Machine Learning for Graphs and Sequential Data

Graphs – Semi-Supervised Learning

Lecturer: Prof. Dr. Stephan Günnemann

www.daml.in.tum.de

Summer Term 2020

Data Analytics and
Machine Learning

Roadmap

- **Chapter: Graphs**
  1. Graphs & Networks
  2. Generative Models
  3. Clustering
  4. Node Embeddings
  5. Ranking
  6. **Semi-Supervised Learning**
     - **Label Propagation**
     - Graph Neural Networks
  7. Limitations of GNNs

Types of Machine Learning Problems on Graphs

- So far we have discussed unsupervised learning problems on graphs
  - generative modeling
  - clustering / community detection
  - node embeddings
  - ranking

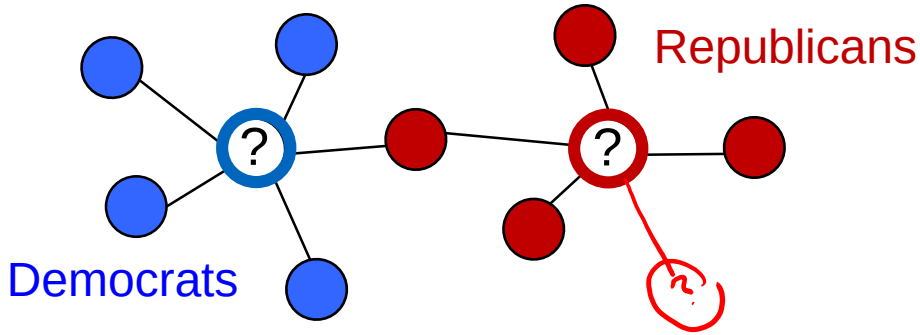
- What about supervised learning tasks, such as
  - classifying role of a protein in a PPI network
  - detecting fraudsters in an e-commerce system
  - predicting user's preferences in a social network

  assuming that some training data is given

- More generally, how do we label/categorize/classify instances in a graph?

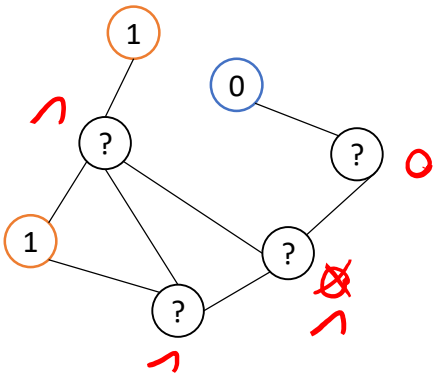
Collective Classification

- Consider the following problem
  - Graph represents a social network, nodes = users, edges = friendship
  - Labels are known for some **labeled nodes**
  - Goal is to classify the **unlabeled nodes**

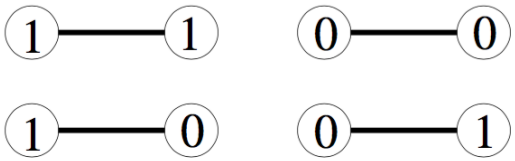


- Standard assumption: **Homophily** (a.k.a. **assortativity**)
  - *"birds of a feather flock together"*
  - *a.k.a. smoothness assumptions*
  - *that is, if nodes are connected by an edge, they are likely to have same labels*

Label Propagation



- Consider the binary case (two classes)
- Formal definition of the problem
  - Nodes $V = S \cup U$
    - Labeled instances S (seeds) and unlabeled instances U
  - Symmetric weighted adjacency matrix $\boldsymbol{W} \in \mathbb{R}^{|V| \times |V|}$
    - $w_{ij} \geq 0$ denotes similarity of nodes i and j
  - $\hat{y}_i \in \{0,1\}$ for $i \in S$ // **given** class labels for the nodes in S
  - $y_i \in \{0,1\}$ for $i \in V$ // class labels we want to **predict** for each node
- Idea: **Energy minimization** $E(\boldsymbol{y}) = \frac{1}{2} \sum_{ij} w_{ij} (y_i - y_j)^2$
  - **smoothness**: adjacent nodes should have same class label



happy, low energy

unhappy, high energy

Energy Minimization as MAP Inference

- Goal: find the labeling $\boldsymbol{y} \in \{0,1\}^N$ that minimizes the energy

$$\min_{\boldsymbol{y}} E(\boldsymbol{y}) = \min_{\mathbf{y}} \frac{1}{2} \sum_{ij} w_{ij} \left(y_i - y_j\right)^2$$

- This is equivalent to the optimization problem

$$\arg \min_{\boldsymbol{y} \in \{0,1\}^N} E(y) = \arg \max_{\boldsymbol{y} \in \{0,1\}^N} \exp\left(-E(\boldsymbol{y})\right)$$

- Consider the following approximation to the posterior

$$p(\boldsymbol{y_U} | \boldsymbol{W}, \boldsymbol{y_L}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{y})\right)$$

where $Z = \sum_{\boldsymbol{y}} \exp\left(-E(\boldsymbol{y})\right)$ is the normalizing constant.

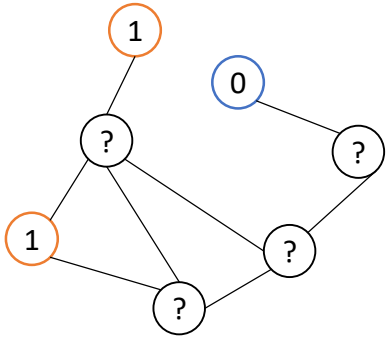
- Energy minimization is equivalent to **MAP inference** in this model!
- Such approaches are called "Energy-Based Learning".
- In general: any nonnegative energy function $E(\boldsymbol{y})$ corresponds to a probability distribution over $\boldsymbol{y}$ (if $\mathrm{dom}(\boldsymbol{y})$ is finite, restrictions apply in other cases).

