,

- ▶ Every node in the first hidden layer is connected to a rectangular subregion
- ▶ Here: subregion = square of $5 \times 5 = 25$ input neurons



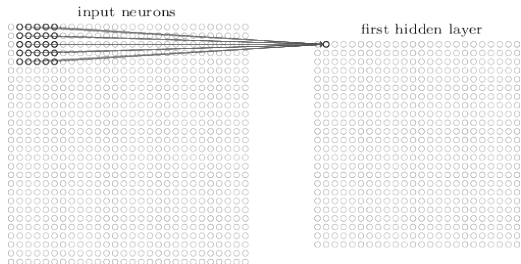
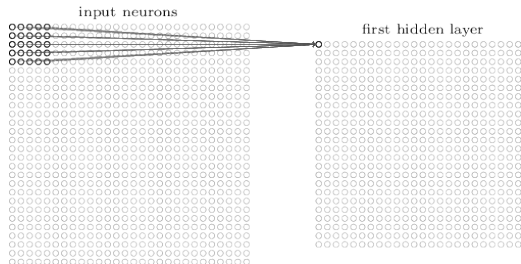
Convolutional filter of size 5×5

Definition

The region in the input images to which a hidden neuron is connected is called the *local receptive field (LRF)* of the hidden neuron.

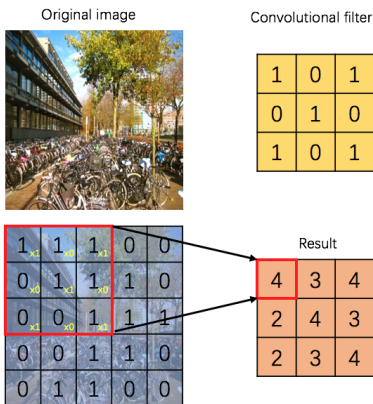
CONVOLUTIONAL NEURAL NETWORKS

LOCAL RECEPTIVE FIELDS



CONVOLUTIONAL NEURAL NETWORKS

CONVOLUTIONAL FILTERS



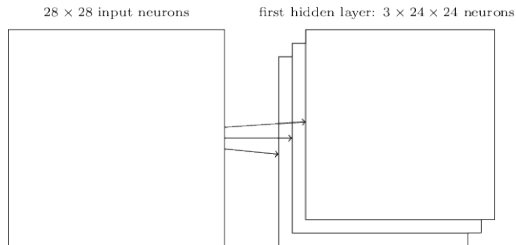
For generating one hidden layer, identical parameters, together defining one convolutional filter, are used

CONVOLUTIONAL NEURAL NETWORKS

CONVOLUTIONAL FILTERS

Definition

A *feature map* is a mapping associated with one convolutional filter.



- ▶ A complete convolutional layer consists of several hidden sublayers
- ▶ Each sublayer is defined by one feature map

NEURAL NETWORKS

CONVOLUTION FILTERS

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

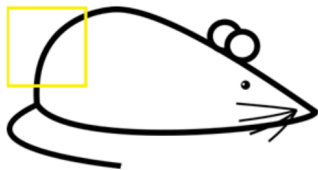
Filter for recognizing a curve

NEURAL NETWORKS

CONVOLUTION FILTERS



Original image



Visualization of the filter on the image

NEURAL NETWORKS

CONVOLUTION FILTERS



Visualization of the
receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive
field

*

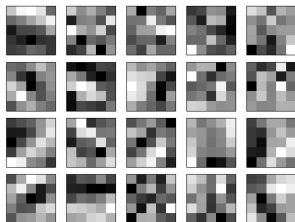
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = $(50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600$ (A large number!)

CONVOLUTIONAL NEURAL NETWORKS

CONVOLUTIONAL FILTERS REAL WORLD EXAMPLE



MNIST example, 20 different filters

- ▶ The darker the more positive, the whiter the more negative
- ▶ In reality, convolutional filters are hard to interpret
- ▶ Literature: M.D. Zeiler, R. Fergus, “Visualizing and Understanding Convolutional Networks”, <https://arxiv.org/abs/1311.2901>

CONVOLUTIONAL NEURAL NETWORKS

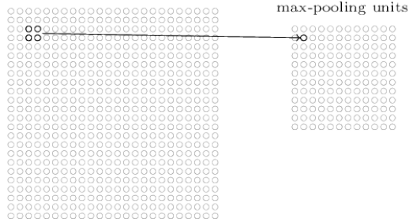
POOLING LAYERS

- ▶ In addition to convolutional layers, CNN's make use of *pooling layers*.
- ▶ Pooling layers generate *condensed feature maps*: it takes a rectangle of neurons, and summarizes their values into one value
- ▶ This generates a considerably smaller layer

