max planck institut informatik

SIC Saarland Informatics Campus

Probabilistic Graphical Models and Their Applications

Graph Neural Networks - Introduction

@ Jan 13, 2021

Bernt Schiele

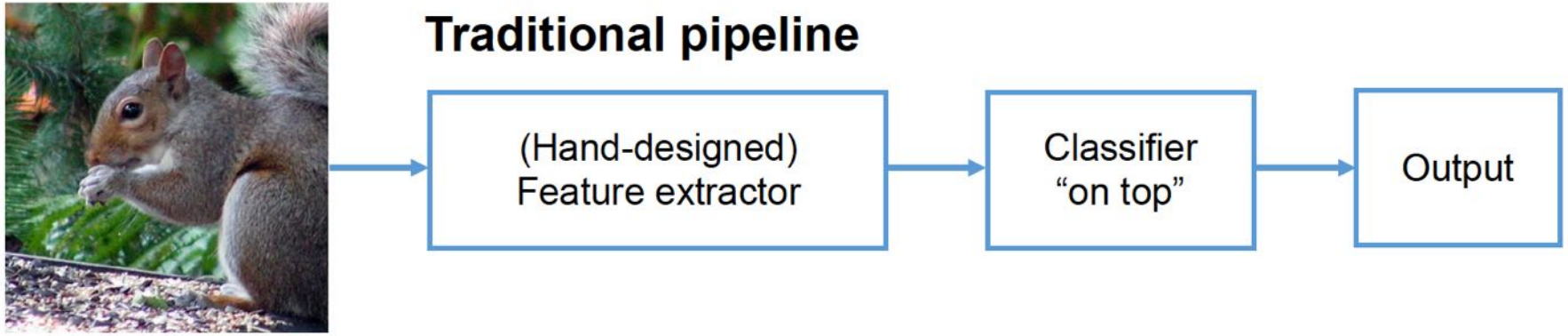
www.mpi-inf.mpg.de/gm/

Max Planck Institute for Informatics & Saarland University,
Saarland Informatics Campus Saarbrücken

Overview Today's Lecture

- Motivation
  - Graph neural networks and neural message passing
  - Machine learning on graphs
- Convolutions in Time, Space and on Graphs
- Graph Neural Networks and Graph Convolutional Networks
- Graphs and Graph Shift Operators

Traditional vs. Deep End-to-End Training



slide credit: Thomas Kipf

Traditional vs. Deep End-to-End Training

Traditional pipeline

(Hand-designed) Feature extractor → Classifier "on top" → Output

End-to-end learning

Deep neural network → Output

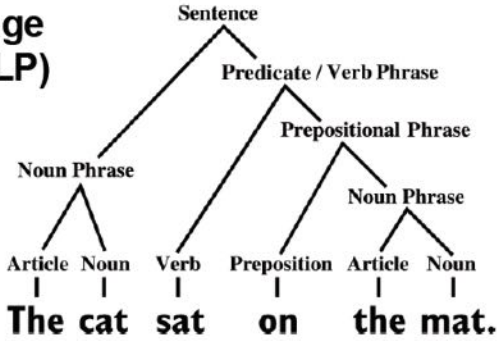
slide credit: Thomas Kipf

Deep Learning for Data on "Grids"

IMAGENET

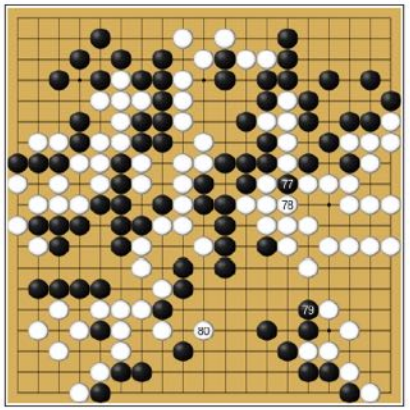
Speech data

Natural language processing (NLP)



...

Grid games



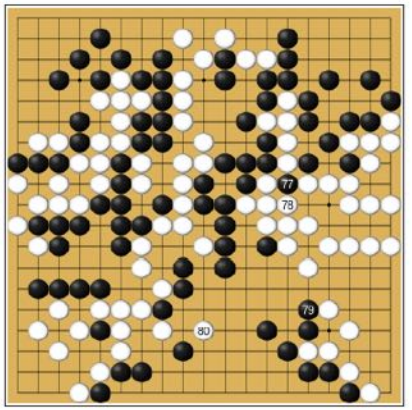
slide credit: Thomas Kipf

Deep Learning for Data on "Grids"

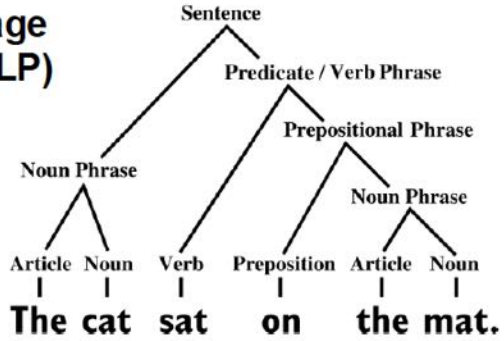
IMAGENET

Speech data

Grid games



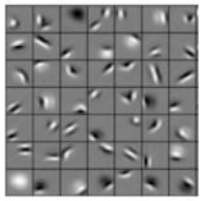
Natural language processing (NLP)

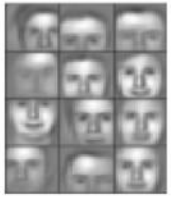


...

Deep neural nets that exploit:

- translation equivariance (weight sharing)
- hierarchical compositionality



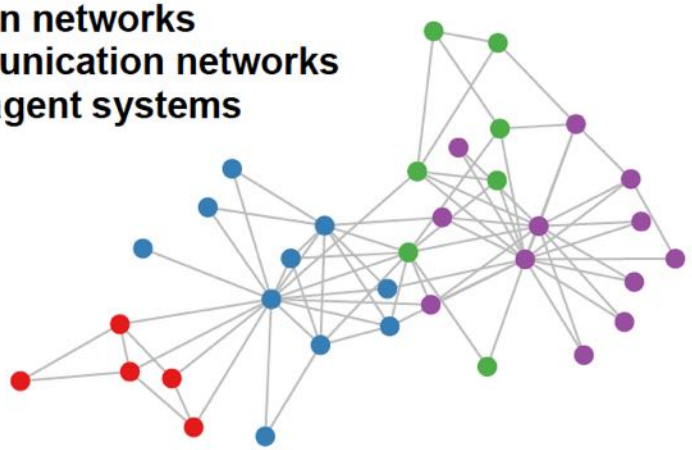


slide credit: Thomas Kipf

Graph-Structured / Relational Data

A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems

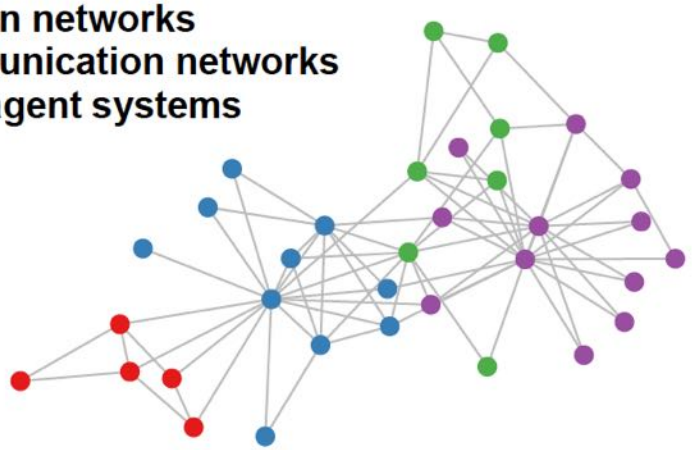


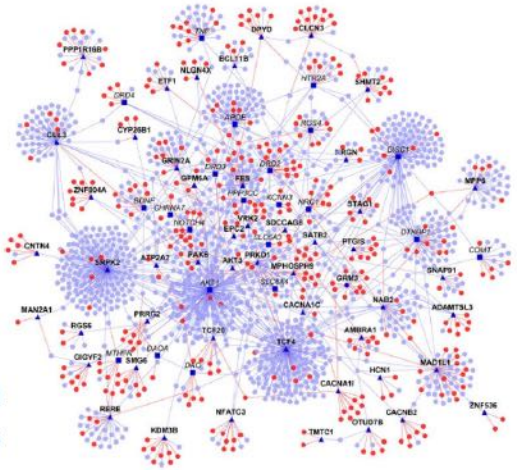
slide credit: Thomas Kipf

Graph-Structured / Relational Data

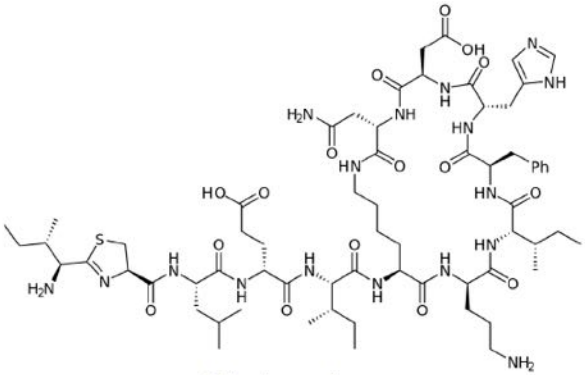
A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems





Protein interaction networks



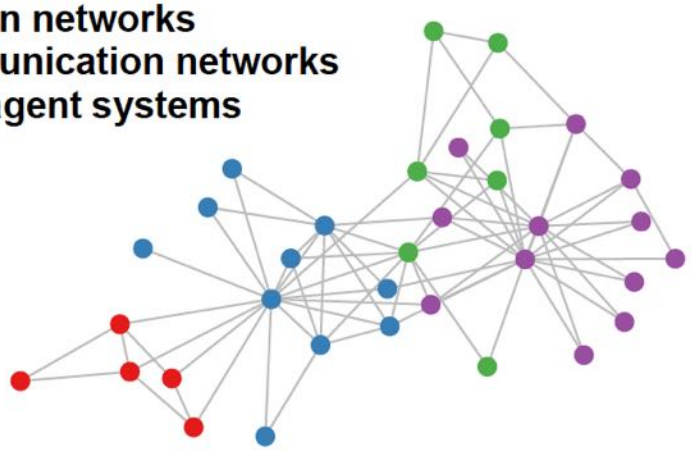
Molecules

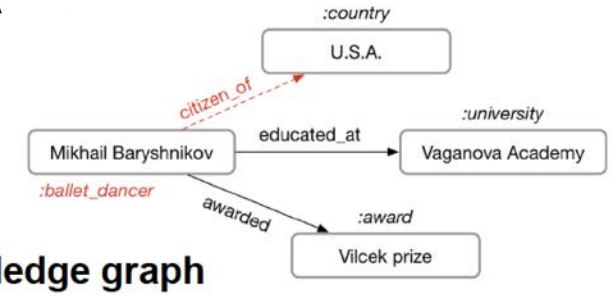
slide credit: Thomas Kipf

Graph-Structured / Relational Data

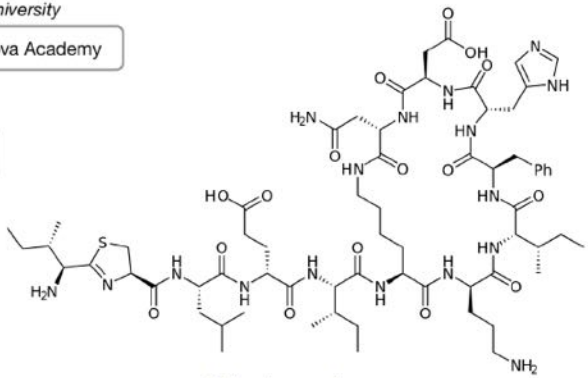
A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems

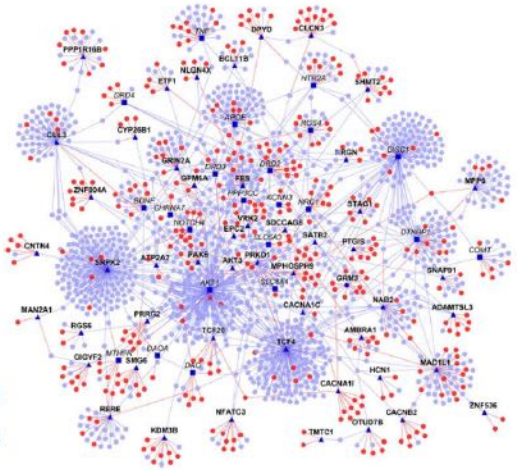




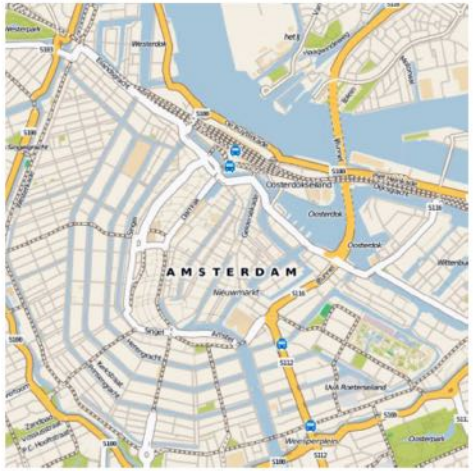
Knowledge graph



Molecules



Protein interaction networks

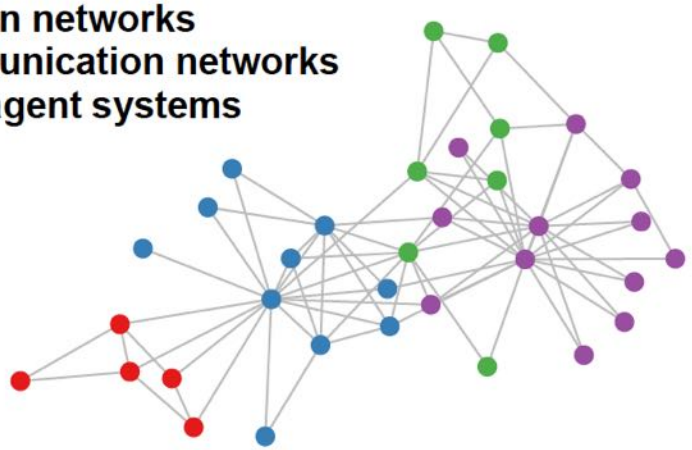


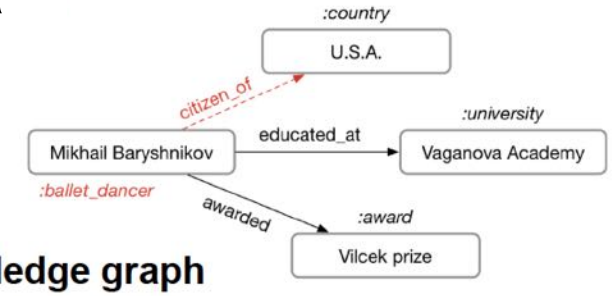
Road maps

Graph-Structured / Relational Data

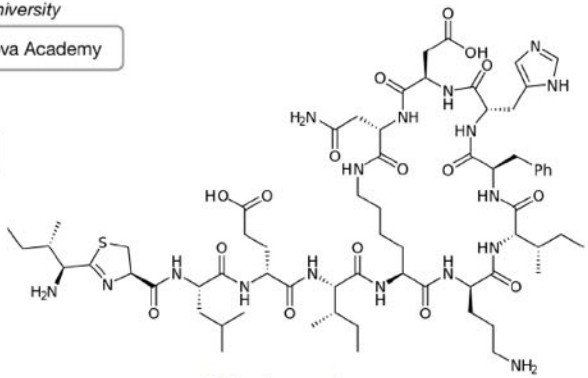
A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems

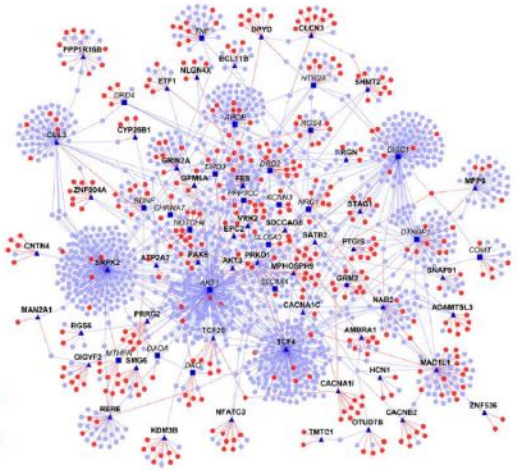




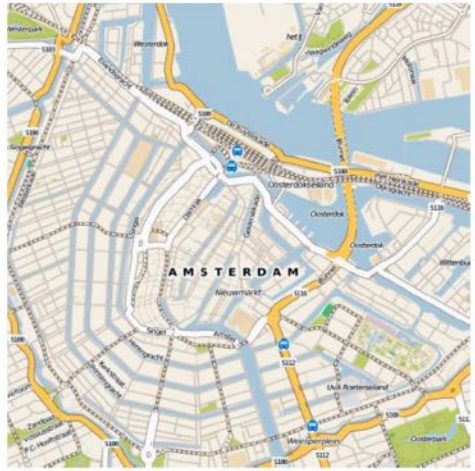
Knowledge graph



Molecules



Protein interaction networks

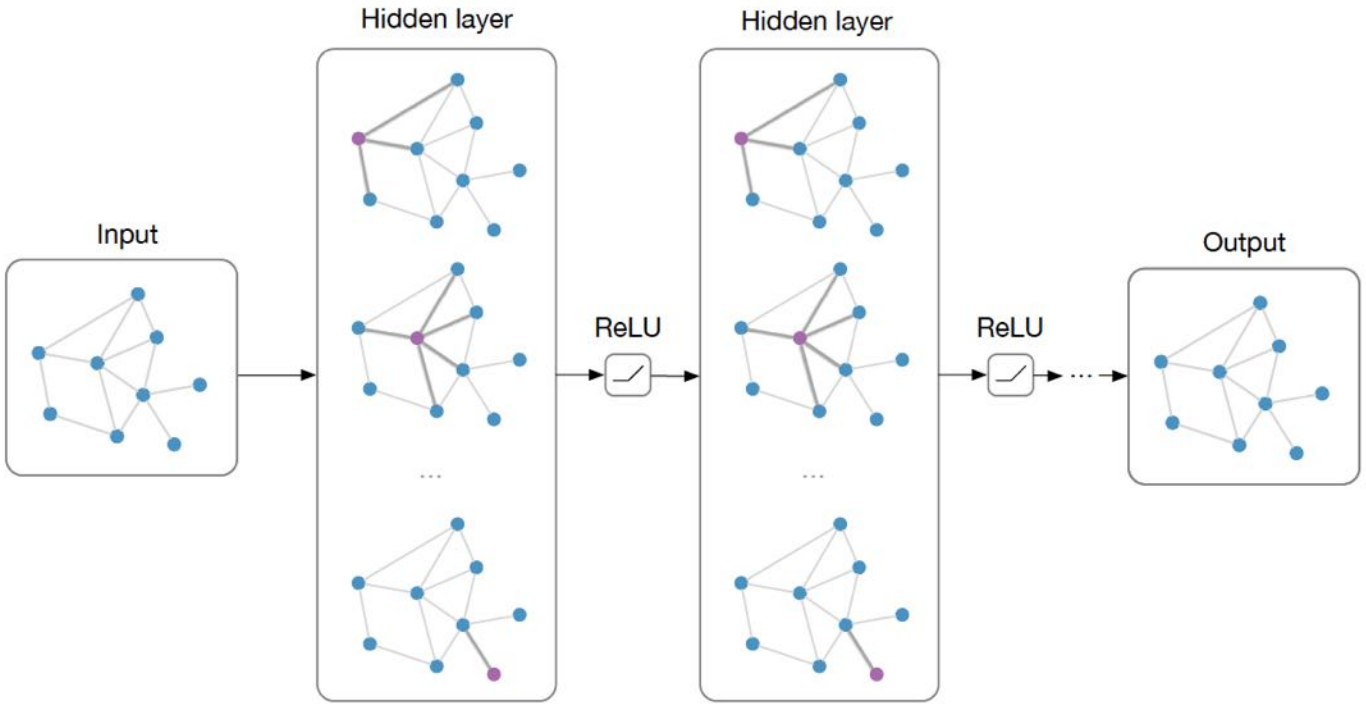


Road maps

Standard deep learning architectures like CNNs and RNNs don’t work here!

Graph Neural Networks

The bigger picture:



Main idea: Pass messages between pairs of nodes & agglomerate

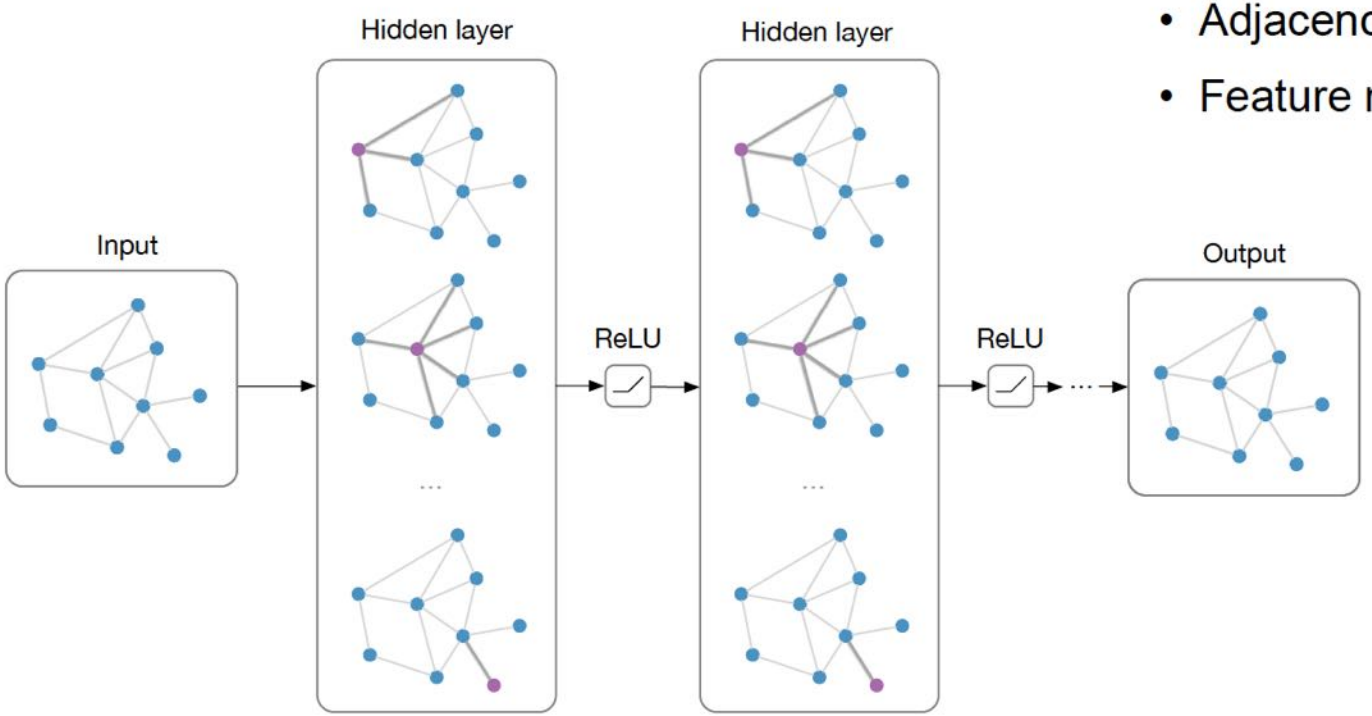
slide credit: Thomas Kipf

Graph Neural Networks

The bigger picture:

Notation: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ **or** $\mathcal{G} = (\mathbf{A}, \mathbf{X})$

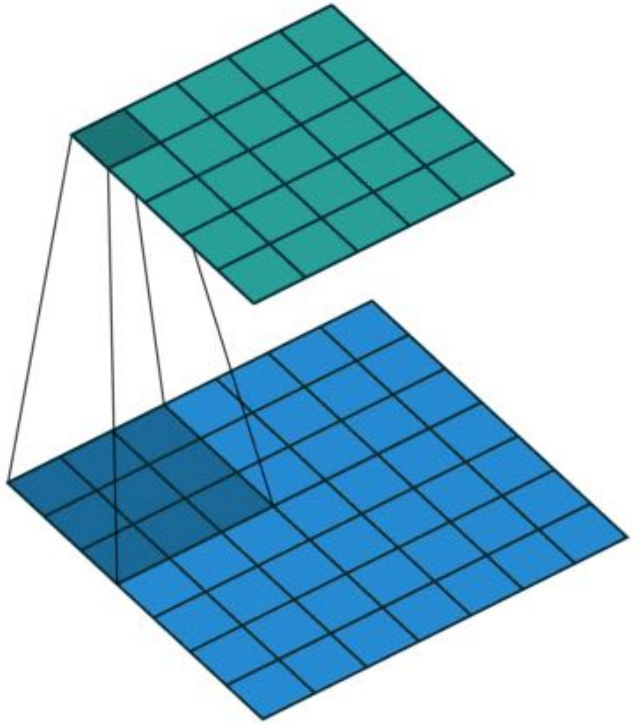
- Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$
- Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$



Main idea: Pass messages between pairs of nodes & agglomerate

slide credit: Thomas Kipf

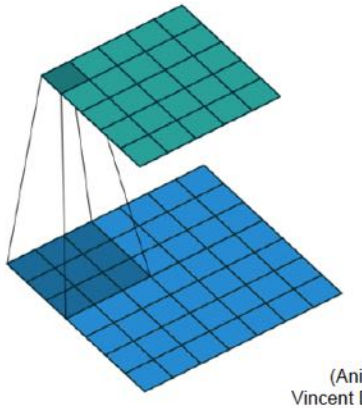
"Classic" Spatial Convolution Filter (3x3 here)



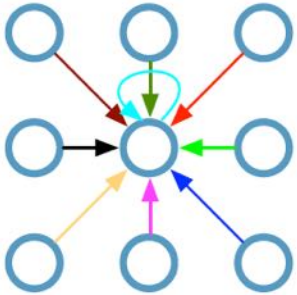
[1] Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning

Convolutional Neural Networks (on Grids)

Single CNN layer with 3x3 filter:



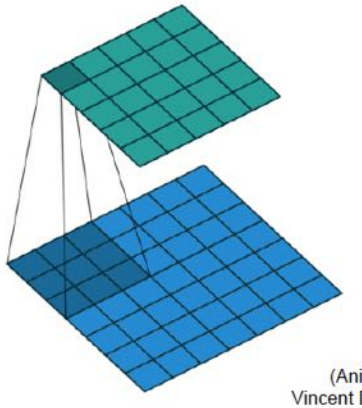
(Animation by Vincent Dumoulin)



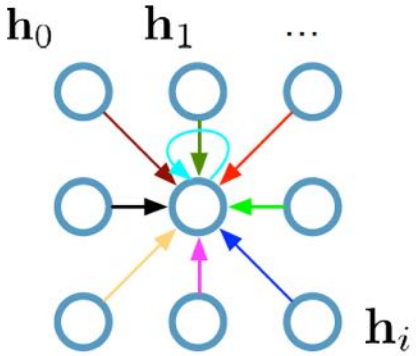
slide credit: Thomas Kipf

Convolutional Neural Networks (on Grids)

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)

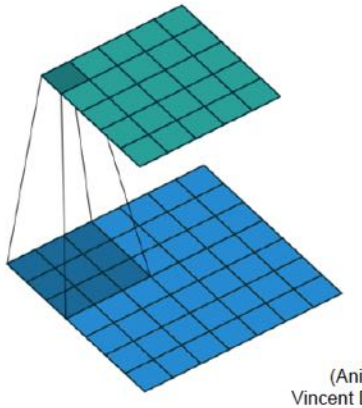


$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

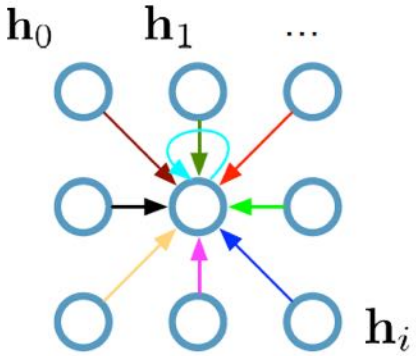
slide credit: Thomas Kipf

Convolutional Neural Networks (on Grids)

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



Update for a single pixel:

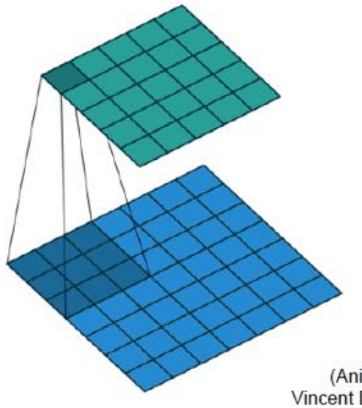
- Transform messages individually $\mathbf{W}_i\mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i\mathbf{h}_i$

$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

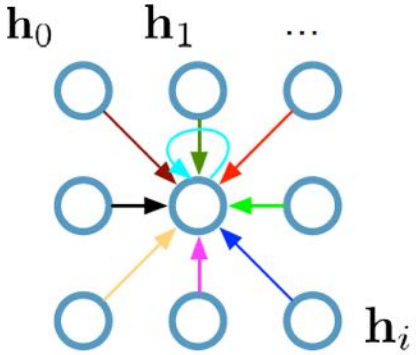
slide credit: Thomas Kipf

Convolutional Neural Networks (on Grids)

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



Update for a single pixel:

- Transform messages individually $\mathbf{W}_i \mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i \mathbf{h}_i$

$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Full update:

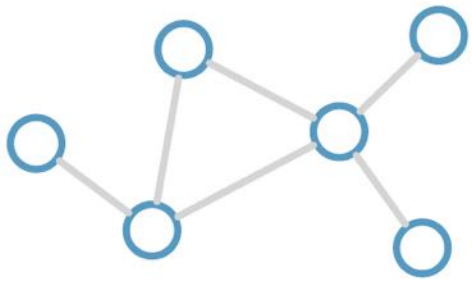
$$\mathbf{h}_4^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)}\mathbf{h}_1^{(l)} + \cdots + \mathbf{W}_8^{(l)}\mathbf{h}_8^{(l)}\right)$$

slide credit: Thomas Kipf

Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:

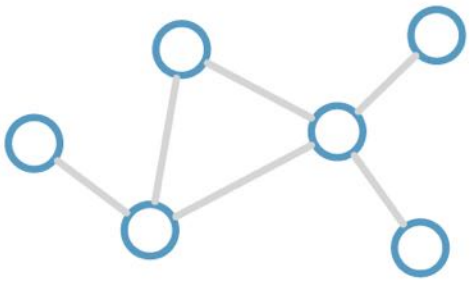


slide credit: Thomas Kipf

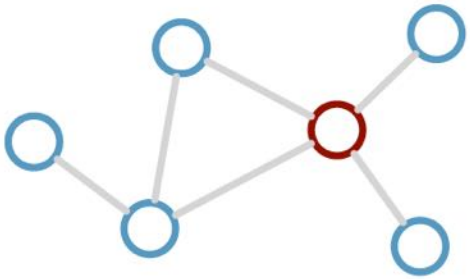
Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:

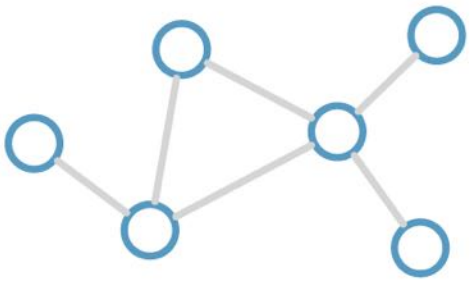


slide credit: Thomas Kipf

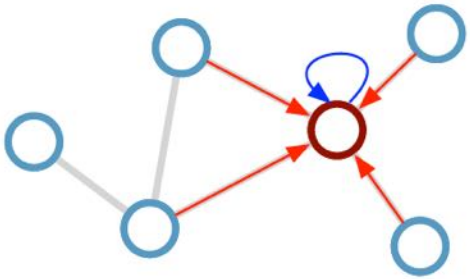
Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:

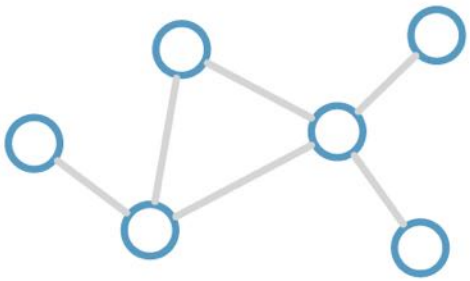


slide credit: Thomas Kipf

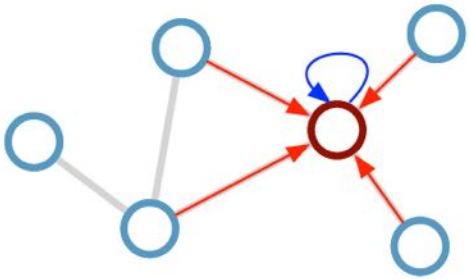
Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

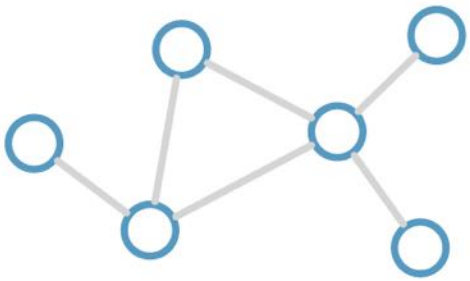
$\mathcal{N}_i$: neighbor indices $\quad$ c_{ij} : norm. constant (fixed/trainable)

slide credit: Thomas Kipf

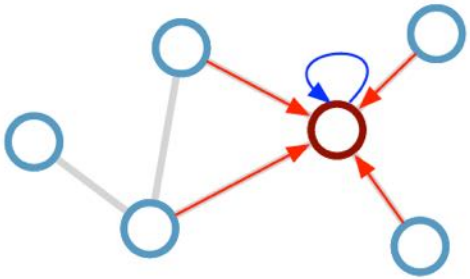
Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

Scalability: subsample messages [Hamilton et al., NIPS 2017]

Desirable properties:

- Weight sharing over all locations
- Invariance to permutations
- Linear complexity O(E)
- Applicable both in transductive and inductive settings

Limitations:

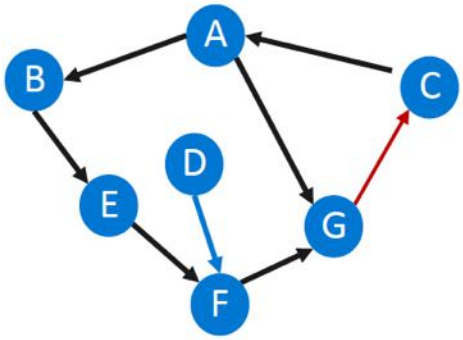
- Does not consider pairwise interactions
- No support for different edge types

$\mathcal{N}_i$: neighbor indices

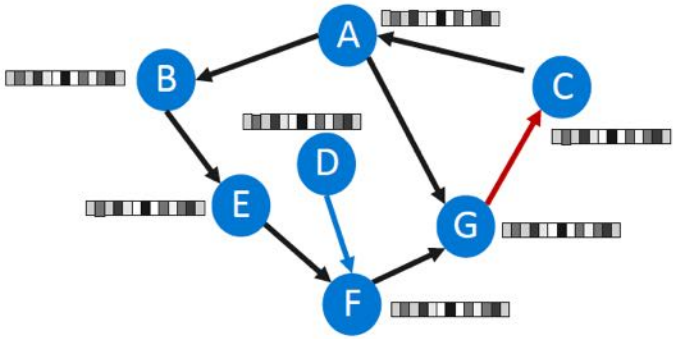
c_{ij} : norm. constant (fixed/trainable)