Label Propagation

- Two aspects: **Smoothness** + **Matching the seed labels**

$$\min_{\boldsymbol{y}\in\{0,1\}^{|V|}} \frac{1}{2}\sum_{ij} w_{ij}\left(y_i - y_j\right)^2 \text{ subject to } y_i = \hat{y}_i \text{ for all } i \in S$$

- Constrained integer optimization problem



- We know how to rewrite the above problem!

$$\min_{\boldsymbol{y}\in\{0,1\}^{|V|}} \boldsymbol{y}^T \boldsymbol{L}\boldsymbol{y} \text{ subject to } y_i = \hat{y}_i \text{ for all } i \in S$$

  - $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ is the graph Laplacian

- And we know how to make it more tractable
  - Drop the integer constraint: y_i $(i \in U)$ can be any real value

$y \in \mathbb{R}^{|V|}$

Label Propagation: Solution

$2 \cdot y_U^T L_{US} \hat{y}_S$

- Task: $\min_{\boldsymbol{y} \in \mathbb{R}^{|V|}} \boldsymbol{y}^T \boldsymbol{L} \boldsymbol{y}$ subject to $y_i = \hat{y}_i$ for all $i \in S$

$\underset{y_U}{\min} \begin{bmatrix} \hat{y}_S \\ y_U \end{bmatrix}^T \cdot \boxed{\begin{matrix} L_{SS} & L_{SU} \\ L_{US} & L_{UU} \end{matrix}} \cdot \begin{bmatrix} \hat{y}_S \\ y_U \end{bmatrix} = \hat{y}_S^T L_{SS} \hat{y}_S$

$\boxed{+ \hat{y}_S^T L_{SU} y_U + y_U^T L_{US} \hat{y}_S}$

$+ \underline{y_U^T L_{UU} y_U} = g(y_U)$

- Solution:
  - w.l.o.g. assume the Laplacian matrix is partitioned into blocks for labeled and unlabeled nodes

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L_{SS}} & \boldsymbol{L_{SU}} \\ \boldsymbol{L_{US}} & \boldsymbol{L_{UU}} \end{bmatrix}$$

  - Accordingly let $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y_S} \\ \boldsymbol{y_U} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{y}}_S \\ \boldsymbol{y_U} \end{bmatrix}$ the vector of labels to be learned
  - Then: $\boldsymbol{y}_U = -\boldsymbol{L}_{UU}^{-1} \cdot \boldsymbol{L}_{US} \cdot \hat{\boldsymbol{y}}_S$

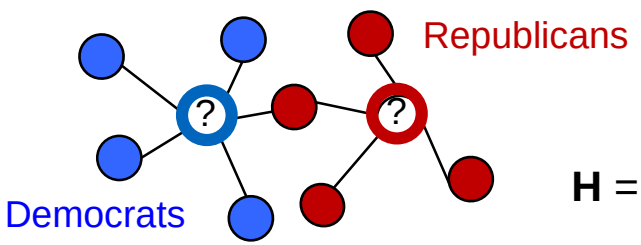
$0 \overset{!}{=} \nabla_{y_U} g(y_U) = \underline{2 \cdot L_{US} \hat{y}_S} + \underline{2 \cdot L_{UU} y_U}$

$\Leftrightarrow -L_{US} \hat{y}_S = L_{UU} y_U$

[Zhu2002]

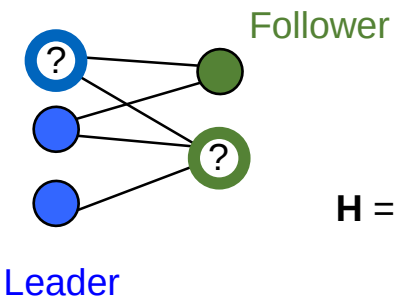
Label Propagation: Generalization

- What if we have K labels?
  - Use one-hot notation $y_{ik} = \begin{cases} 1 & \text{if node } i \text{ is of class } k \\ 0 & \text{else} \end{cases}$
  - Energy function $E(\boldsymbol{Y}) = \sum_{i,j} w_{ij} (\boldsymbol{y}_i - \boldsymbol{y}_j)^T (\boldsymbol{y}_i - \boldsymbol{y}_j)$
- Other types of **network effects** – encode with a compatibility matrix $\boldsymbol{H}$



$$\mathbf{H} = \begin{array}{c|cc} & D & R \\ \hline D & 0.8 & 0.2 \\ R & 0.2 & 0.8 \end{array}$$

homophily
"birds of a feather flock together"



$$\mathbf{H} = \begin{array}{c|cc} & L & F \\ \hline L & 0.2 & 0.8 \\ F & 0.8 & 0.2 \end{array}$$

heterophily
"opposites attract"

- Energy function $E(\boldsymbol{Y}) = \sum_{i,j} w_{ij} (\boldsymbol{y}_i - \boldsymbol{y}_j)^T \boldsymbol{H} (\boldsymbol{y}_i - \boldsymbol{y}_j)$

Label Propagation vs. SBM

- At first glance both models seem very similar
  - labels $\boldsymbol{y}_i$ look a lot like community affiliations $\boldsymbol{z}_i$
  - compatibility matrix $\boldsymbol{H}$ from LP looks like $\boldsymbol{\eta}$ from SBM