CONVOLUTIONAL NEURAL NETWORKS

POOLING LAYERS

hidden neurons (output from feature map)

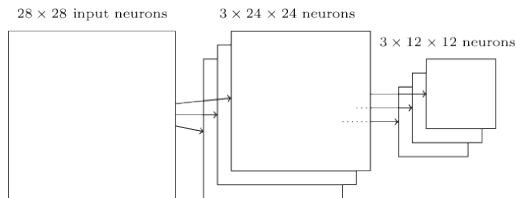


2×2 pooling

- ▶ *Max pooling*: Each $L \times L$ rectangle is mapped onto the maximum of its values
- ▶ *L2 pooling*: Each $L \times L$ rectangle is mapped to the rooted average of the squares of the values
- ▶ This overall yields a layer that is $L \times L$ times smaller

CONVOLUTIONAL NEURAL NETWORKS

COMBINING CONVOLUTIONAL AND POOLING LAYERS



Convolutional layer followed by pooling layer

- ▶ Convolutional and pooling layers are used in combination
- ▶ Pooling layers usually follow convolutional layers
- ▶ *Intuition:*
 - ▶ The exact location of the occurrence of a feature is not important
 - ▶ Pooling helps to handle distortions and rotations

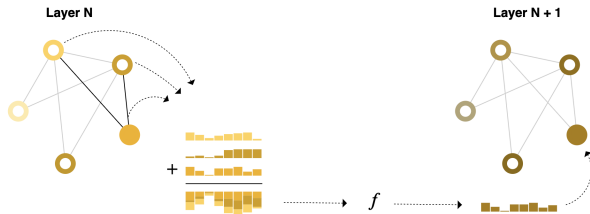
Message Passing

MESSAGE PASSING: MOTIVATION

- ▶ Simple GNN's so far presented
 - ▶ do not pool within the GNN layer
 - ▶ have learned embeddings unaware of graph connectivity
- ▶ *Goal:* Neighboring nodes and edges
 - ▶ exchange information
 - ▶ influence each other's updated embeddings
- ▶ *Solution:* Message passing

MESSAGE PASSING: PROTOCOL

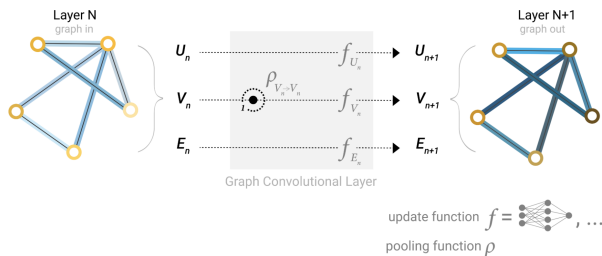
1. Each node: gather all embeddings (= *messages*) of neighboring nodes
2. Aggregate all messages using an aggregation function
3. Pooled messages passed through update function (e.g. learned NN)



Message passing: Aggregating information from neighboring nodes

From <https://distill.pub/2021/gnn-intro/>

MESSAGE PASSING AND CONVOLUTION I

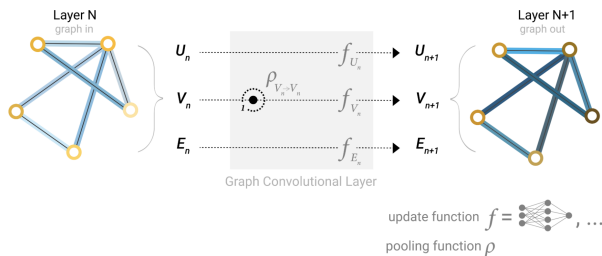


Message passing as convolution on graphs

From <https://distill.pub/2021/gnn-intro/>

- ▶ Message passing and convolution are similar in spirit
- ▶ *Commonality*: Process element's neighbors to update element
 - ▶ *Graphs*: Elements are nodes
 - ▶ *Images*: Elements are pixels

MESSAGE PASSING AND CONVOLUTION II

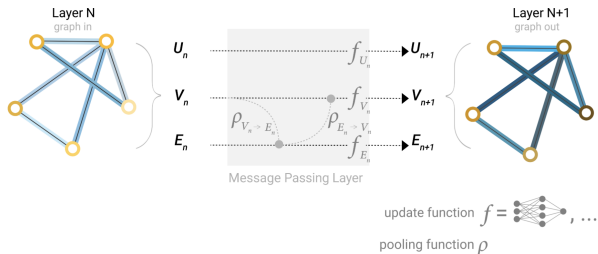


Message passing as convolution on graphs

From <https://distill.pub/2021/gnn-intro/>

- ▶ Message passing and convolution are similar in spirit
- ▶ *Difference:*
 - ▶ *Graphs:* Number of neighbors varies per node
 - ▶ *Images:* Number of neighbors constant per pixel

