Machine Learning for Graphs and Sequential Data

Graphs – Classification (Semi-Supervised Learning)

Lecturer: Prof. Dr. Stephan Günnemann

cs.cit.tum.de/daml

Summer Term 23

Data Analytics and
Machine Learning

Roadmap

- **Chapter: Graphs**
  1. Graphs & Networks
  2. Generative Models
  3. Ranking
  4. Clustering
  5. **Classification (Semi-Supervised Learning)**
  6. Node/Graph Embeddings
  7. Graph Neural Networks (GNNs)

Types of Machine Learning Problems on Graphs

- So far we have discussed unsupervised learning problems on graphs
  - generative models
  - clustering / community detection
  - 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

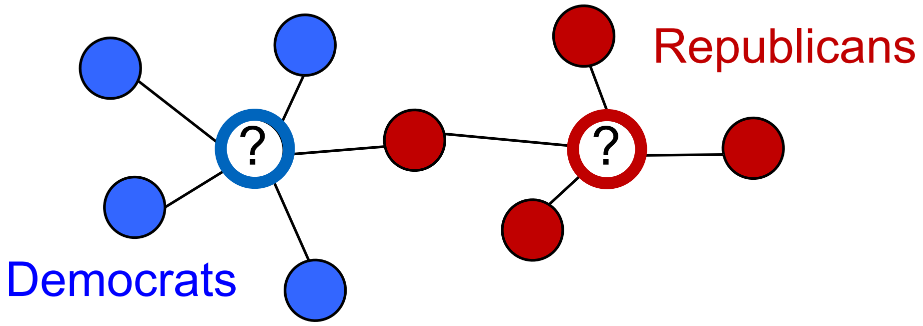
  } assuming that some training data is given

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

Collective Classification

don't need features
only need structure/edges
relationship

- 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**

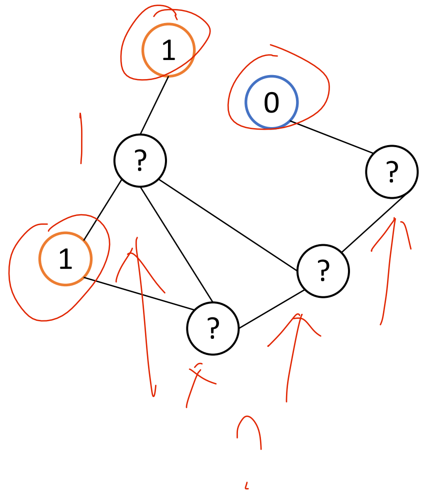


- Standard assumption: **Homophily** (a.k.a. **assortativity**)
  - *"birds of a feather flock together"*
  - *a.k.a. smoothness assumptions*
  - *that is, if nodes are connected by an edge, they are likely to have same labels*

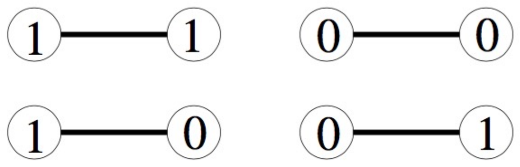
Label Propagation

optimization problem

- 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** $E(\boldsymbol{y}) = \frac{1}{2}\sum_{ij} w_{ij}\left(y_i - y_j\right)^2$ $\Leftrightarrow$ min $E(y)$
  - **smoothness**: adjacent nodes should have same class label



happy, low energy

unhappy, high energy

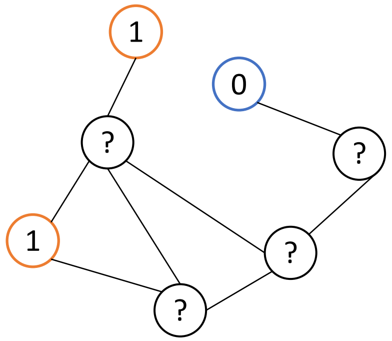
Label Propagation

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

find →

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

- Constrained integer optimization problem



- We know how to rewrite the above problem!

hard

discrete problem →

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

unknown vector

  - $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ 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