- Is LP equivalent to inference in SBM with some $\boldsymbol{z}_i$s observed?
  - Label propagation is a discriminative model that only models the conditional distribution of labels given the similarity graph $p(\boldsymbol{Y}|\boldsymbol{W})$
  - on the other hand, SBM is a generative model that models $\Pr(\boldsymbol{A}|\boldsymbol{Z})$ and $\Pr(\boldsymbol{Z})$
    - we can use SBM to generate new graphs – not the case for LP!
  - for SBM we get the posterior $\Pr(\boldsymbol{Z}|\boldsymbol{A}) = \frac{\Pr(\boldsymbol{A}|\boldsymbol{Z})\Pr(\boldsymbol{Z})}{\Pr(\boldsymbol{A})}$ using Bayes' formula

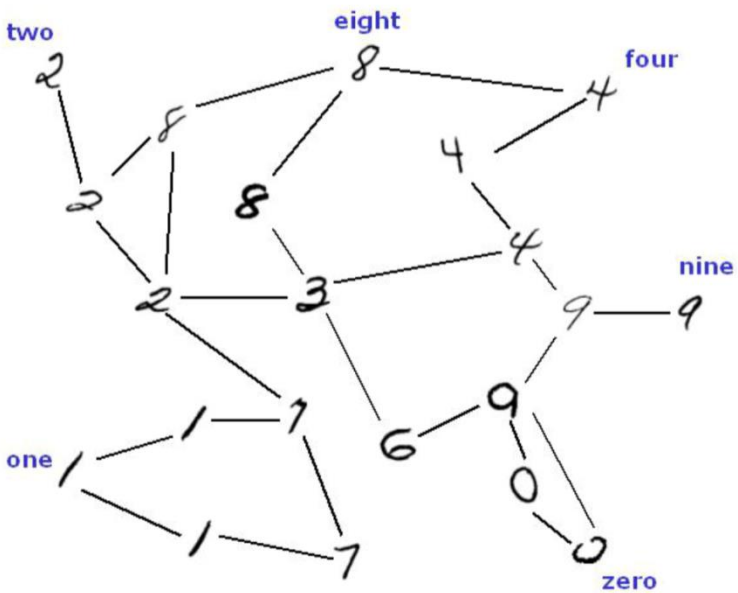
- SBM and LP solve different problems
  - SBM: estimate what parameters generated a given graph $\boldsymbol{A}$ (unsupervised)
  - LP: predict labels of the nodes in U given observed labels and $\boldsymbol{W}$ (supervised)

Non-Graph Data

- What if we only have vector data (no graph is available)?
- Simply construct a graph connecting similar data points
  - graph construction: see section on spectral clustering (e.g. k-NN graph)

- Apply label propagation just like before.

- Features are used to construct the graph, but then only the graph is used for classification.



Transductive Learning

- Label Propagation is a special case of so-called **transductive learning**.
  - Given
    - (i) a set of labeled training instances $T = \{(x_i, y_i)_{i=1\ldots N}\} \subset \mathcal{X} \times \mathcal{Y}$
    - (ii) a set of unlabeled test instances $U = \{(x_i)_{i=1\ldots M}\} \subset \mathcal{X}$
    - [+ potentially some other knowledge (graph structure $\boldsymbol{W}$, affinity matrix $\boldsymbol{H}$)]
  - Goal
    - predict labels **only** for the unlabeled instances U (i.e. learn $f\colon\ U\ \rightarrow \mathcal{Y}$)
  - *"When trying to solve some problem, one should not solve a more difficult problem as an intermediate step"* – Vapnik's principle
- "Traditional" supervised learning (e.g. NN, SVM) is **inductive learning**:
  - Given
    - (i) a set of labeled training instances $T = \{(x_i, y_i)_{i=1\ldots N}\} \subset \mathcal{X} \times \mathcal{Y}$
    - [+ potentially some other knowledge]
  - Goal
    - learn a prediction function (mapping) $f\ :\mathcal{X} \rightarrow \mathcal{Y}$ (that can be applied to any $x_{new} \in \mathcal{X}$)

Transduction vs. Semi-Supervised Learning

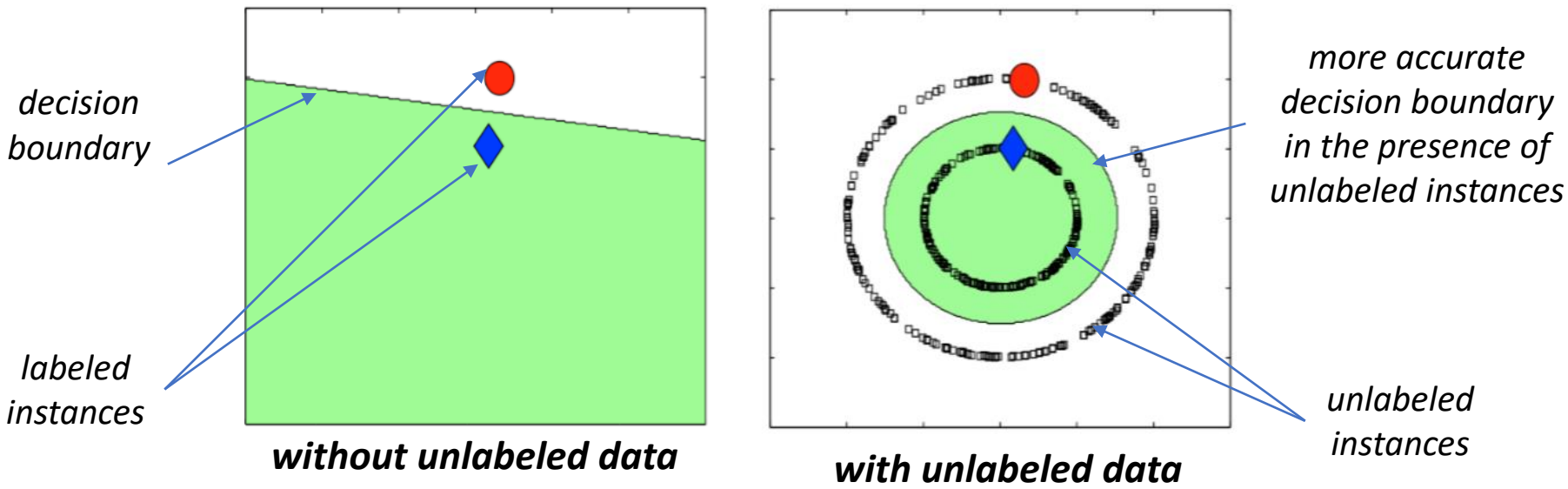
- LP in graphs is often referred to as "graph-based semi-supervised learning"
  - not a complete misnomer, but a more specific term would be: graph-based transductive learning