POOLING WITHIN LAYERS I

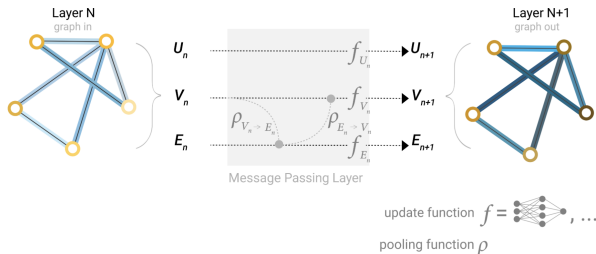


Passing messages from edges to nodes within layer

From <https://distill.pub/2021/gnn-intro/>

- ▶ *Situation:* Want node from edge information
- ▶ *Solution:*
 - ▶ Pool information of neighboring edges and transfer to node: $\rho_{E_n \rightarrow V_n}$
 - ▶ After first iteration: add node information, and transfer pooled information from nodes to edges: $\rho_{V_n \rightarrow E_n}$

POOLING WITHIN LAYERS II

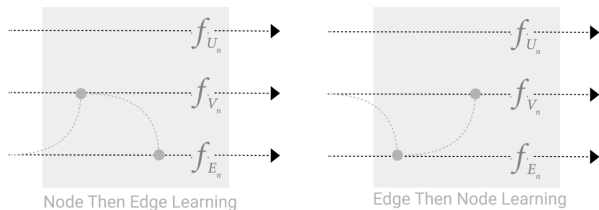


Passing messages from edges to nodes within layer

From <https://distill.pub/2021/gnn-intro/>

- ▶ *Issue:* Node and edge information may differ in size
- ▶ *Solution:* Linear map transforms node into edge information
- ▶ And vice versa. Other maps than linear maps conceivable

POOLING WITHIN LAYERS III

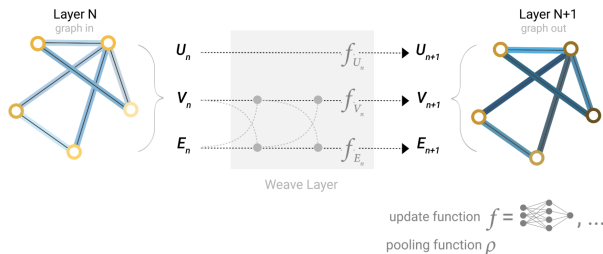


Message passing: different modes

From <https://distill.pub/2021/gnn-intro/>

- ▶ *Left:* Learning edge information from node information
- ▶ *Right:* Learning node information from edge information

POOLING WITHIN LAYERS IV

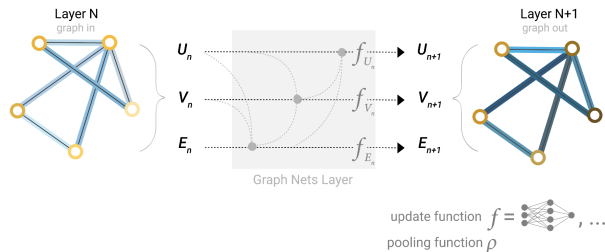


Weave layer: learning node information from edges and learning edge information from nodes

From <https://distill.pub/2021/gnn-intro/>

- ▶ f_{V_n} processes node information from edge information and node itself
- ▶ f_{E_n} processes edge information from node information and edge itself

POOLING WITHIN LAYERS V

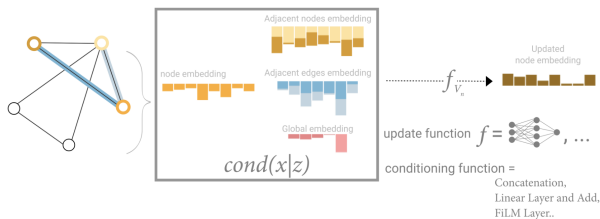


Global information: aggregate from nodes and edges

From <https://distill.pub/2021/gnn-intro/>

- ▶ *Issue:* After k layers, nodes can reach k -neighborhoods at most
- ▶ *Solution:* Consider *master node* or *global context vector*
- ▶ Update global context vector by pooling node and/or edge information

POOLING WITHIN LAYERS VI



Node information from pooling node, edge and global information

From <https://distill.pub/2021/gnn-intro/>

- ▶ *Situation:* Want node information based on all information
- ▶ *Issue:* Different informations differ in size
- ▶ *Solution:* Xcondition node information x on other information z
- ▶ Use *conditioning function*: concatenation, linear etc.