slide credit: Thomas Kipf

Graph Neural Networks



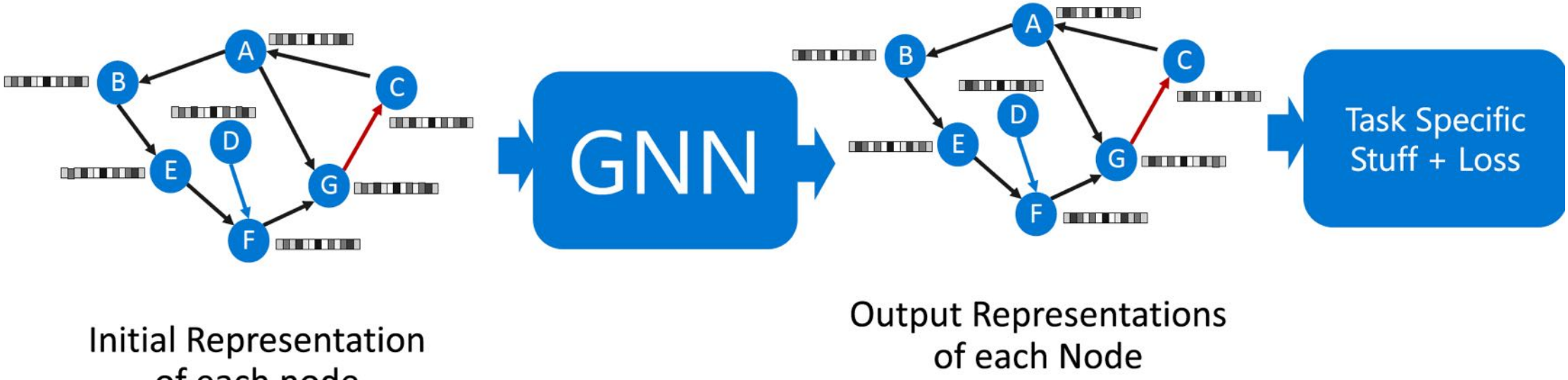
Graph Representation of Problem



Initial Representation of each node

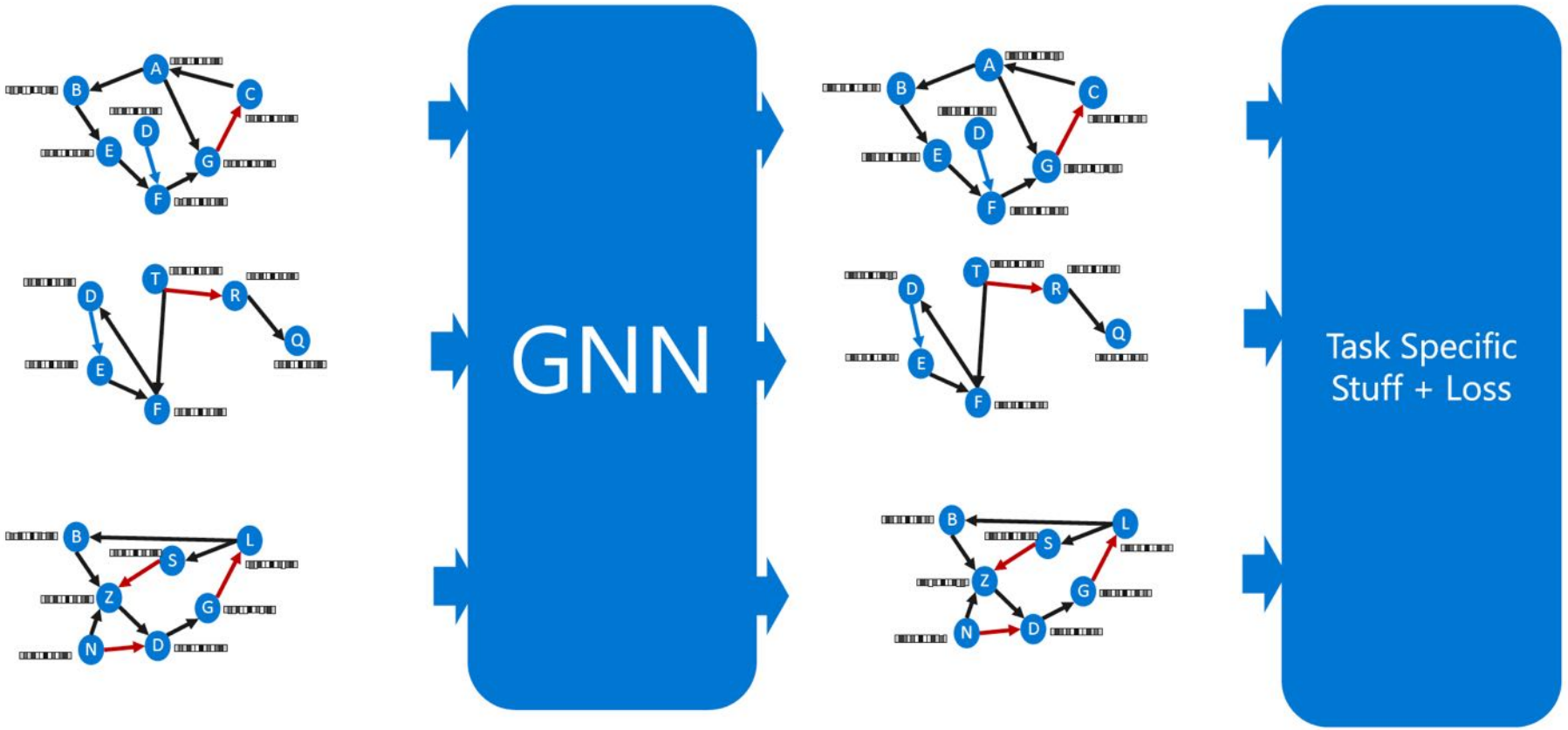
slide credit: Miltos Allamanis

Graph Neural Networks



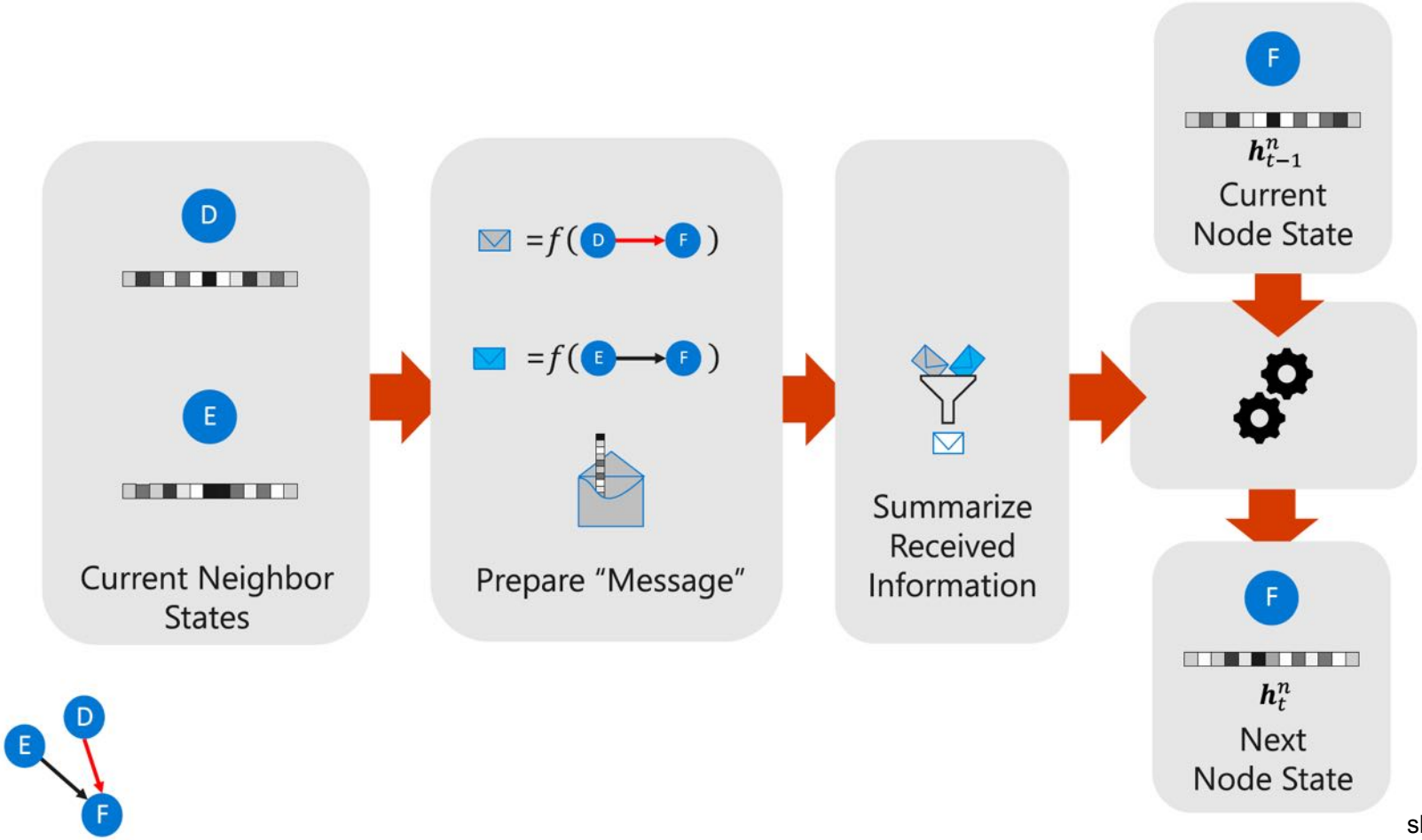
slide credit: Miltos Allamanis

Graph Neural Networks



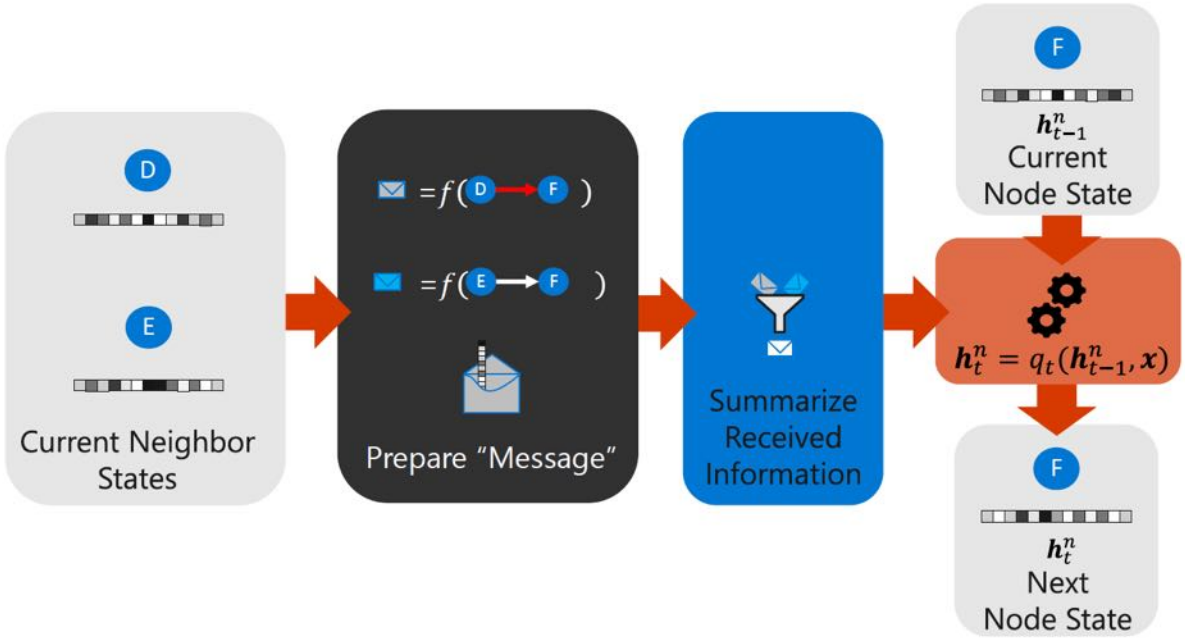
slide credit: Miltos Allamanis

Neural Message Passing



slide credit: Miltos Allamanis

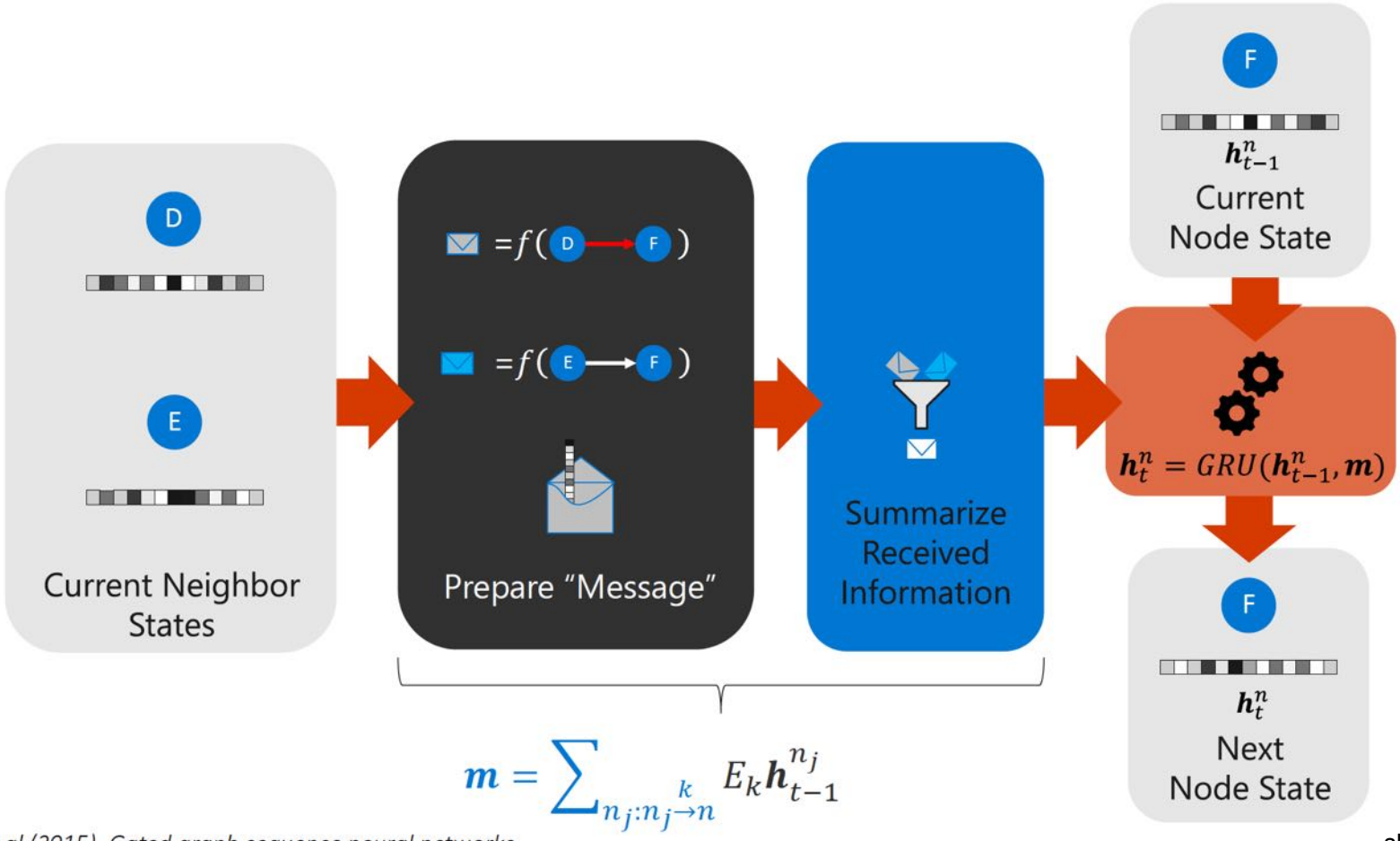
Neural Message Passing



$$h_t^n = q_t\left(h_{t-1}^n, \bigcup_{\substack{k \\ n_j : n_j \to n}} f_t\left(h_{t-1}^n, k, h_{t-1}^{n_j}\right)\right)$$

slide credit: Miltos Allamanis

Gated GNNs

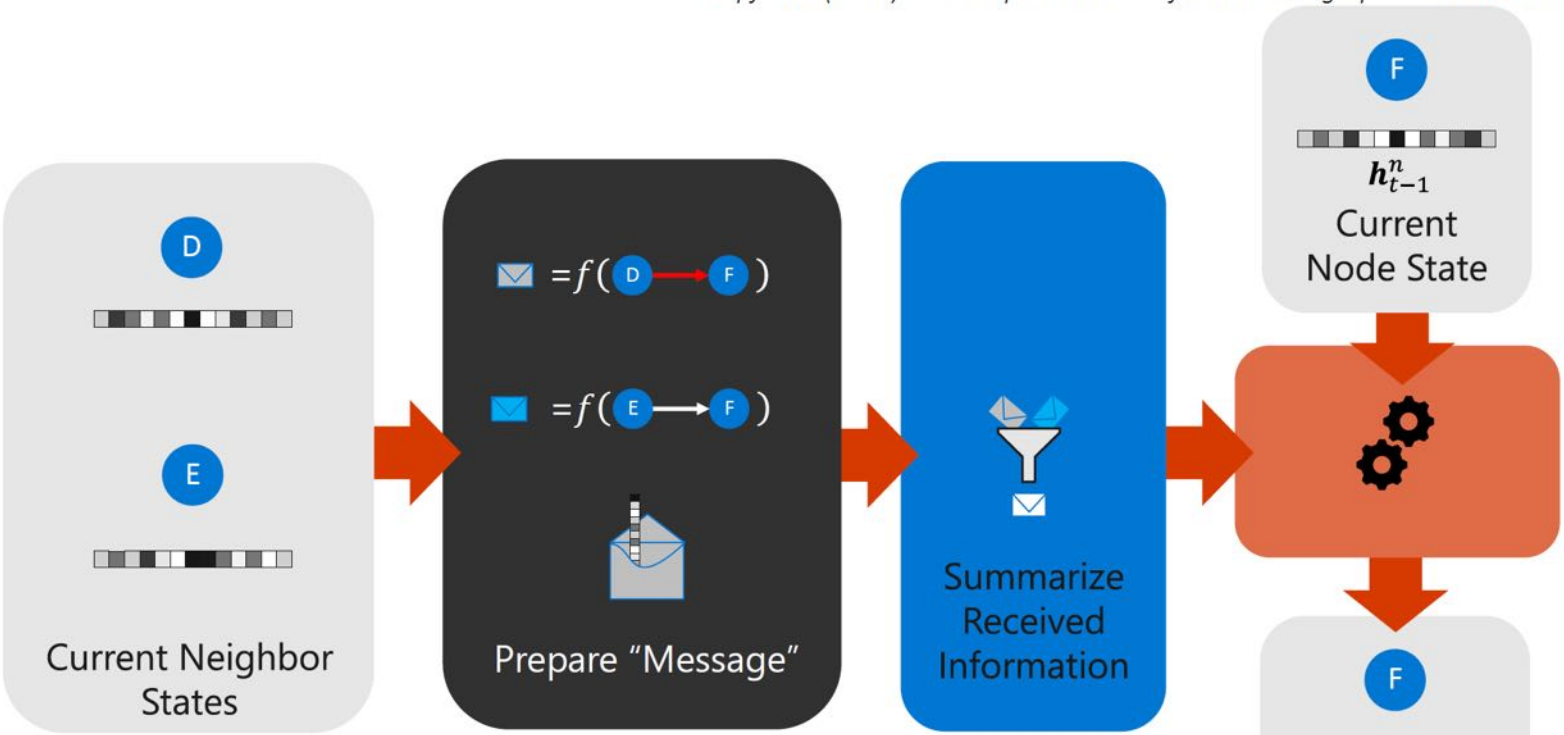


Li et al (2015). Gated graph sequence neural networks.

slide credit: Miltos Allamanis

Graph Convolutional Neural Networks (GCNs)

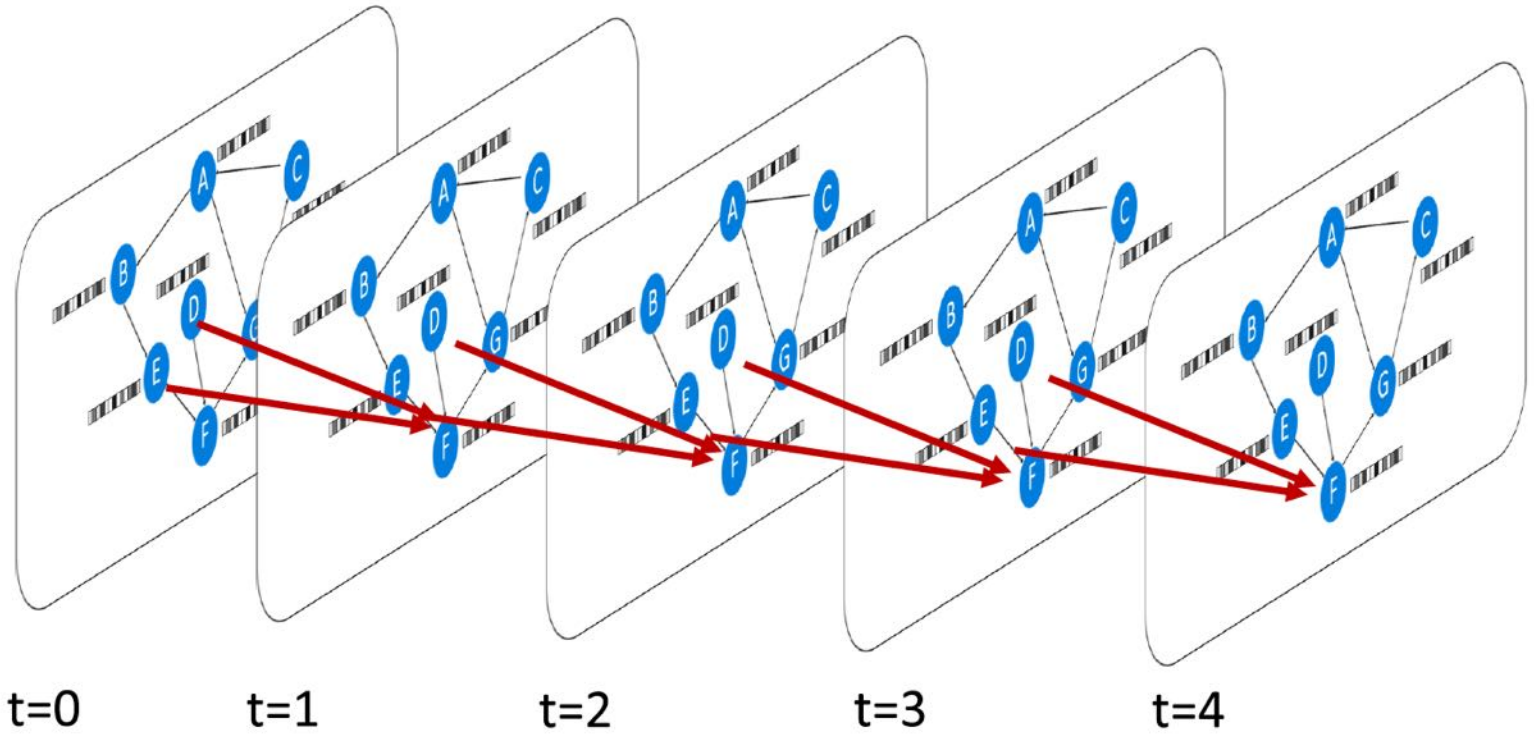
Kipf et al (2016). Semi-supervised classification with graph convolutional networks.



$$h_t^n = \sigma\left(\frac{1}{numNeighbors + 1} W_t \left(h_{t-1}^n + \sum_{n_j : n_j \rightarrow n} h_{t-1}^{n'}\right)\right)$$

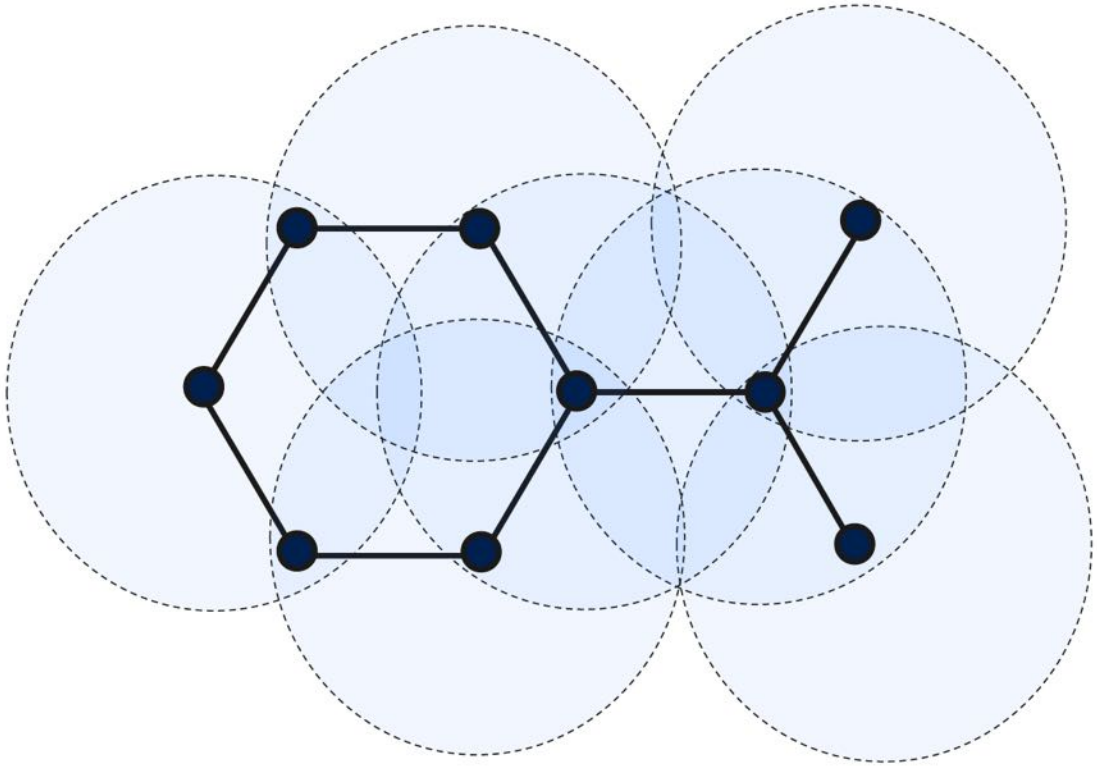
slide credit: Miltos Allamanis

Graph Neural Networks: Message Passing



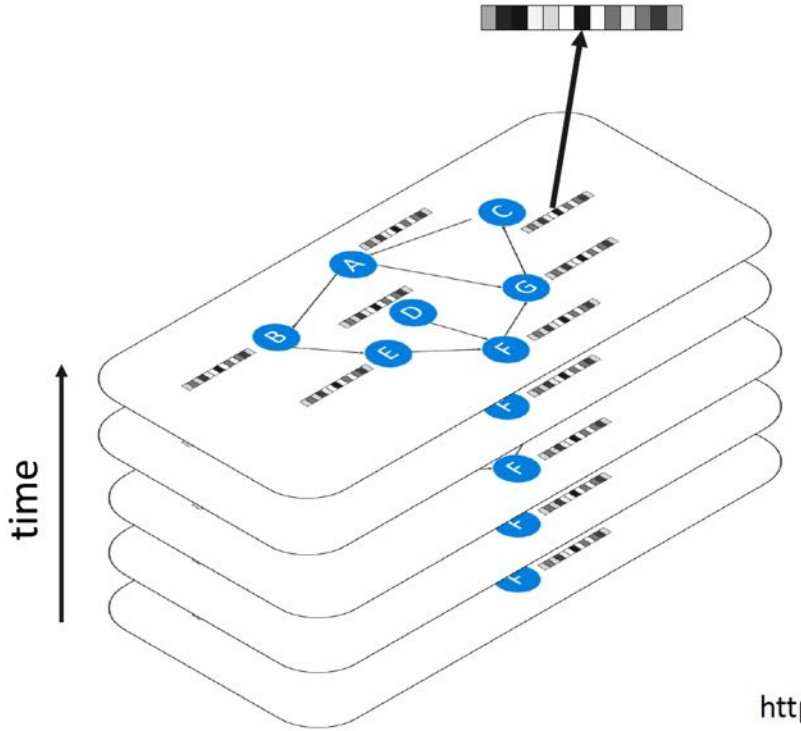
slide credit: Miltos Allamanis

Graph Neural Networks: Message Passing (synchronous - all to all)