- **Semi-supervised learning** (SSL) is a more generic principle.
- Standard definition:
  - Given: labeled data $T = \{(x_i, y_i)_{i=1\ldots N}\}$ and unlabeled data $U = \{(x_i)_{i=1\ldots M}\}$.
  - Main idea: Use **both** T **and** U to learn a mapping f.
    This can be either inductive ($f: \mathcal{X} \to \mathcal{Y}$) or transductive ($f: U \to \mathcal{Y}$).

- Transductive learning is almost always semi-supervised:
  - We are given T and U. The goal is to predict labels only for U.
  - Of course we will use U to do this! ⇒ Semi-supervised learning

Why Semi-Supervised Learning Works?

- How can unlabeled data be helpful?
  - Unlabeled data helps us to better model the data distribution

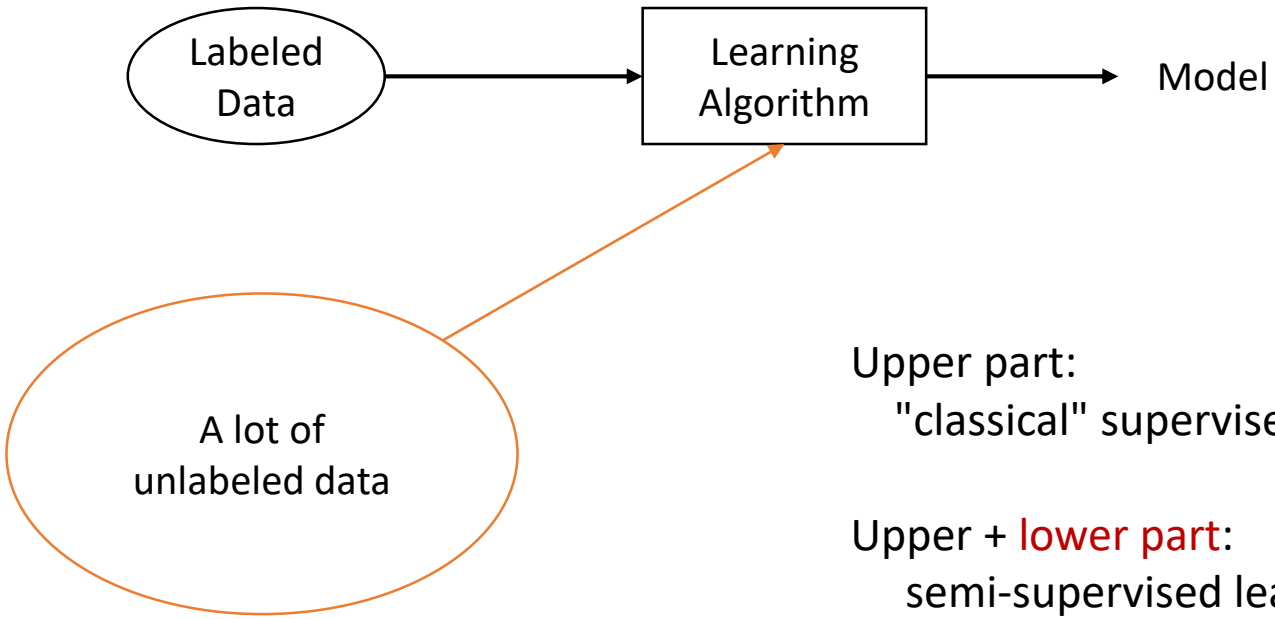


Example from [Belkin et al., JMLR 2006]

- Caveat:
  - We need to make assumptions about the data/label distribution (e.g. manifold / smoothness / cluster / low-density separation assumptions)
  - If the assumptions are wrong, SSL may perform even worse than simple SL!

Semi-supervised Learning: Motivation

- Why semi-supervised learning?
- Large amounts of unlabeled data, small amounts of labeled data
- Labeling/annotating data is expensive



Upper part:
"classical" supervised learning

Upper + lower part:
semi-supervised learning

Roadmap

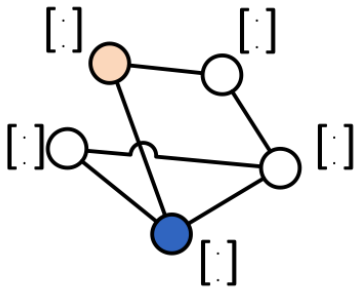
- **Chapter: Graphs**
  1. Graphs & Networks
  2. Generative Models
  3. Clustering
  4. Node Embeddings
  5. Ranking
  6. **Semi-Supervised Learning**
     - Label Propagation
     - **Graph Neural Networks**
  7. Limitations of GNNs

(Semi-supervised) Deep Learning on Graphs

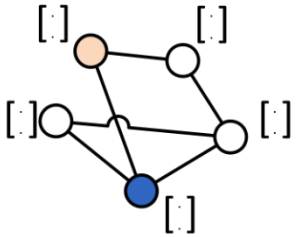
- Neural Networks have achieved outstanding performance for many data types
  - However: usually restricted to simple grid-like (images) or sequential data (text)
- How about neural network for graphs?