Label Propagation: Solution

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

  any real value vector

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

    $$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_{SS} & \boldsymbol{L}_{SU} \\ \boldsymbol{L}_{US} & \boldsymbol{L}_{UU} \end{bmatrix}$$ how label and unlabel connect

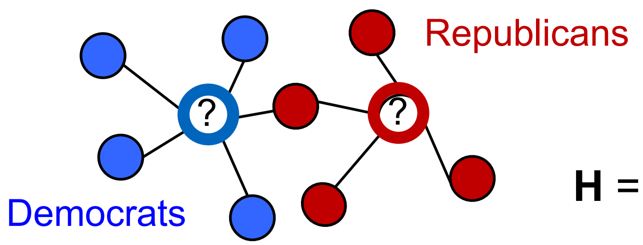
  - Accordingly let $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_S \\ \boldsymbol{y}_U \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{y}}_S \\ \boldsymbol{y}_U \end{bmatrix}$ the vector of labels to be learned
  - Then: $\boxed{\boldsymbol{y}_U = -\boldsymbol{L}_{UU}^{-1} \cdot \boldsymbol{L}_{US}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

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.

Label Propagation: Generalization

k class

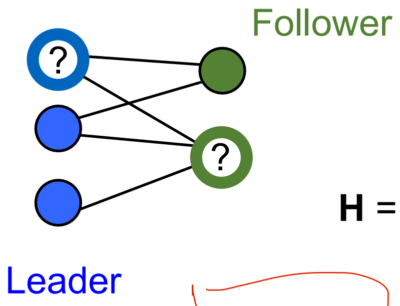
- What if we have K labels?
  - Use one-hot notation $y_{ik} = \begin{cases} 1 & \text{if node } i \text{ is of class } k \\ 0 & \text{else} \end{cases}$
  - Energy function $E(\boldsymbol{Y}) = \sum_{i,j} w_{ij} (\boldsymbol{y}_i - \boldsymbol{y}_j)^T (\boldsymbol{y}_i - \boldsymbol{y}_j)$
- Other types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

different type



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

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 $p(\boldsymbol{Y}|\boldsymbol{W})$ only perdict
  - 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! generate graph
  - 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)

Transductive Learning

- Label Propagation is a special case of so-called **transductive learning**.
  - Given
    - (i) a set of labeled training instances $T = \{(x_i, y_i)_{i=1...N}\} \subset \mathcal{X}\times\mathcal{Y}$
    - (ii) a set of unlabeled test instances $U = \{(x_i)_{i=1...M}\} \subset \mathcal{X}$
    - [+ potentially some other knowledge (graph structure $\boldsymbol{W}$, affinity matrix $\boldsymbol{H}$)]
  - Goal
    - predict labels **only** for the unlabeled instances U (i.e. learn $f\colon U \rightarrow \mathcal{Y}$) no new data
  - *"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 $T = \{(x_i, y_i)_{i=1...N}\} \subset \mathcal{X}\times\mathcal{Y}$
    - [+ potentially some other knowledge]
  - Goal
    - for new data
    - 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