slide credit: Miltos Allamanis

Graph Neural Networks: Output

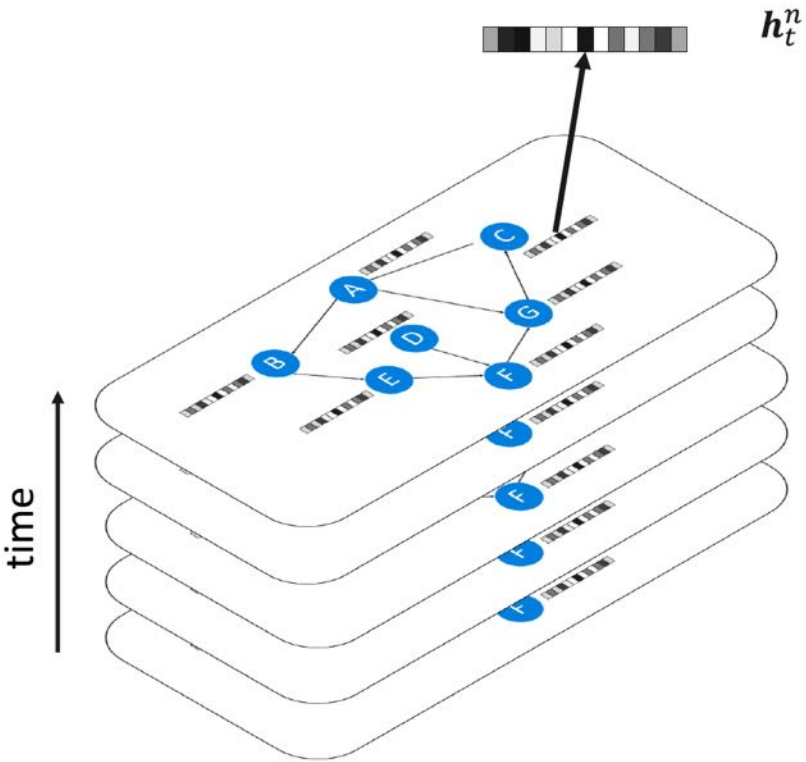


- node selection
- node classification
- graph classification

https://github.com/microsoft/tf-gnn-samples/

slide credit: Miltos Allamanis

Output Example: Binary Node Classification



$$x_n = \sigma\left(\boldsymbol{w}^T \boldsymbol{h}_t^n + b\right)$$

Binary cross entropy

$$\mathcal{L}(x_n, y_n) = y_n \cdot \log x_n + (1 - y_n) \log(1 - x_n)$$

slide credit: Miltos Allamanis

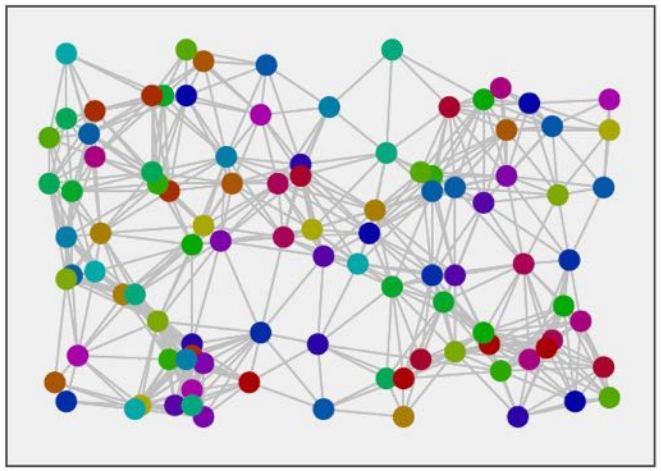
Overview Today's Lecture

- Motivation
  - Graph neural networks and neural message passing
  - Machine learning on graphs
- Convolutions in Time, in Space and on Graphs
- Graph Neural Networks and Graph Convolutional Networks
- Graphs and Graph Shift Operators

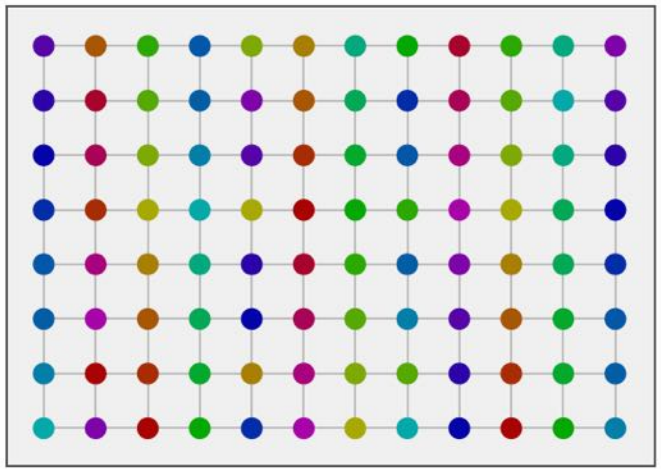
Neural Networks and Convolutional Neural Networks

- There is overwhelming empirical and theoretical justification to choose a neural network (NN)

Challenge is we want to run a NN over this



But we are good at running NNs over this



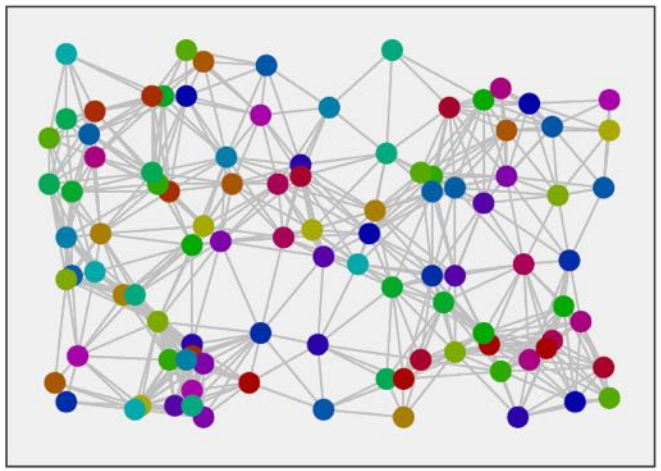
- Generic NNs do not scale to large dimensions $\Rightarrow$ Convolutional Neural Networks (CNNs) do scale

slide credit: Alejandro Ribeiro

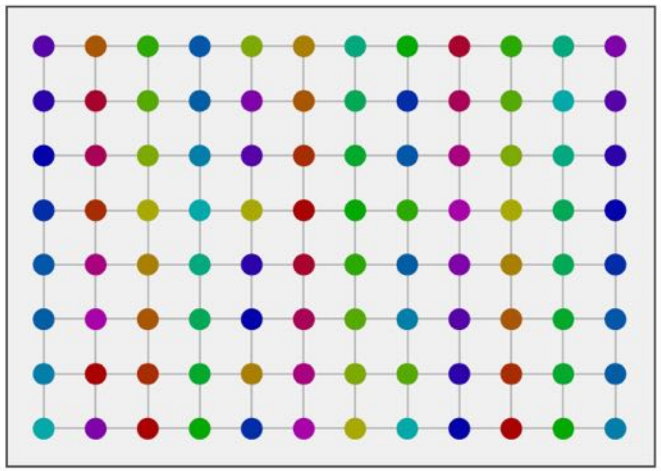
Convolutional Neural Networks and Graph Neural Networks

- CNNs are made up of layers composing convolutional filter banks with pointwise nonlinearities

Process graphs with graph convolutional NNs



Process images with convolutional NNs



- Generalize convolutions to graphs $\Rightarrow$ Compose graph filter banks with pointwise nonlinearities
- Stack in layers to create a graph (convolutional) Neural Network (GNN)

slide credit: Alejandro Ribeiro

Convolutions in Time, in Space, and on Graphs

- How do we generalize convolutions in time and space to operate on graphs?

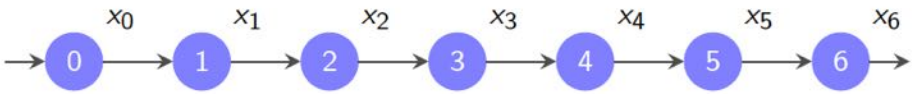
  $\Rightarrow$ Even though we do not often think of them as such, convolutions are operations on graphs

slide credit: Alejandro Ribeiro

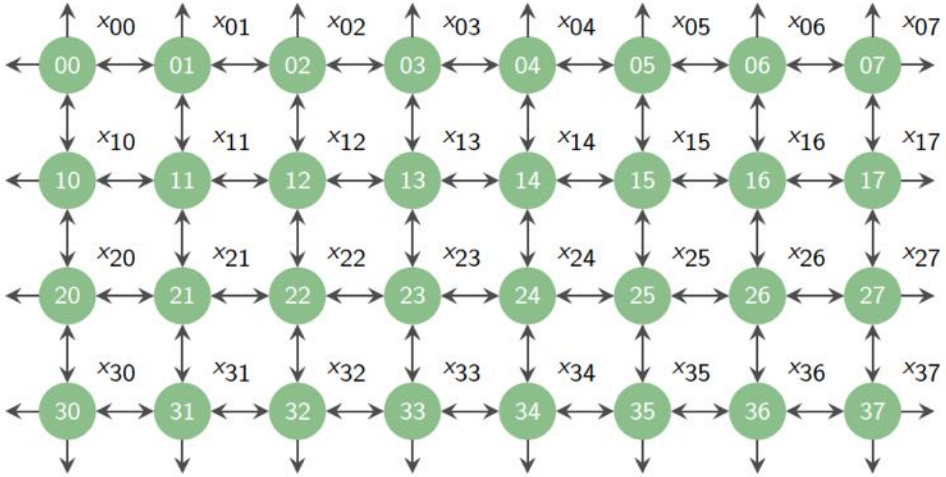
Time and Space can be Represented as Graphs

- We can describe discrete time and space using graphs that support time or space signals

Description of time with a directed line graph



Description of images (space) with a grid graph



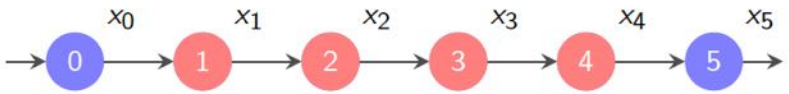
- Line graph represents adjacency of points in time. Grid graph represents adjacency of points in space

slide credit: Alejandro Ribeiro

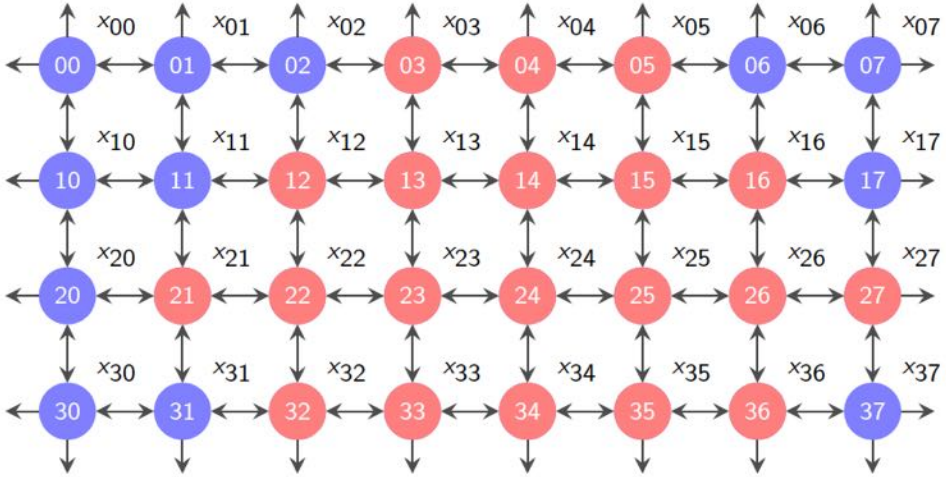
Convolutions in Time and Space

- Use line and grid graphs to write convolutions as polynomials on respective adjacency matrices $\mathbf{S}$

Description of time with a directed line graph



Description of images (space) with a grid graph



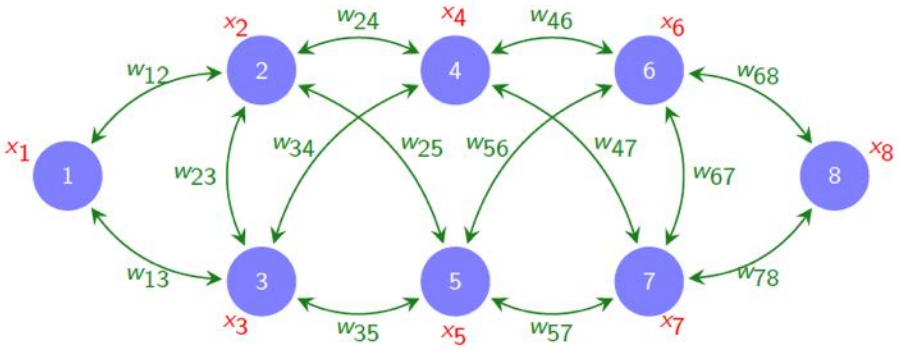
- Filter with coefficients h_k $\Rightarrow$ Output $\mathbf{z} = h_0\,\mathbf{S}^0\mathbf{x} + h_1\,\mathbf{S}^1\mathbf{x} + h_2\,\mathbf{S}^2\mathbf{x} + h_3\,\mathbf{S}^3\mathbf{x} + \ldots = \sum_{k=0}^{\infty} h_k\,\mathbf{S}^k\mathbf{x}$

slide credit: Alejandro Ribeiro

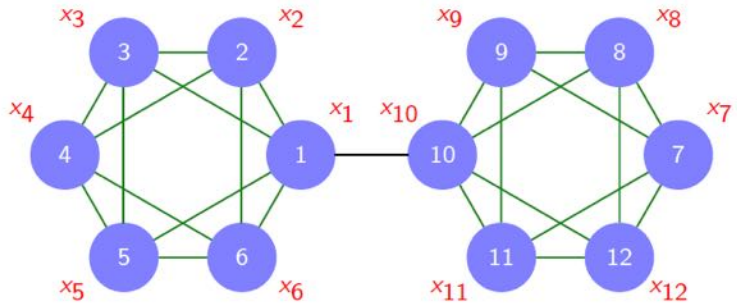
Arbitrary Graphs

- Time and Space are pervasive and important, but still a (very) limited class of signals
- Use graphs as generic descriptors of signal structure with signal values associated to nodes and edges expressing expected similarity between signal components