- Additional motivation: In a lot of real world graphs the nodes have attributes
  - In citation networks nodes are papers, the text gives rise to attributes, and edges are citations; in protein-protein interaction networks the properties of the proteins can be considered as attributes
  - Label propagation, however, considers only the network structure
  - How can we perform semi-supervised learning on graphs taking both the network structure and the attributes into account?

- Idea: Differentiable message passing
  - a.k.a. neural message passing, graph convolutions, (or more general: graph neural networks)



Differentiable Message Passing Framework (I)



- Given:
  - A graph $G = (V, E)$ with a set of nodes V, E set of edges, E_{vu} edge weight
  - Each node v has features $x_v \in \mathbb{R}^d$
  - A subset of labeled nodes $S \subseteq V$, where y_v denotes the label of node $v \in S$

- Let $h_v^{(k-1)}$ be the hidden representation of node v at previous $k-1$ layer

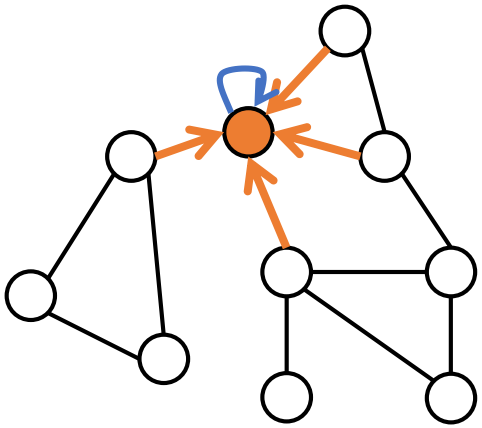
- For each node do:
  1. Gather messages from all neighbors $m_v^{(k)} = \sum_{u \in N(v)} M(h_v^{(k-1)}, h_u^{(k-1)}, E_{vu})$
  2. Update the hidden representation $h_v^{(k)} = U(h_v^{(k-1)}, m_v^{(k)})$

- M and U are any differentiable functions, e.g. neural networks

Differentiable Message Passing Framework (II)

- For each node do:
  1. Gather messages from all neighbors $m_v^{(k)} = \sum_{u \in N(v)} M(h_v^{(k-1)}, h_u^{(k-1)}, E_{vu})$
  2. Update the hidden representation $h_v^{(k)} = U(h_v^{(k-1)}, m_v^{(k)})$

- Example: Calculating the update for the node in orange



1. Gather message from all neighbors

2. Update hidden representation

Example Instantiation of the Framework

- Let $h_v^{(0)} = x_v$
  - At the first layer the representations are the node features

- Set the message aggregation function to be the average over the neighbors

$$m_v^{(k)} = \sum_{u \in N(v)} \frac{1}{d_v} (W^{(k)} h_u^{(k-1)} + b^{(k)})$$

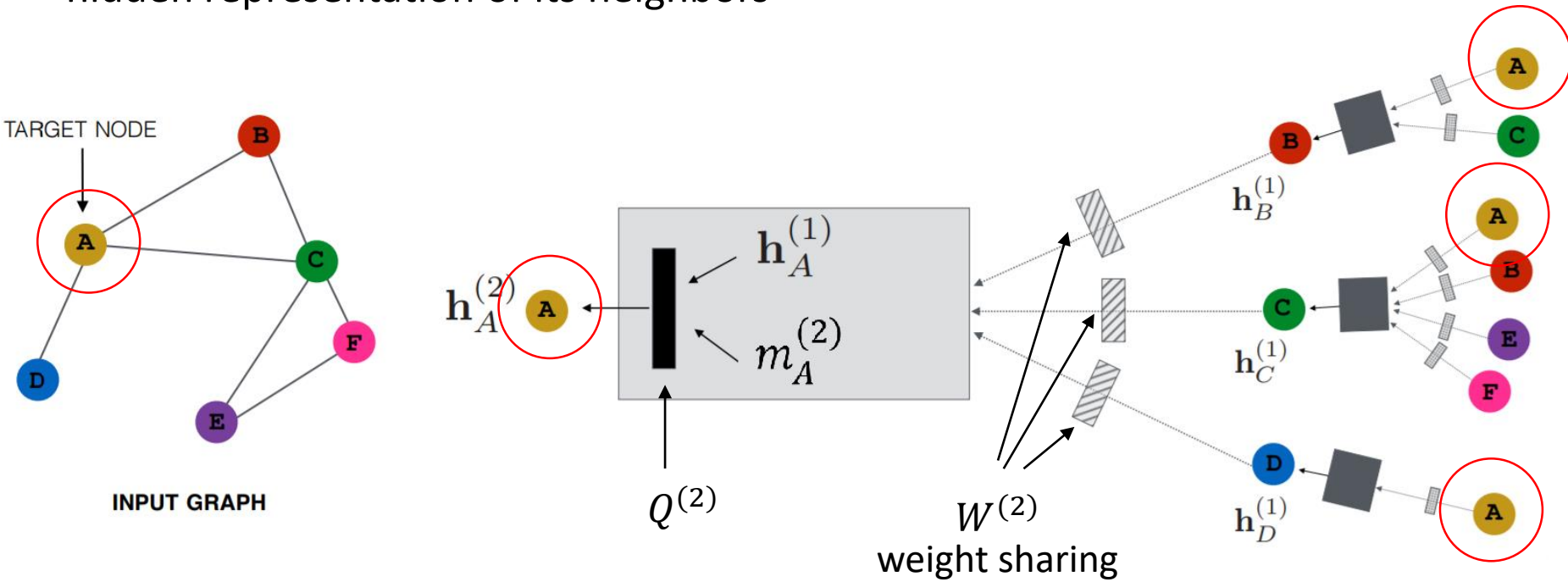
- Calculate hidden representations as simple NNs with a non-linearity

