Advanced Data Visualization

CS 6965

Spring 2018

Prof. Bei Wang Phillips

University of Utah

Lecture 27

Graph-based ML
ML on graphs

NV

Announcement

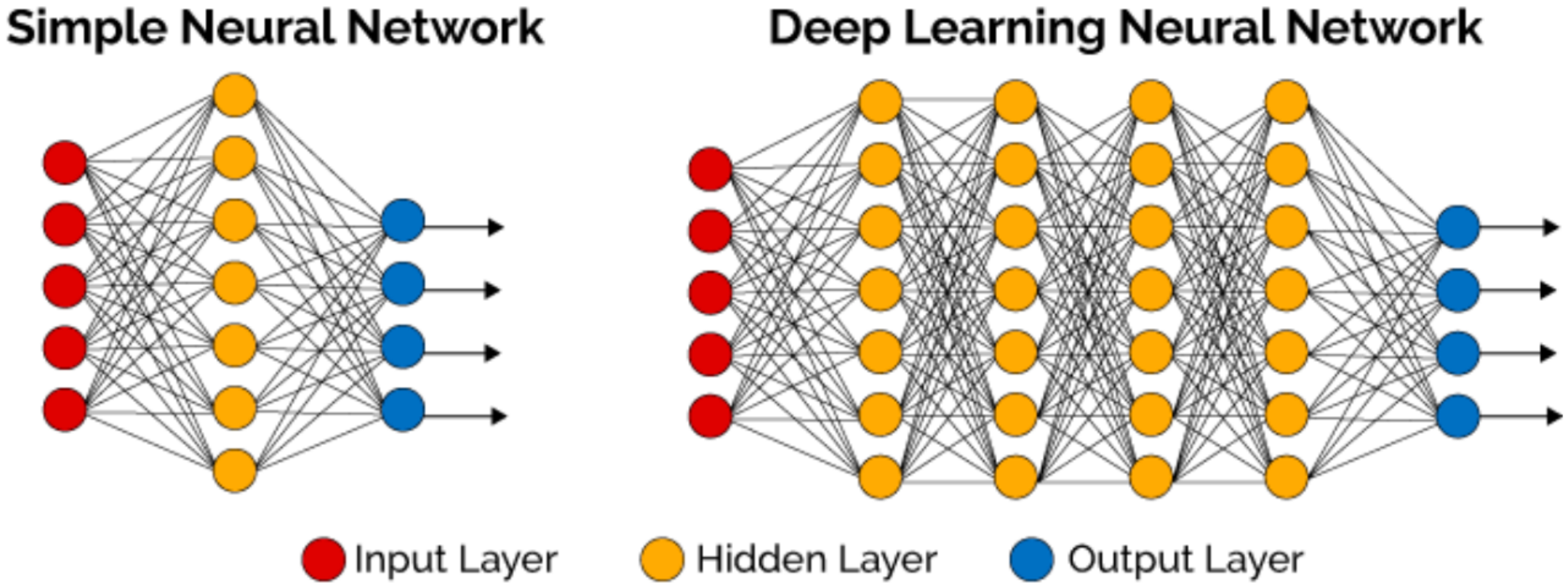
- Final project presentation sign up!
- April 24 (Tuesday) 9:10 - 10:30 a.m, and April 27 (Friday) 8:00 to 10:00 a.m.

Graph-Based ML

https://research.googleblog.com/2016/10/graph-powered-machine-learning-at-google.html

What ML algorithms are based on graphs?

Deep Learning!



First train a system using labeled data with features and then apply the trained system to unlabeled data

http://www.global-engage.com/life-science/deep-learning-in-digital-pathology/

Graph-powered Machine Learning at Google

Thursday, October 06, 2016

Posted by Sujith Ravi, Staff Research Scientist, Google Research

Recently, there have been significant advances in Machine Learning that enable computer systems to solve complex real-world problems. One of those advances is Google's large scale, graph-based machine learning platform, built by the Expander team in Google Research. A technology that is behind many of the Google products and features you may use everyday, graph-based machine learning is a powerful tool that can be used to power useful features such as reminders in Inbox and smart messaging in Allo, or used in conjunction with deep neural networks to power the latest image recognition system in Google Photos.

https://research.googleblog.com/2016/10/graph-powered-machine-learning-at-google.html

Graph-based semi-supervised learning

- Sparse training data
- Model labeled and unlabeled data jointly during learning, leveraging the underlying structure in the data
- Easily combine multiple types of signals (for example, relational information from Knowledge Graph along with raw features) into a single graph representation and learn over them.