A signal supported on a graph



Another signal supported on another graph



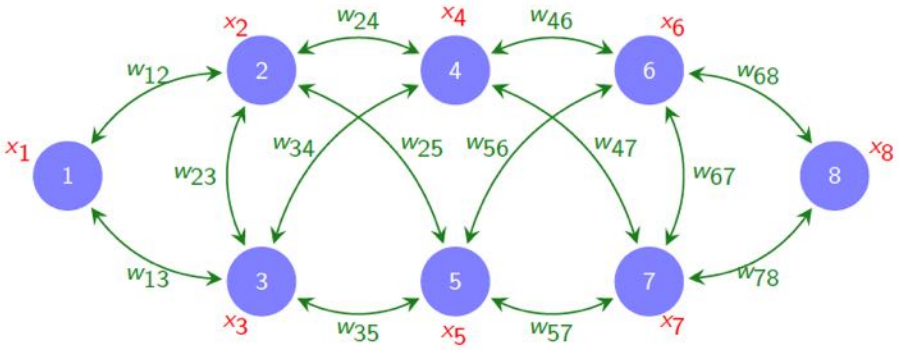
- Nodes are customers. Signal values are product ratings. Edges are cosine similarities of past scores

slide credit: Alejandro Ribeiro

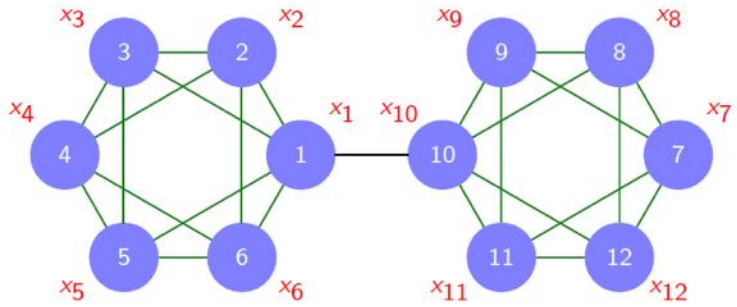
Arbitrary Graphs

- Time and Space are pervasive and important, but still a (very) limited class of signals
- Use graphs as generic descriptors of signal structure with signal values associated to nodes and edges expressing expected similarity between signal components

A signal supported on a graph



Another signal supported on another graph



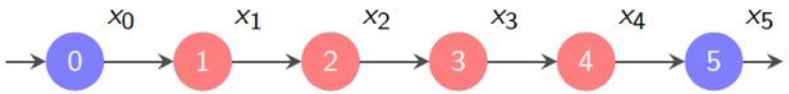
- Nodes are drones. Signal values are velocities. Edges are sensing and communication ranges

slide credit: Alejandro Ribeiro

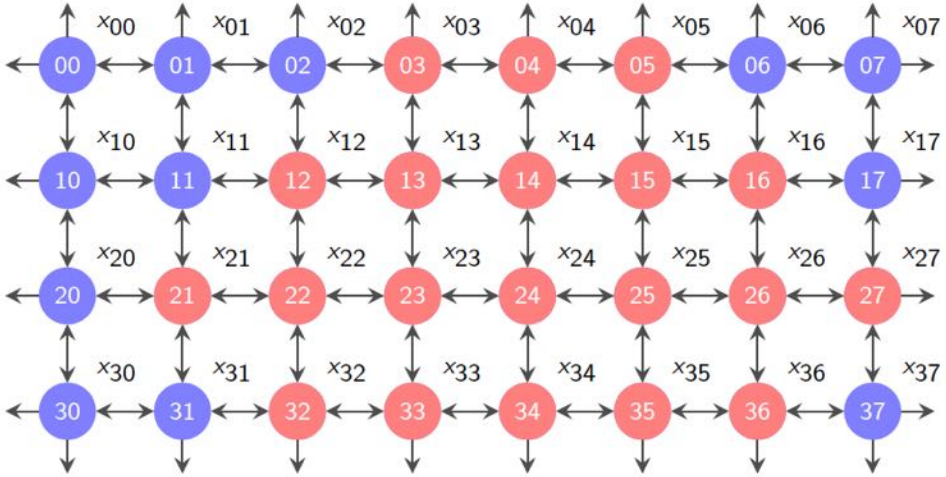
Convolutions on Graphs

- We've already seen that convolutions in time and space are polynomials on adjacency matrices

Description of time with a directed line graph



Description of images (space) with a grid graph



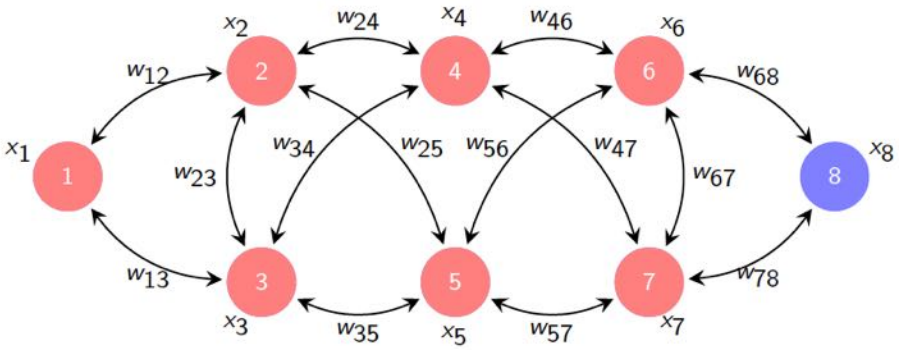
- Filter with coefficients h_k $\Rightarrow$ Output $\mathbf{z} = h_0 \mathbf{S}^0\mathbf{x} + h_1 \mathbf{S}^1\mathbf{x} + h_2 \mathbf{S}^2\mathbf{x} + h_3 \mathbf{S}^3\mathbf{x} + \ldots = \sum_{k=0}^{\infty} h_k \mathbf{S}^k\mathbf{x}$

slide credit: Alejandro Ribeiro

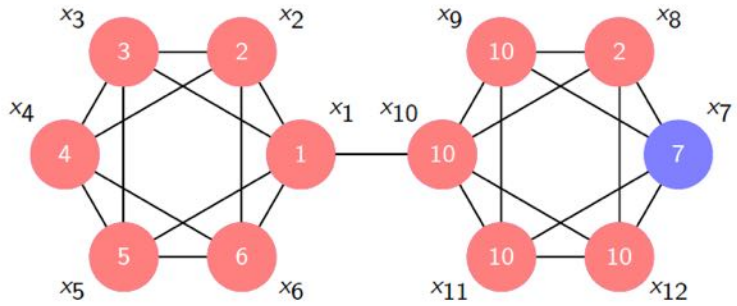
Convolutions on Graphs

- For graph signals we define graph convolutions as polynomials on matrix representations of graphs

A signal supported on a graph



Another signal supported on another graph



- Filter with coefficients h_k $\Rightarrow$ Output $\mathbf{z} = h_0 \mathbf{S}^0\mathbf{x} + h_1 \mathbf{S}^1\mathbf{x} + h_2 \mathbf{S}^2\mathbf{x} + h_3 \mathbf{S}^3\mathbf{x} + \ldots = \sum_{k=0}^{\infty} h_k \mathbf{S}^k\mathbf{x}$

- Graph convolutions share the locality of conventional convolutions. Recovered as particular case

Convolutional Neural Networks and Graph Neural Networks

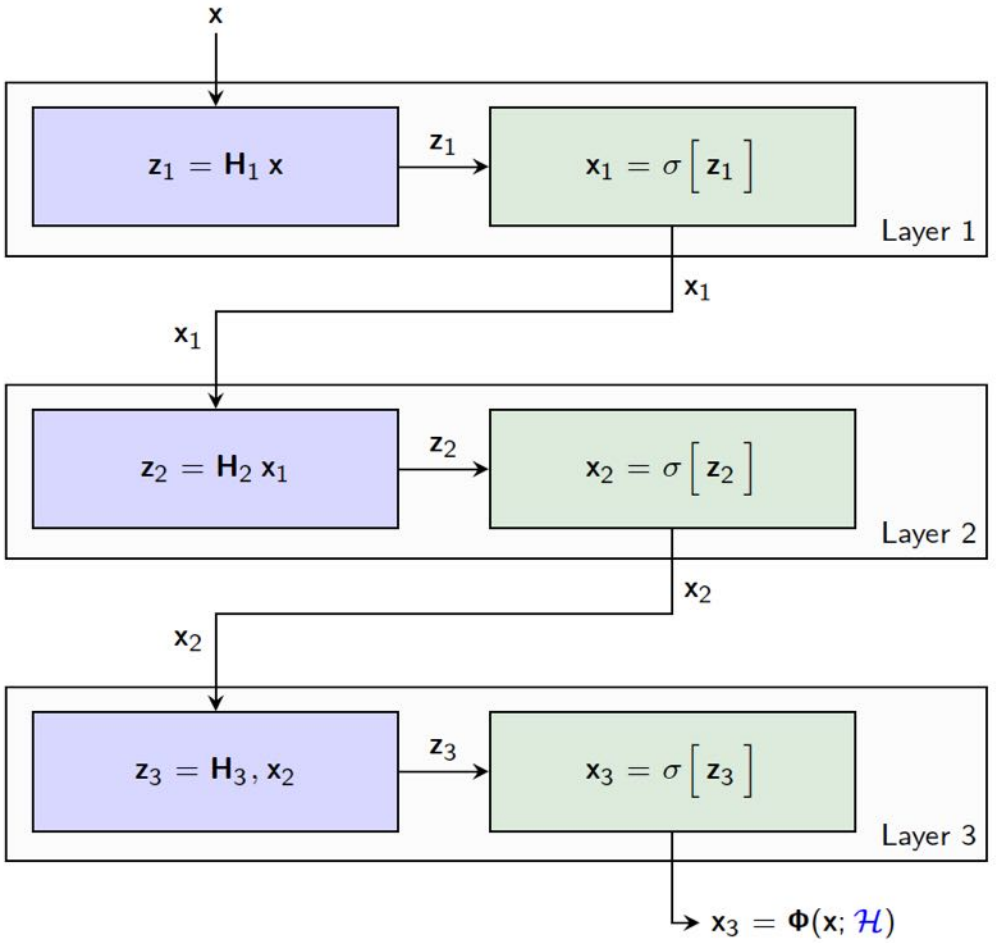
- CNNs and GNNe are minor variations of linear convolutional filters

  $\Rightarrow$ Compose filters with pointwise nonlinearities and compose these compositions into several layers

slide credit: Alejandro Ribeiro

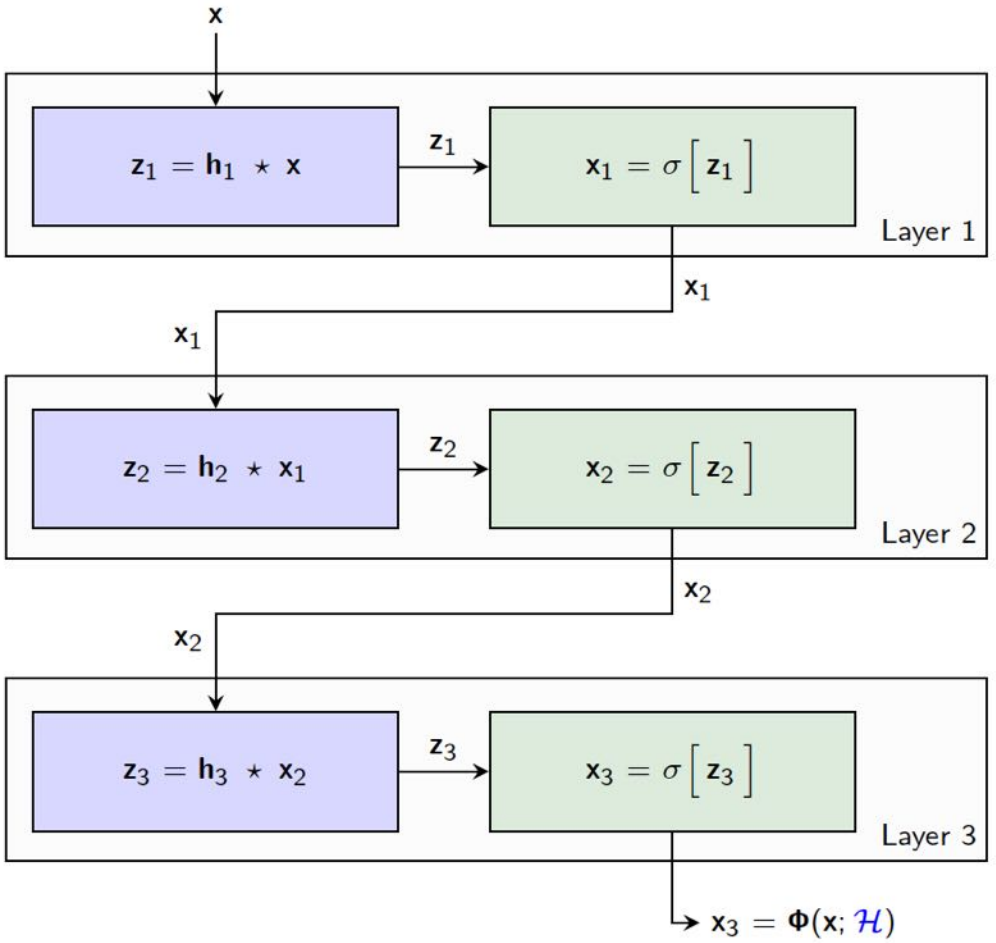
Neural Networks

- A neural network composes a cascade of layers
- Each of which are themselves compositions of linear maps with pointwise nonlinearities
- Does not scale to large dimensional signals $\mathbf{x}$



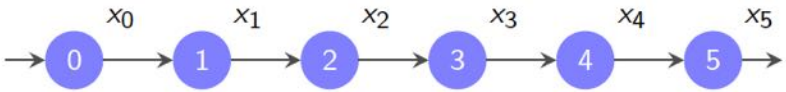
Convolutional Neural Networks (CNNs)

- A convolutional NN composes a cascade of layers
- Each of which are themselves compositions of convolutions with pointwise nonlinearities
- Scales well. The Deep Learning workhorse
- A CNNs are minor variation of convolutional filters
  - $\Rightarrow$ Just add nonlinearity and compose
  - $\Rightarrow$ They scale because convolutions scale