$$h_v^{(k)} = relu(Q^{(k)} h_v^{(k-1)} + p^{(k)} + m_v^{(k)})$$

- Here $W^{(k)}, b^{(k)}, Q^{(k)}, p^{(k)}$ are the **trainable parameters** of the k-th layer

Recursive View of Differentiable Message Passing

- The hidden representation of each node is recursively defined in terms of the hidden representation of its neighbors



How do Neighbors Influence a Given Node?

- Observation 1: K hidden layers equal to K steps of message passing
  - At step 1, node v aggregates information from its 1-hop neighbors
  - At step 2, node v (implicitly) aggregates information from the neighbors of its 1-hop neighbors, i.e. from its 2-hop neighbors
  - ….
  - At step K, node v (implicitly) aggregates information from its K-hop neighbors

- Observation 2: K hidden layers mean that the representation $h_v^{(K)}$ of node v is based on information from all nodes in its K-hop neighborhood

How to Perform Semi-Supervised Node Classification?

- Consider the representations $h_v^{(K)}$ at the final layer (K) as logits

- And use the softmax function to obtain the probability of node v to belong to class c

$$p_v = softmax\left(h_v^{(K)}\right)$$

- Train the network using the standard cross entropy loss between the predicted probabilities and the observed labels
  - Recall: y_{vc} is the one-hot vector encoding the label for node v
  - Denote with p_{vc} the probability that node v to belong to class c
  - Denote with $\mathcal{C}$ the set of all classes, and S the set of labeled nodes

$$\min_{\{W^{(k)},Q^{(k)},p^{(k)},b^{(k)}\}_{k=1..K}} -\sum_{v\in S}\sum_{c\in\mathcal{C}} y_{vc}\log p_{vc}$$

LABELED NODES

Graph Neural Networks

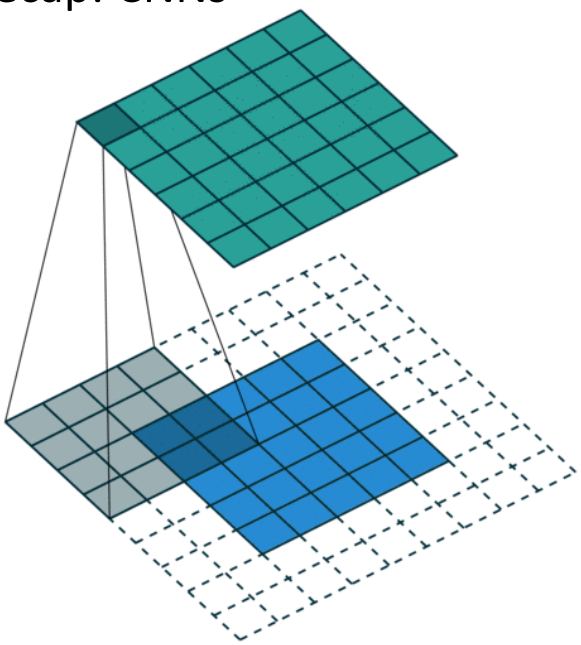
- Exploding interest in recent years:
  - Graph Neural Network (Gori et al., 2005)
  - Spectral Networks and Locally Connected Networks … (Bruna et al., 2014)
  - Gated Graph Neural Network (Li et al., 2016)
  - Convolutional Neural Networks on Graphs with Fast … (Defferrard et al., 2017)
  - Neural Message Passing for Quantum Chemistry (Gilmer et al., 2017)
  - Semi-Supervised Classification with Graph Convolutional Nets (Kipf et al., 2017)
  - Graph Attention Networks (Veličković et al., 2018)
  - Predict then Propagate: Graph Neural Networks meet Personalized PageRank (Klicpera et al., 2019)
- Many tasks beyond semi-supervised node classification
  - graph classification, recommendation, 3d-shape-matching, etc.
- Sometimes called geometric deep learning

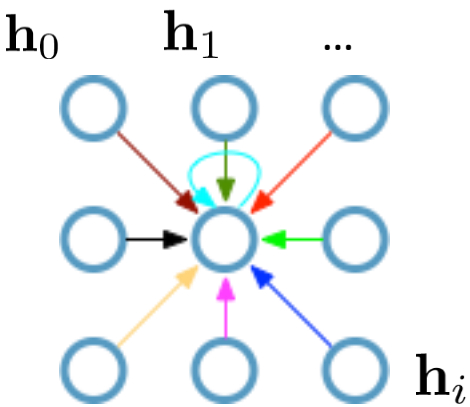
Graph Neural Networks

- Increasing interest in recent years:
  - Graph Neural Network (Gori et al., 2005)
  - Spectral Networks and Locally Connected Networks … (Bruna et al., 2014)
  - Gated Graph Neural Network (Li et al., 2016)
  - Convolutional Neural Networks on Graphs with Fast … (Defferrard et al., 2017)
  - Neural Message Passing for Quantum Chemistry (Gilmer et al., 2017)
  - Semi-Supervised Classification with Graph Convolutional Nets (Kipf et al., 2017)
  - Graph Attention Networks (Veličković et al., 2018)
  - Predict then Propagate: Graph Neural Networks meet Personalized PageRank (Klicpera et al., 2019)
- Many tasks beyond semi-supervised node classification
  - graph classification, recommendation, 3d-shape-matching, etc.
- Sometimes called geometric deep learning

From Matrix Convolution ...