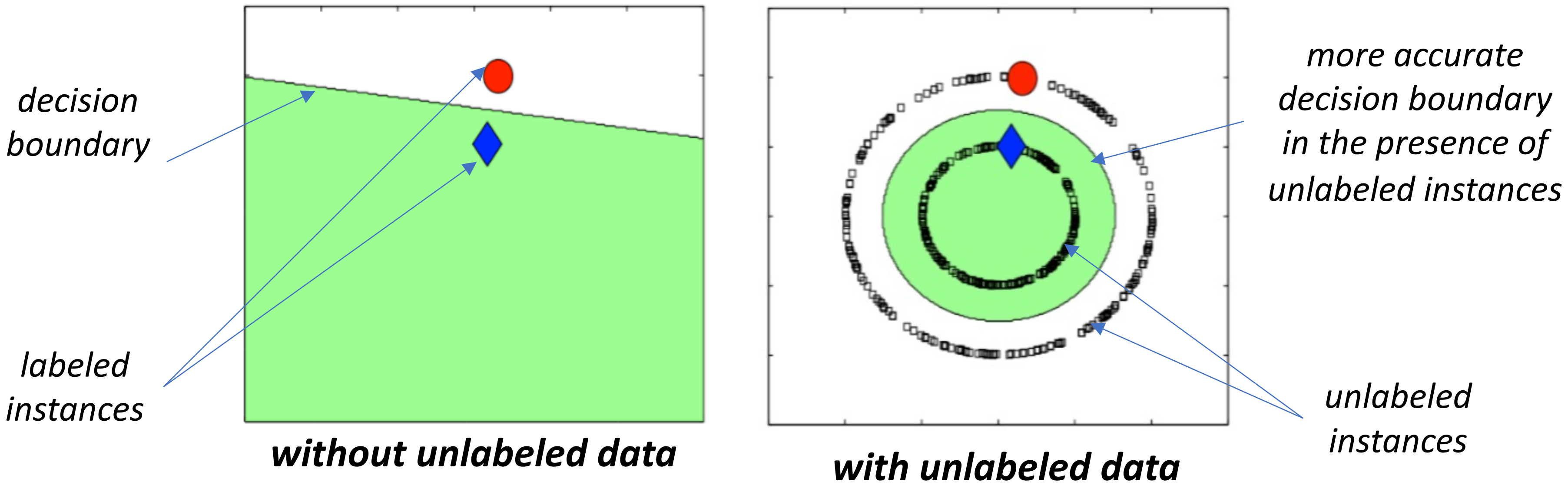
- LP in graphs is often referred to as "graph-based semi-supervised learning"
  - not a complete misnomer, but a more specific term would be: graph-based transductive learning

- **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 Does Semi-Supervised Learning Work?

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

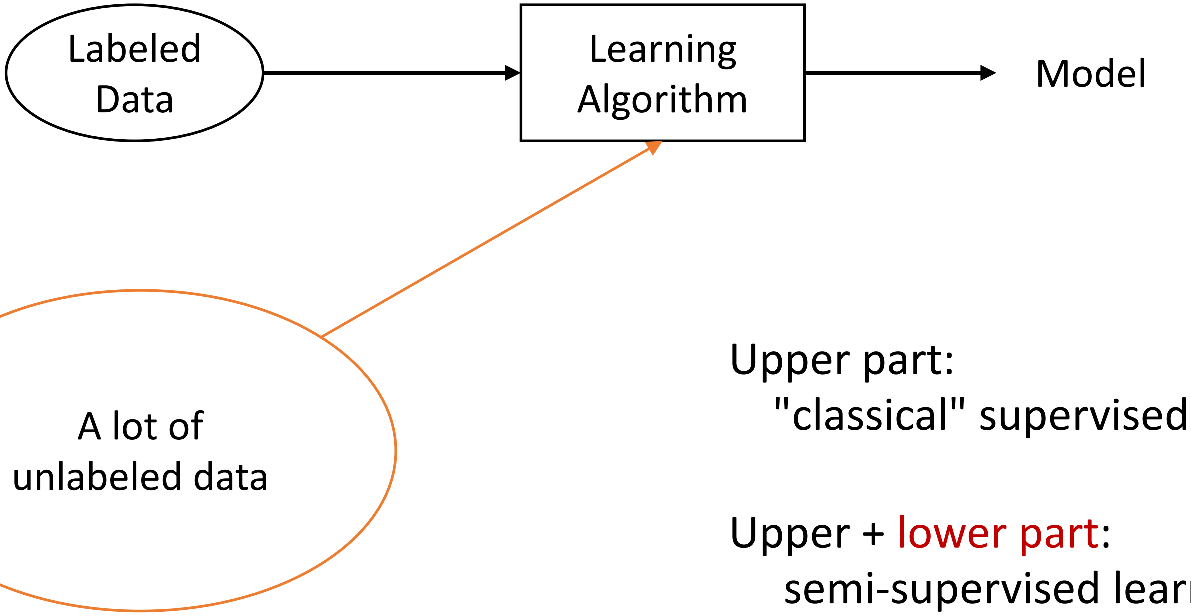


Example from [Belkin et al., JMLR 2006]