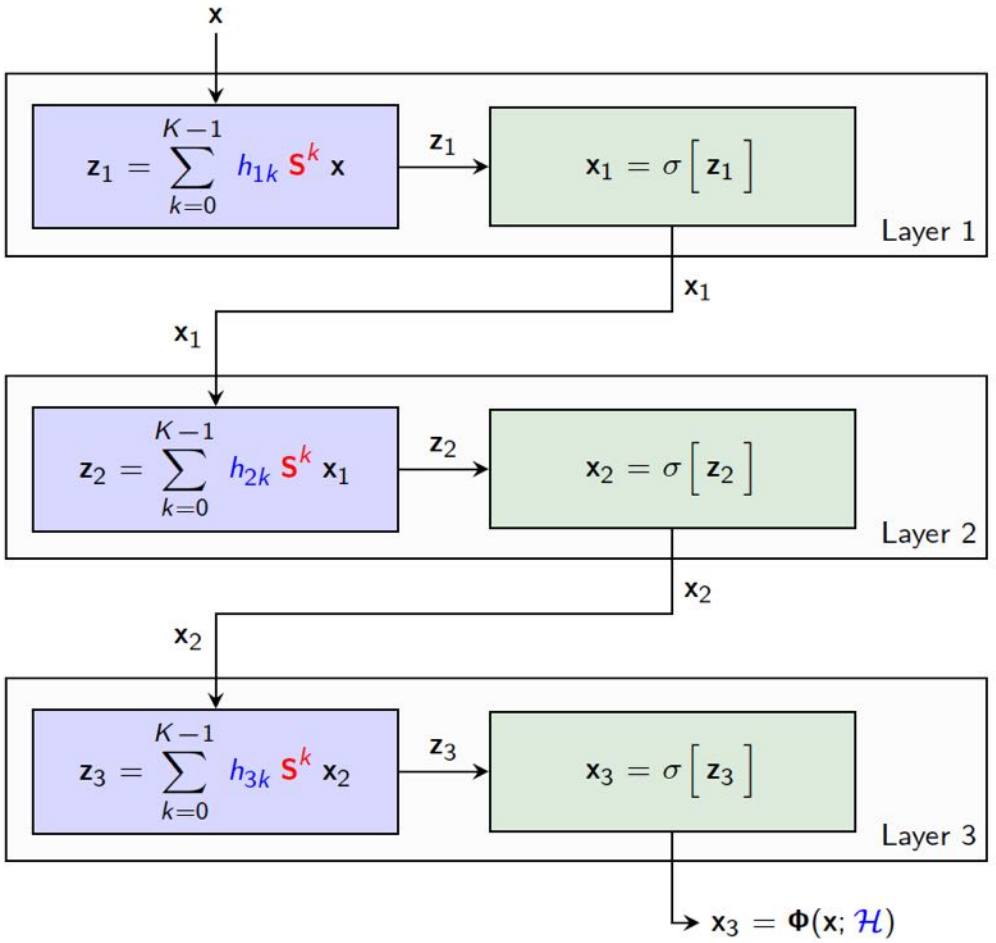


When we Think of Time Signal as Supported by a Line Graph

- Those convolutions are polynomials on the adjacency matrix of a line graph

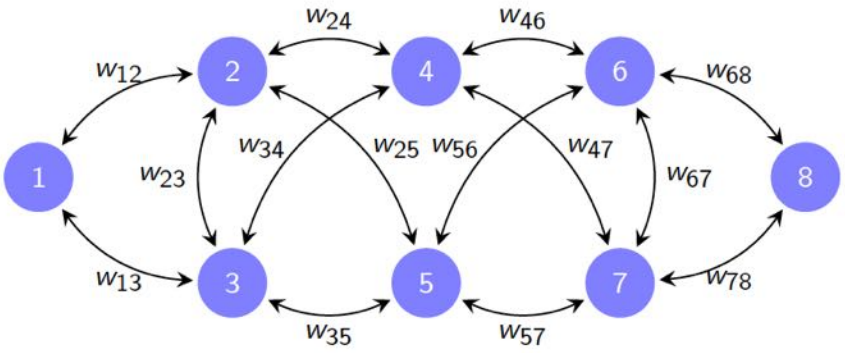


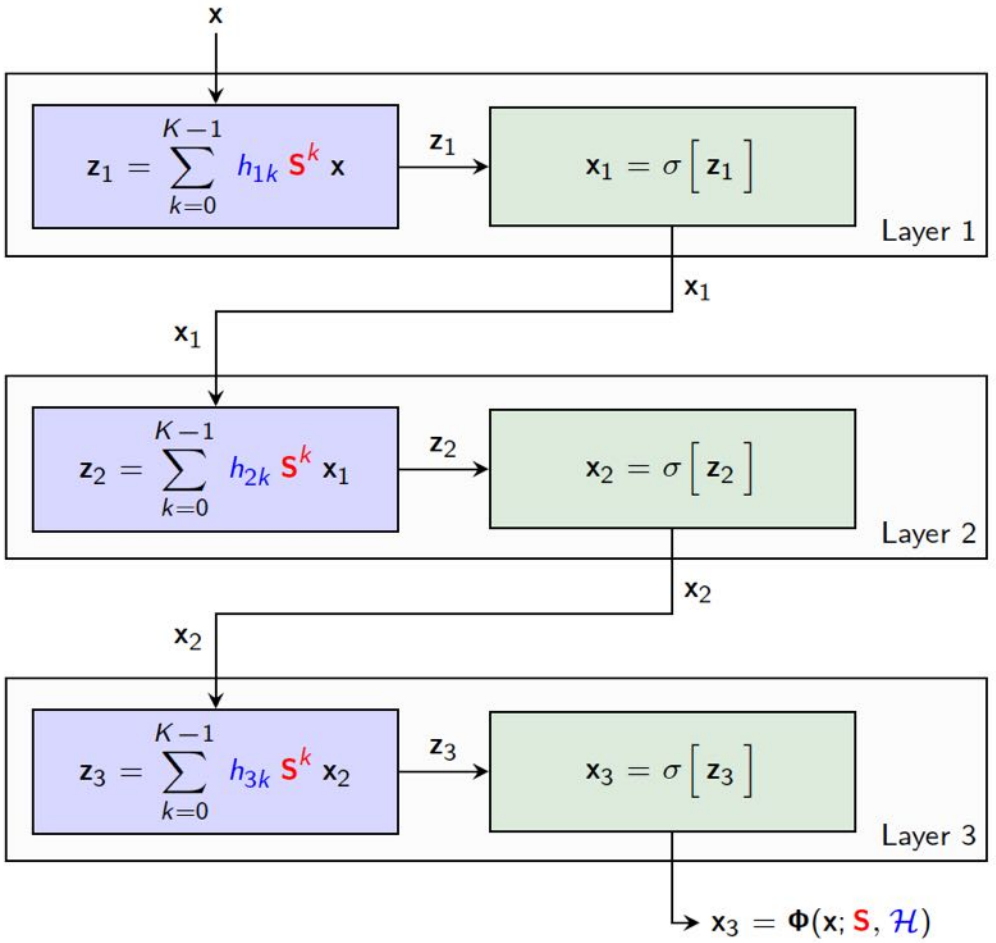
- Just another way of writing convolutions and Just another way of writing CNNs
- But one that lends itself to generalization



Graph Neural Networks (GNNs)

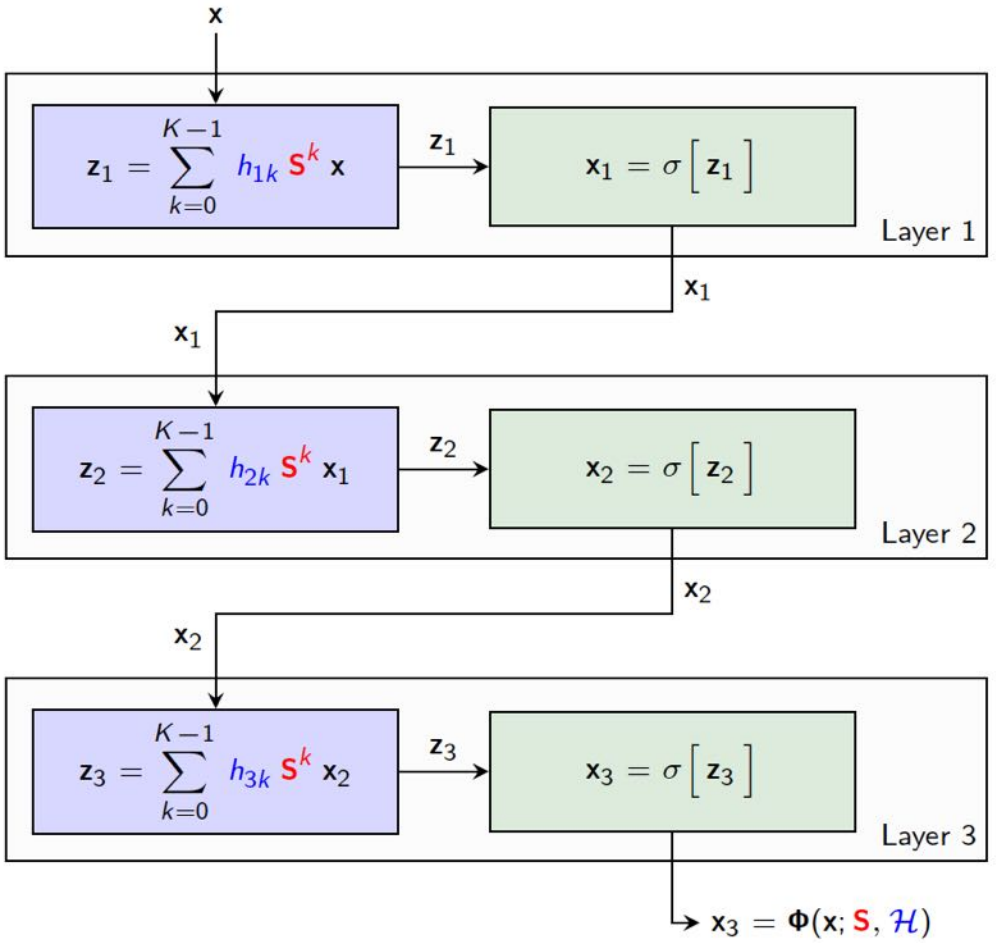
- The graph can be any arbitrary graph
- The polynomial on the matrix representation **S** becomes a graph convolutional filter





Graph Neural Networks (GNNs)

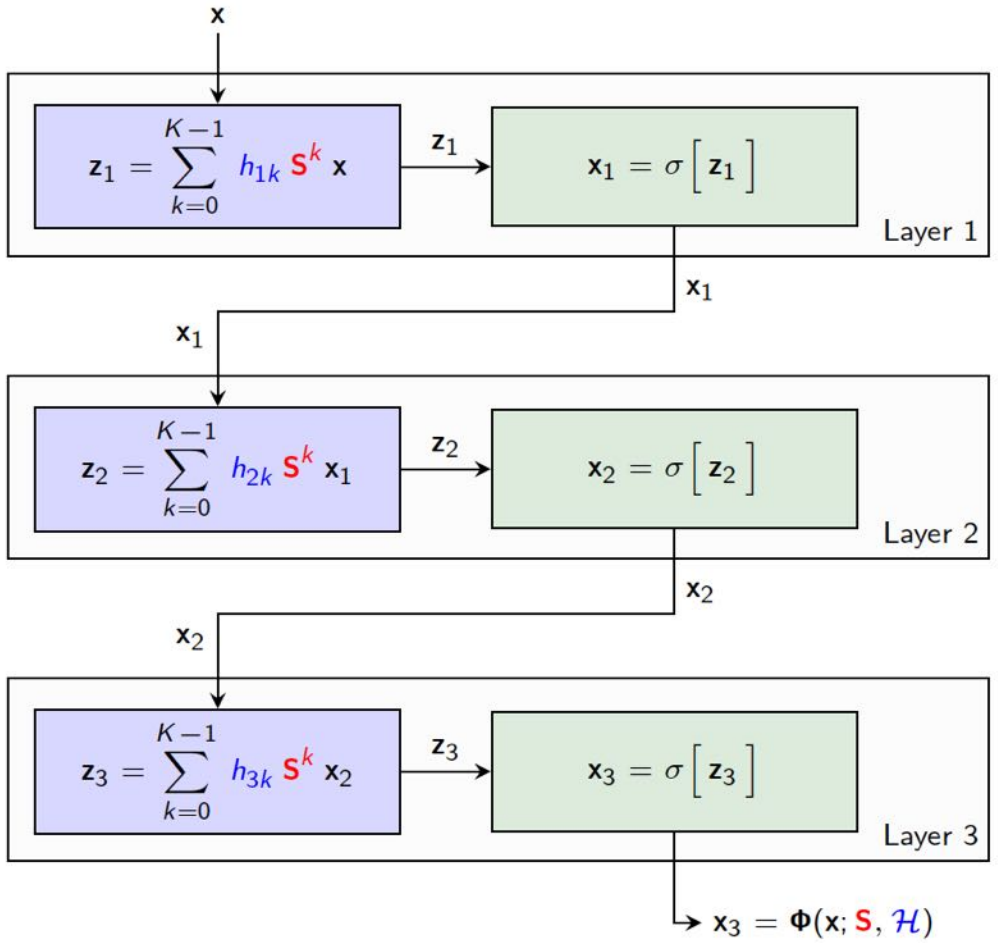
- A graph NN composes a cascade of layers
- Each of which are themselves compositions of graph convolutions with pointwise nonlinearities
- A NN with linear maps restricted to convolutions
- Recovers a CNN if $\mathbf{S}$ describes a line graph



Graph Neural Networks (GNNs)

- There is growing evidence of scalability.
- A GNN is a minor variation of a graph filter

  $\Rightarrow$ Just add nonlinearity and compose
- Both are scalable because they leverage the signal structure codified by the graph



The Road Ahead

slide credit: Alejandro Ribeiro

Objectives of This Part of the Course

Develop the ability to use Graph Neural Networks in practical applications

Understand the fundamental properties of Graph Neural Networks

Define Graph Neural Network Architectures

slide credit: Alejandro Ribeiro

Some Interim Remarks...

- I told you a lot about architectures today in the form of convolutions. Just to give you a taste.

  $\Rightarrow$ Don't worry if you didn't understand. Will revisit graph filters and graph neural networks

  $\Rightarrow$ We will also study graph recurrent neural networks

- Can't use blindly $\Rightarrow$ GNNs have properties that explain why they work. And why they don't

  $\Rightarrow$ Permutation Equivariance. Stability to deformations. Transferability

Ruiz-Gama-Ribeiro, *Graph Neural Networks: Architectures, Stability and Transferability,* PIEEE 2020, arxiv.org/pdf/2008.01767

slide credit: Alejandro Ribeiro

Overview Today's Lecture

- Motivation
  - Graph neural networks and neural message passing
  - Machine learning on graphs
- Convolutions in Time, Space and on Graphs
- Graph Neural Networks and Graph Convolutional Networks
- Graphs and Graph Shift Operators

Graphs

slide credit: Alejandro Ribeiro

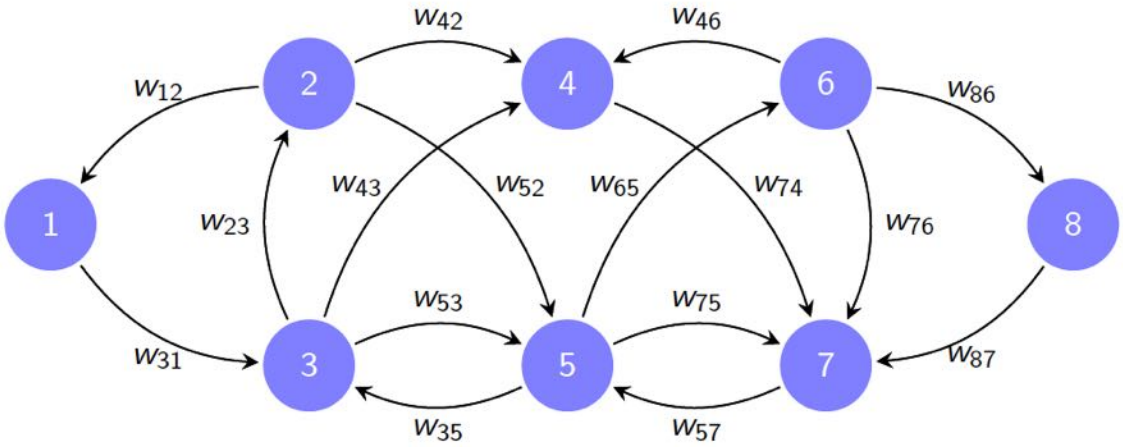
Nodes, Edges, Weights

- A graph is a triplet $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, which includes vertices $\mathcal{V}$, edges $\mathcal{E}$, and weights $\mathcal{W}$

  $\Rightarrow$ Vertices or nodes are a set of n labels. Typical labels are $\mathcal{V} = \{1, \ldots, n\}$

  $\Rightarrow$ Edges are ordered pairs of labels (i, j). We interpret $(i, j) \in \mathcal{E}$ as "i can be influenced by j."