- Alternative view of message passing framework is that we are essentially performing a "graph convolution"
- Recap: CNNs





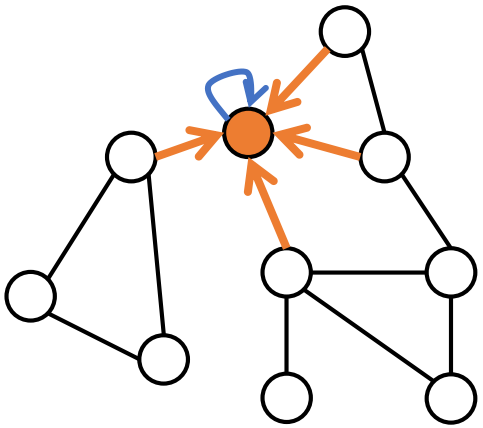
Update for a single pixel:

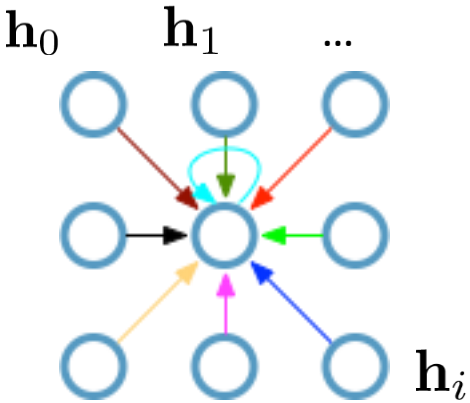
- Transform messages individually $\mathbf{W}_i\mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i\mathbf{h}_i$

- Images are a special kind of graph: every pixel is a node connected to 8 other nodes (up, down, left, right, etc. pixels)

… to Graph Convolution

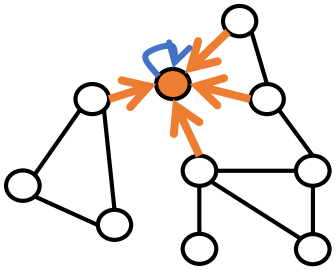
- Alternative view of message passing framework is that we are essentially performing a "graph convolution"
  - Note: Unlike sequences/images, where the convolution operation is clearly defined, there are various versions for graphs. Indeed, one often distinguishes between spatial and spectral approaches.





- Images are a special kind of graph: every pixel is a node connected to 8 other nodes (up, down, left, right, etc. pixels)

Graph Convolutional Neural Networks



- In the spatial domain, a graph convolution updates each nodes' features by considering the local neighborhood.
  - In effect it is some kind of message passing
  - Example: $X_i^{(t+1)} = X_i^{(t)} + \sum_{(i,j)\in E} X_j^{(t)}$
    or in matrix notation $X^{(t+1)} = X^{(t)} + AX^{(t)} = (A + I_n)X^{(t)}$

$\in \mathbb{R}^H$ $\in \mathbb{R}^H$ $\in \mathbb{R}^{N\times N}$ $\in \mathbb{R}^{N\times H}$ $\tilde{A}$ $\in \mathbb{R}^{N\times H}$

- Similar to a (learnable) convolutional layer we can formulate a (learnable) graph convolutional layer

$\in \mathbb{R}^{N\times H_{l+1}}$ $\in \mathbb{R}^{N\times H_l}$

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right)$$

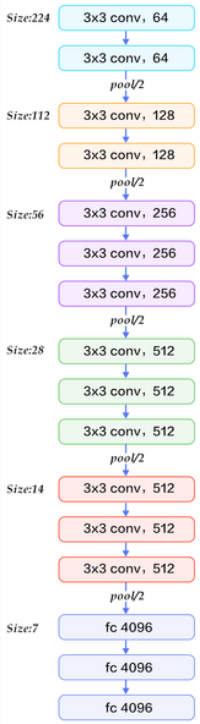
$\in \mathbb{R}^{N\times H_{l+1}}$ $\in$

Non-linearity Message Passing Feature Transformation

- Normalizing the propagation matrix avoids exploding gradients
  - Choose $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ where $\tilde{A} = A + I_n$ and $D_{ii} = \sum_j \tilde{A}_{ij}$ is the degree matrix of $\tilde{A}$

[Kipf2017]

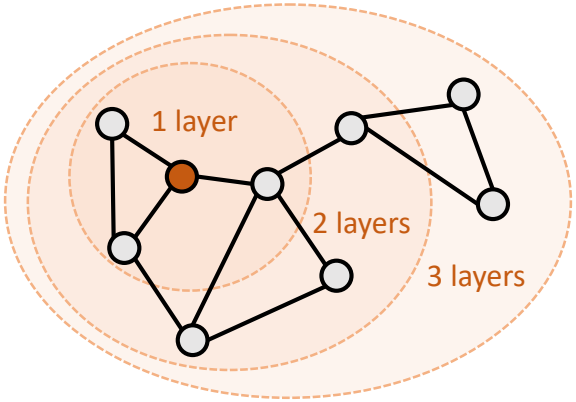
How Deep are Graph Neural Networks?