- 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

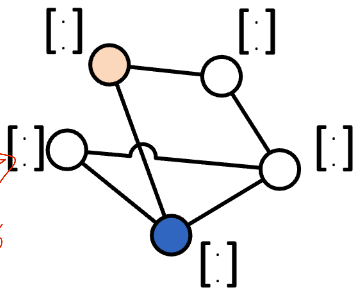
- Why semi-supervised learning?
- Large amounts of unlabeled data, small amounts of labeled data
- Labeling/annotating data is expensive



Upper part:
"classical" supervised learning

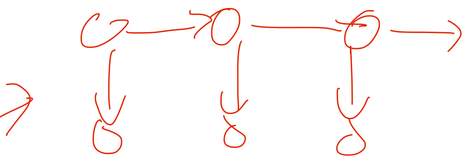
Upper + lower part:
semi-supervised learning

Attributed Graphs



How to handle attributed graphs? observable: features vectors

– e.g. each node is additionally annotated with a feature vector in $\mathbb{R}^d$



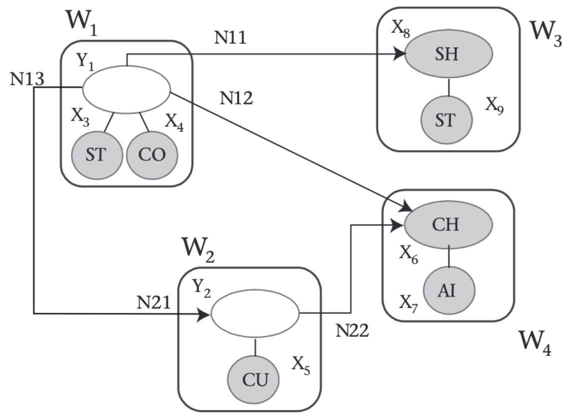
▪ Traditional approach: Markov random fields

– Similar to Hidden Markov Models (HMMs)

– Latent variables are not sequential but graph-structured

– Semi-supervised: Parts of the latent variables are observed

– More details: Sen et al., 2008 Collective classification in network data. *AI magazine*, *29*(3), 93-93.

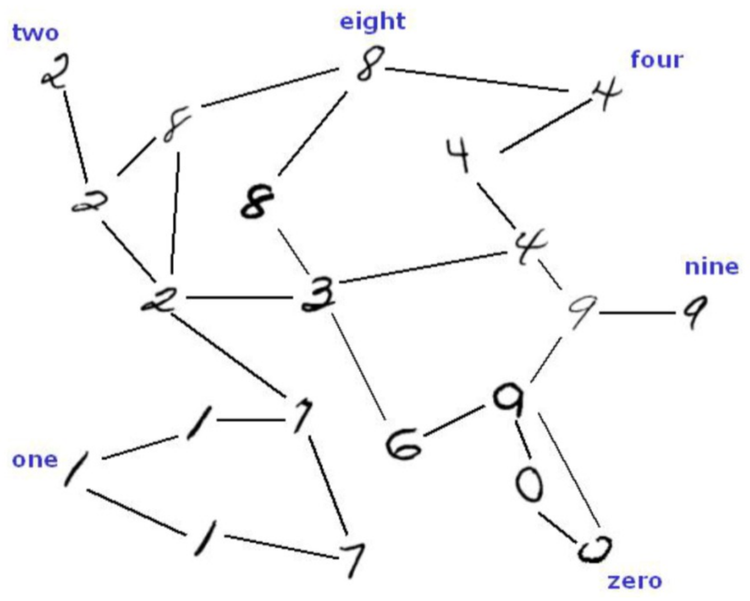


▪ Nowadays: Graph Neural Networks (Chapter 7)