  $\Rightarrow$ Weights $w_{ij} \in \mathbb{R}$ are numbers associated to edges (i, j). "Strength of the influence of j on i."



slide credit: Alejandro Ribeiro

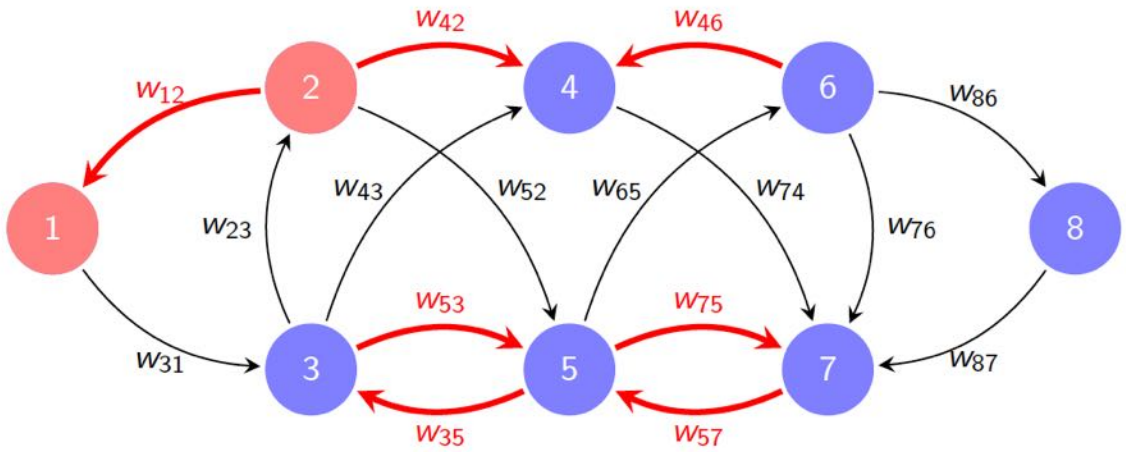
Directed Graphs

- Edge (i, j) is represented by an arrow pointing from j into i. Influence of node j on node i

  $\Rightarrow$ This is the opposite of the standard notation used in graph theory

- Edge (i, j) is different from edge (j, i) $\Rightarrow$ It is possible to have $(i, j) \in .\mathcal{E}$ and $(j, i) \notin \mathcal{E}$

- If both edges are in the edge set, the weights can be different $\Rightarrow$ It is possible to have $w_{ij} \neq w_{ji}$



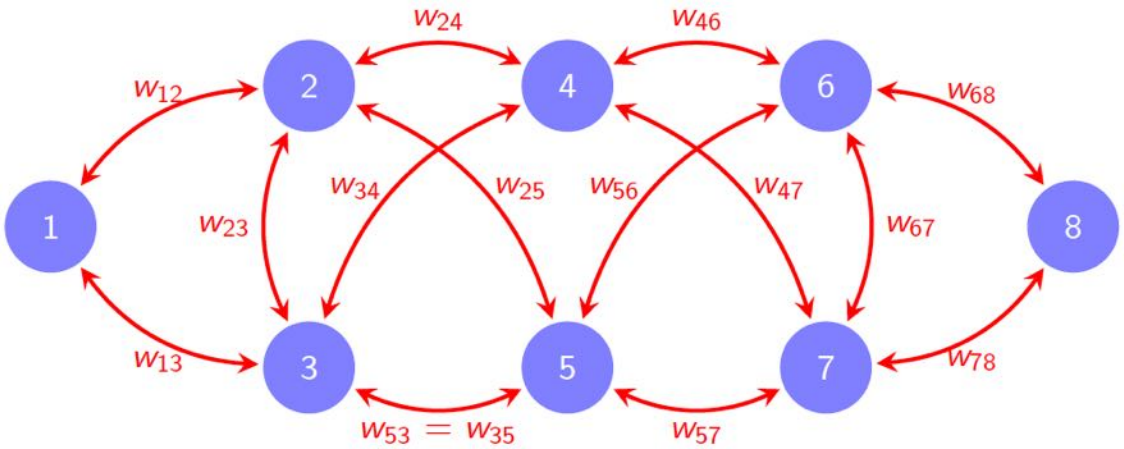
slide credit: Alejandro Ribeiro

Symmetric Graphs

- A graph is symmetric or undirected if both, the edge set and the weight are symmetric

  $\Rightarrow$ Edges come in pairs $\Rightarrow$ We have $(i,j) \in \mathcal{E}$ if and only if $(j,i) \in \mathcal{E}$

  $\Rightarrow$ Weights are symmetric $\Rightarrow$ We must have $w_{ij} = w_{ji}$ for all $(i,j) \in \mathcal{E}$



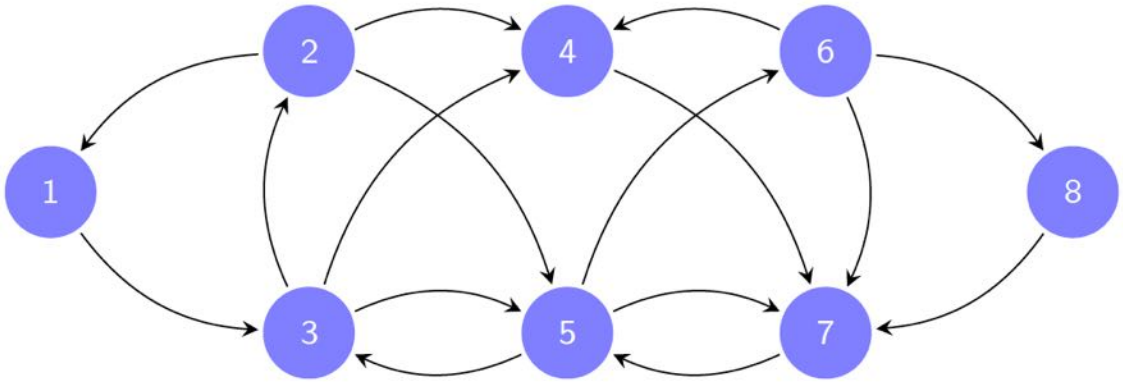
slide credit: Alejandro Ribeiro

Unweighted Graph

- A graph is unweighted if it doesn't have weights

  $\Rightarrow$ Equivalently, we can say that all weights are units $\Rightarrow$ $w_{ij} = 1$ for all $(i, j) \in \mathcal{E}$

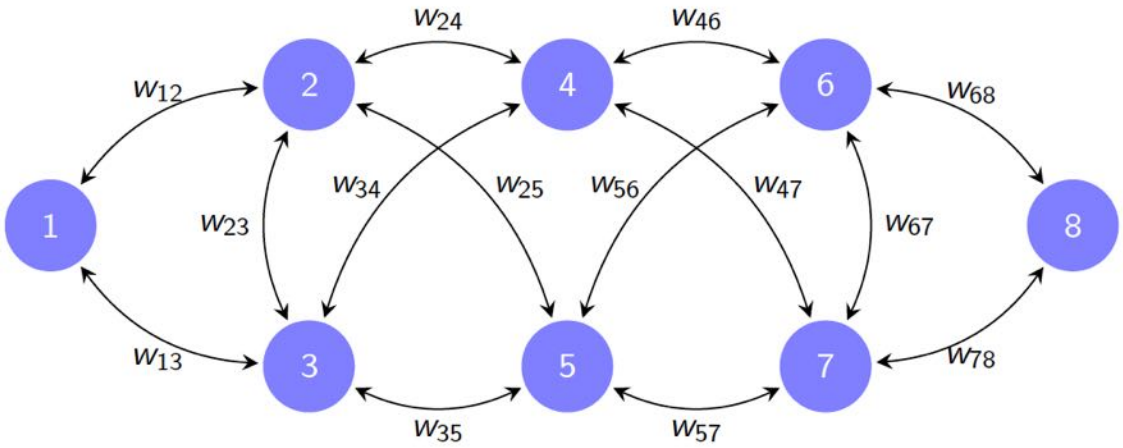
- Unweighted graphs could be directed or symmetric



slide credit: Alejandro Ribeiro

Weighted Symmetric Graph

- Graphs can be directed or symmetric. Separately, they can be weighted or unweighted.
- Most of the graphs we encounter in practical situations are symmetric and weighted



slide credit: Alejandro Ribeiro

Graph Shift Operators

- Graphs have matrix representations. Which in this course, we call graph shift operators (GSOs)

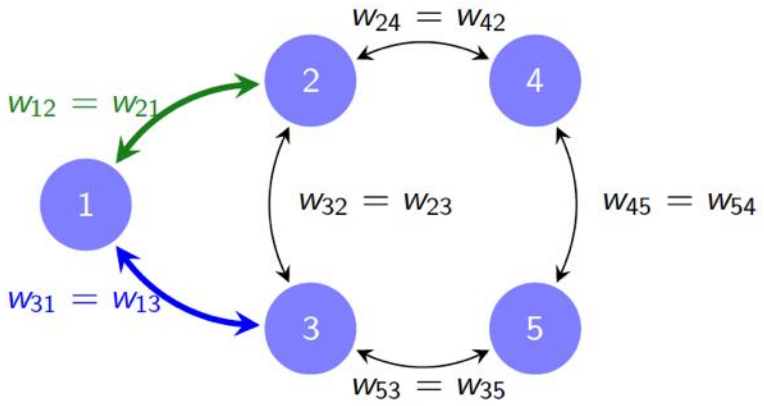
slide credit: Alejandro Ribeiro

Adjacency Matrix

- The adjacency matrix of graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ is the sparse matrix $\mathbf{A}$ with nonzero entries

$$A_{ij} = w_{ij}, \text{ for all } (i,j) \in \mathcal{E}$$

- If the graph is symmetric, the adjacency matrix is symmetric $\Rightarrow \mathbf{A} = \mathbf{A}^T$. As in the example



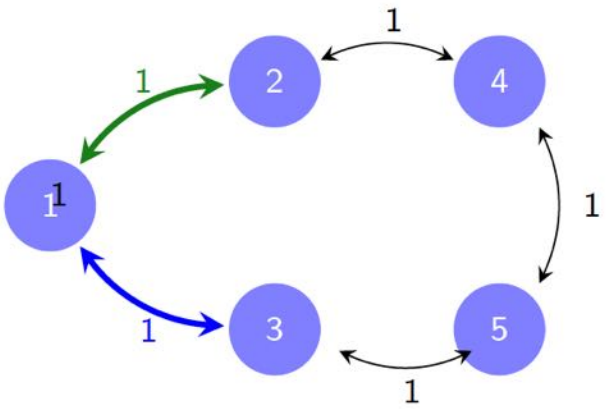
$$\mathbf{A} = \begin{bmatrix} 0 & w_{12} & w_{13} & 0 & 0 \\ w_{21} & 0 & w_{23} & w_{24} & 0 \\ w_{31} & w_{32} & 0 & 0 & w_{35} \\ 0 & w_{42} & 0 & 0 & w_{45} \\ 0 & 0 & w_{53} & w_{54} & 0 \end{bmatrix}.$$

slide credit: Alejandro Ribeiro

Adjacency Matrix for Unweighted Graph

- For the particular case in which the graph is unweighted. Weights interpreted as units

$$A_{ij} = 1, \qquad \text{for all} \quad (i,j) \in \mathcal{E}$$

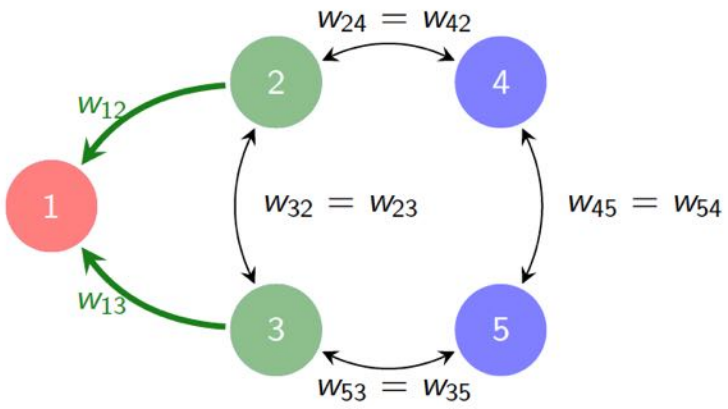


$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

slide credit: Alejandro Ribeiro

Neighborhood and Degree

- The neighborhood of node i is the set of nodes that influence i $\Rightarrow n(i) := \{j : (i,j) \in \mathcal{E}\}$
- Degree d_i of node i is the sum of the weights of its incident edges $\Rightarrow d_i = \sum_{j \in n(i)} w_{ij} = \sum_{j:(i,j)\in\mathcal{E}\}} w_{ij}$

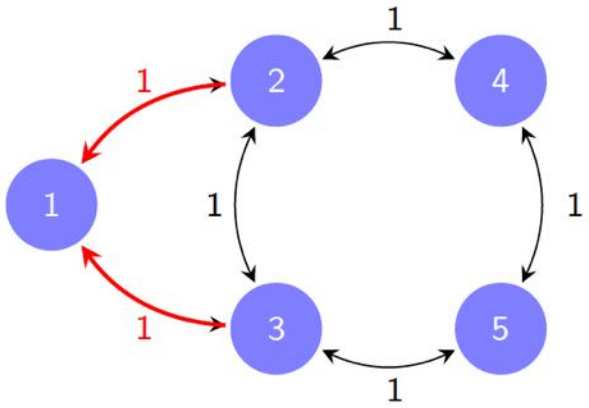


- Node 1 neighborhood $\Rightarrow n(1) = \{2, 3\}$
- Node 1 degree $\Rightarrow n(1) = w_{12} + w_{13}$

slide credit: Alejandro Ribeiro

Degree Matrix

- The degree matrix is a diagonal matrix $\mathbf{D}$ with degrees as diagonal entries $\Rightarrow D_{ii} = d_i$
- Write in terms of adjacency matrix as $\mathbf{D} = \text{diag}(\mathbf{A1})$. Because $(\mathbf{A1})_i = \sum_j w_{ij} = d_i$



$$\mathbf{D} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

slide credit: Alejandro Ribeiro

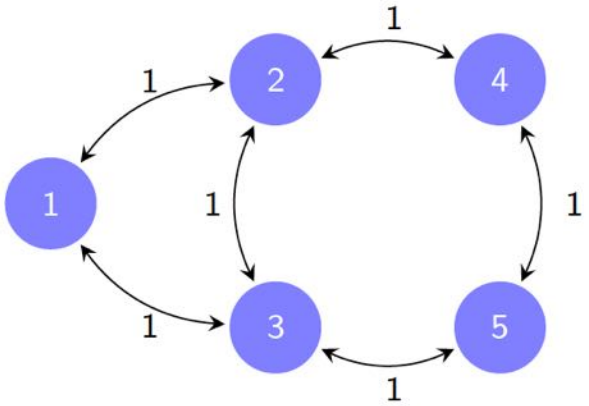
Laplacian Matrix

- The Laplacian matrix of a graph with adjacency matrix $\mathbf{A}$ is $\Rightarrow \mathbf{L} = \mathbf{D} - \mathbf{A} = \text{diag}(\mathbf{A1}) - \mathbf{A}$
- Can also be written explicitly in terms of graph weights $A_{ij} = w_{ij}$

  $\Rightarrow$ Off diagonal entries $\Rightarrow L_{ij} = -A_{ij} = -w_{ij}$

  $\Rightarrow$ Diagonal entries $\Rightarrow L_{ii} = d_i = \sum_{j \in n(i)} w_{ij}$

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 \\ 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix}$$



slide credit: Alejandro Ribeiro

Normalized Graph Representations

- Normalized adjacency and Laplacian matrices express weights relative to the nodes' degrees
- Normalized adjacency matrix $\Rightarrow \bar{\mathbf{A}} := \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ $\Rightarrow$ Results in entries $(\bar{\mathbf{A}})_{ij} = \frac{w_{ij}}{\sqrt{d_i d_j}}$
- The normalized adjacency is symmetric if the graph is symmetric $\Rightarrow \bar{\mathbf{A}}^T = \bar{\mathbf{A}}$.

slide credit: Alejandro Ribeiro

Normalized Graph Representations

- Normalized Laplacian matrix $\Rightarrow \bar{\mathbf{L}} := \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. Same normalization of adjacency matrix

- Given definitions normalized representations $\Rightarrow \bar{\mathbf{L}} = \mathbf{D}^{-1/2}\Big(\mathbf{D} - \mathbf{A}\Big)\mathbf{D}^{-1/2} = \mathbf{I} - \bar{\mathbf{A}}$

  $\Rightarrow$ The normalized Laplacian and adjacency are essentially the same linear transformation.

- Normalized operators are more homogeneous. The entries in the vector $\mathbf{A1}$ tend to be similar.

slide credit: Alejandro Ribeiro

Graph Shift Operator

- The Graph Shift Operator $\mathbf{S}$ is a stand in for any of the matrix representations of the graph

Adjacency Matrix	Laplacian Matrix	Normalized Adjacency	Normalized Laplacian
$\mathbf{S} = \mathbf{A}$	$\mathbf{S} = \mathbf{L}$	$\mathbf{S} = \bar{\mathbf{A}}$	$\mathbf{S} = \bar{\mathbf{L}}$

- If the graph is symmetric, the shift operator $\mathbf{S}$ is symmetric $\Rightarrow \mathbf{S} = \mathbf{S}^T$

- The specific choice matters in practice but most of results and analysis hold for any choice of $\mathbf{S}$

slide credit: Alejandro Ribeiro