MESSAGE PASSING AND RANDOM WALKS

- ▶ Let $n := |V|$ be the number of nodes of a graph (V, E)
- ▶ Let $A \in \{0, 1\}^{n \times n}$ be its adjacency matrix
- ▶ Let m be the length of node information vectors
- ▶ Let $X \in \mathbb{R}^{n \times m}$ be the node feature matrix
 - ▶ Rows in X are m -dimensional information vectors of nodes

Consider

$$B = AX$$

We obtain

$$B_{ij} = A_{i1}X_{1j} + \dots + A_{in}X_{nj} = \sum_{\substack{k=1 \\ A_{ik} > 0}}^n A_{ik}X_{kj}$$

MESSAGE PASSING AND RANDOM WALKS

Consider

$$B = AX$$

We obtain

$$B_{ij} = A_{i1}X_{1j} + \dots + A_{in}X_{nj} = \sum_{\substack{k=1 \\ A_{ik}>0}}^n A_{ik}X_{kj}$$

Interpretation:

- ▶ Each row B_i reflects a new information vector for node v_i
- ▶ B_i again has dimension m
- ▶ Each B_{ij} is the aggregation of j -th entries of information vectors of neighbors of v_i
 - ↳ Note that $A_{ik} = 1$ if and only if v_i and v_k are neighbors

MESSAGE PASSING AND RANDOM WALKS

Consider

$$B = AX$$

We obtain

$$B_{ij} = A_{i1}X_{1j} + \dots + A_{in}X_{nj} = \sum_{\substack{k=1 \\ A_{ik}>0}}^n A_{ik}X_{kj}$$

Interpretation:

- ▶ Replacing A with A^K yields aggregation of information vectors of K -neighbors
 - ☞ $A_{ik}^K = 1$ iff (sic!) v_i and v_k can be connected by path of length K
- ▶ This relates to random walks on the graph
 - ☞ Recall the random walk mechanism for computing PageRank

GRAPH ATTENTION NETWORKS

Motivation:

- ▶ When aggregating one would like to consider weighted sums

$$B_{ij} = w_{ij,1}A_{i1}X_{1j} + \dots + w_{ij,n}A_{in}X_{nj} = \sum_{\substack{k=1 \\ A_{ik}>0}}^n w_{ij,k}A_{ik}X_{kj}$$

- ☞ Some neighbors are more important than others
- ▶ *Challenge:* How to compute weights in permutation invariant way?

GRAPH ATTENTION NETWORKS

Solution:

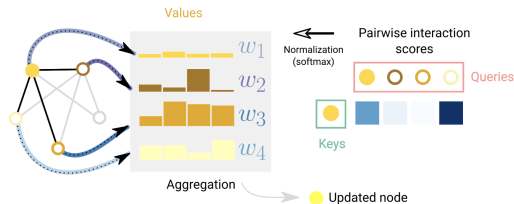
- ▶ *Solution:* Base weights on pairs of nodes alone, so

$$w_{ij,k} = f(v_i, v_k)_j$$

where

- ▶ $f(v_i, v_j)$ is m -dimensional vector that
- ▶ depends on information of nodes v_i and v_j alone
 - ☞ Renumbering nodes does not matter

GRAPH ATTENTION NETWORKS

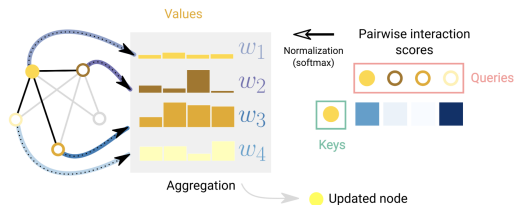


Graph attention network: mechanism

From <https://distill.pub/2021/gnn-intro/>

- ▶ *Attention Networks: Compute value from comparing key and query*
 - ▶ *Original motivation:* Compute strength of dependency between words in (or across) sentences
 - ▶ *Original application:* Used in language translation for example

GRAPH ATTENTION NETWORKS



Graph attention network: mechanism

From <https://distill.pub/2021/gnn-intro/>

- ▶ *Here:* Compare information vectors of two nodes
 - ▶ One node is query, other node is key, weight is value
 - ▶ *Example:*

$$f(v_i, v_j) = \langle v_i, v_k \rangle$$

evaluates as scalar product of information vectors of v_i and v_k

OUTLOOK

- ▶ Convolution on Graphs
- ▶ The Graph Laplacian
- ▶ Polynomial Filters

Thanks for your attention!