Graph Construction

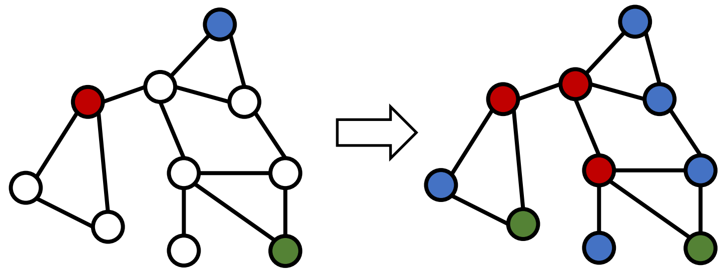
How to handle vector data (when no graph-structure is available)?

- Simply construct a graph connecting similar data points
  - see section on spectral clustering (e.g. k-NN graph)
- Then apply label propagation just like before

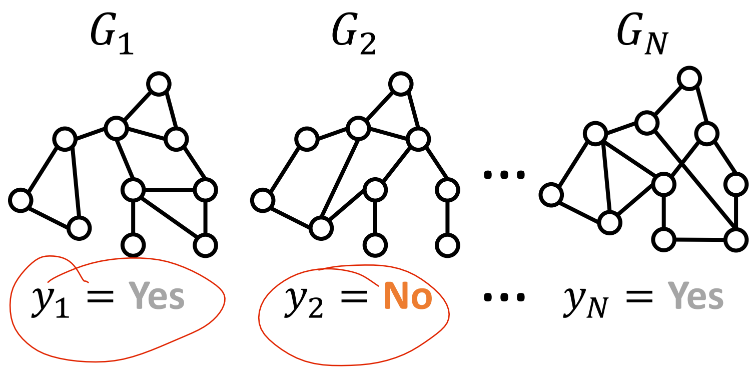


Node Classification vs. Graph Classification

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



- What if we have multiple graphs as input and the target is for the graph?
  - Given:
    - A set of training graphs $\{G_i = (V_i, E_i)\}_{i=1..N}$
    - along with labels y_i denoting the graph level target of graph G_i
  - Goal: Learn a function $f: \mathcal{G} \rightarrow Y$ which maps (new) graphs to labels
    - $\mathcal{G}$ is the set of all graphs of interest
    - Y is the set of class labels



E.g. each input is a molecule (a graph of atoms) and the graph level target is whether it is an effective drug against some disease

Graph Classification

How to handle graph classification?

- This is a standard i.i.d. learning set-up (just the input-data is more complex)
- Graph kernels
  - Machine learning: Kernels are symmetric, positive (semi-)definite functions that measure similarity between instances via inner product

$$k(G_1, G_2) := \phi(G_1)^T \phi(G_2)$$

  - Use graphs as the input to the kernel function
    → Graph kernel function $k: \mathcal{G} \, x \, \mathcal{G} \rightarrow \mathbb{R}$
    - E.g. Random walk kernel, Shortest-path kernel, …
  - Then use, e.g., Support Vector Machines (SVMs) for graph classification
- Or use Graph Neural Networks (Chapter 7)

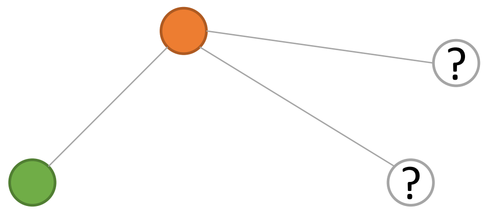
Borgwardt, K., Ghisu, E., Llinares-López, F., O'Bray, L., & Rieck, B. (2020). Graph kernels: State-of-the-art and future challenges. *Foundations and Trends in Machine Learning*, *13*(5-6), 531-712.
Kriege, N. M., Johansson, F. D., & Morris, C. (2020). A survey on graph kernels. *Applied Network Science*, *5*(1), 1-42.

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
- Attributed graphs can be handled with Markov Random Fields
- Graph classification can be handled with Graph Kernels

Questions

- Consider the graph below. What is the influence of the green node on the unlabeled nodes in Label Propagation? Why?



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