- In an MLP or CNN, depth means the number of non-linear transformations of the input
- On graphs we can also consider the furthest distance that a model propagates information as depth

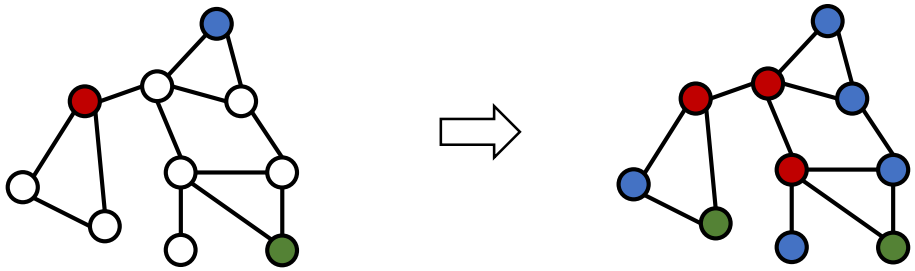
$$H^{(l+1)} = \sigma\left(\tilde{D}^{-½}\tilde{A}\tilde{D}^{-½}\boxed{H^{(l)}W^{(l)}}\right)$$

- GCNs apply 1 transformation per message passing step
- *Transformation depth* and *propagation depth* are orthogonal in GNNs
  - We could transform the messages with a multi-layer network in each step or propagate without transforming for multiple steps

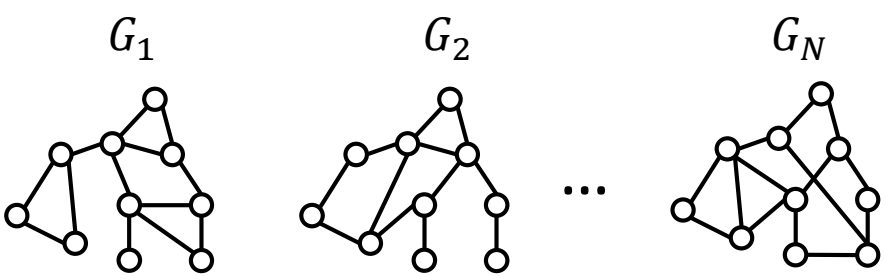


Single vs. Multi-graph Learning

- So far we had a single graph $G = (V, E)$ and we learned targets for the nodes
  - For example predict the classes for each node (red, green or blue) in a single large graph



- What if we have multiple graphs as input and the target is for the graph?
  - For example each input is a molecule (i.e. a graph of atoms) and the graph level target is whether it is a effective drug against some disease



G_1 = Yes
G_2 = No
…
G_N = Yes

Multi-graph Learning Framework

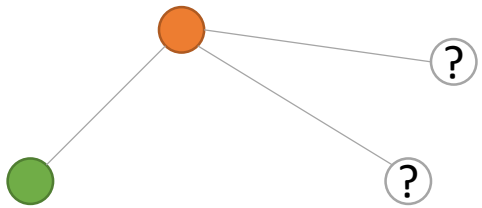
- Given:
  - A set of graphs $\mathcal{G} = \{G_i = (V_i, E_i)\}_{i=1..N}$
  - A subset of labeled graphs $\mathcal{H} \subseteq \mathcal{G}$, with y_{G_i} denotes the graph level target of the graph $G_i \in \mathcal{H}$
- Obtain the hidden representation of the last layer for all nodes in all graphs according to the message passing framework
  - Let $H_{G_i} = \left[h_1^{(K)}, h_2^{(K)}, \dots, h_{|V_i|}^{(K)}\right]$ be a matrix where we stacked the final representations **for all nodes** for graph G_i
- Define an aggregation function $R(H_{G_i})$ over the node representations of a given graph that produces a representation for the **entire** graph $h_{G_i} = R(H_{G_i})$
- For classification: consider h_{G_i} as the logits and train a model using standard cross-entropy loss, i.e. $\mathcal{L}(y_{G_i}, softmax(h_{G_i}))$
  - For regression, e.g. predicting functional properties of molecules you can use squared loss
- Example agg. function $R(H_{G_i}) = \frac{1}{|V_i|}\sum_{v \in V_i} h_v^{(K)}$ is the average of the node embedding

Summary

- Semi-supervised learning / graph-based transductive learning
  - Leverage unlabeled data to improve performance of supervised learning
  - Helps if assumptions about the data distribution are correct, e.g. homophily
- Label Propagation spreads labels along the edges of a graph by minimizing the difference between neighbors
  - Usually assumes smoothness but other kinds of network effects can be modeled as well
- Differentiable message passing: / GNN / GCN
  - Flexible framework to apply the power of deep learning to graphs
  - The nodes aggregate information from their k-hop neighbors
  - Message passing is a generalization of convolution from grids to general graphs
- We distinguish between node-level learning (e.g. node classification) and graph-level learning (e.g. graph classification)

Questions

- Consider the graph below. What is the influence of the green node on the unlabeled nodes in Label Propagation? Why? How about GNNs with $K = 2$?



- Does semi-supervised learning exist outside of learning on graphs?

- What could be alternative aggregation functions beside summation in the Gather step of GNNs?

- Can you apply GNNs to vector data without a specified graph structure?

Reading Material

- [Zhu2002] Zhu, X., & Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Center for Automated Learning and Discovery, CMU: Carnegie Mellon University, USA.
- [Kipf2017] Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. In ICLR 2017 : International Conference on Learning Representations 2017