Graph-Based Semi-Supervised Learning

Semi-Supervised Learning Tutorial

Xiaojin Zhu

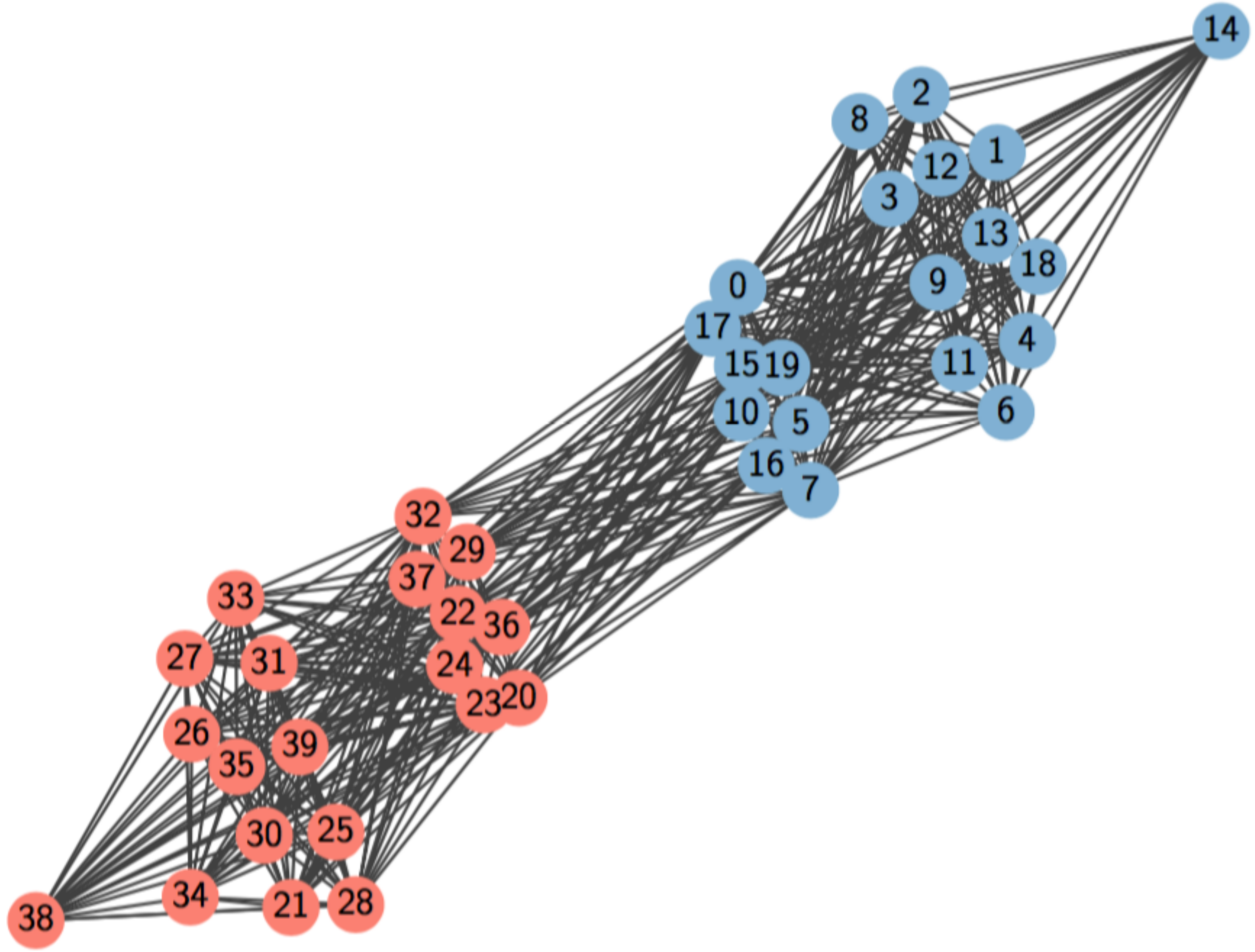
Department of Computer Sciences
University of Wisconsin, Madison, USA

ICML 2007

http://pages.cs.wisc.edu/~jerryzhu/icml07tutorial.html

Spectral Clustering

Spectral Clustering



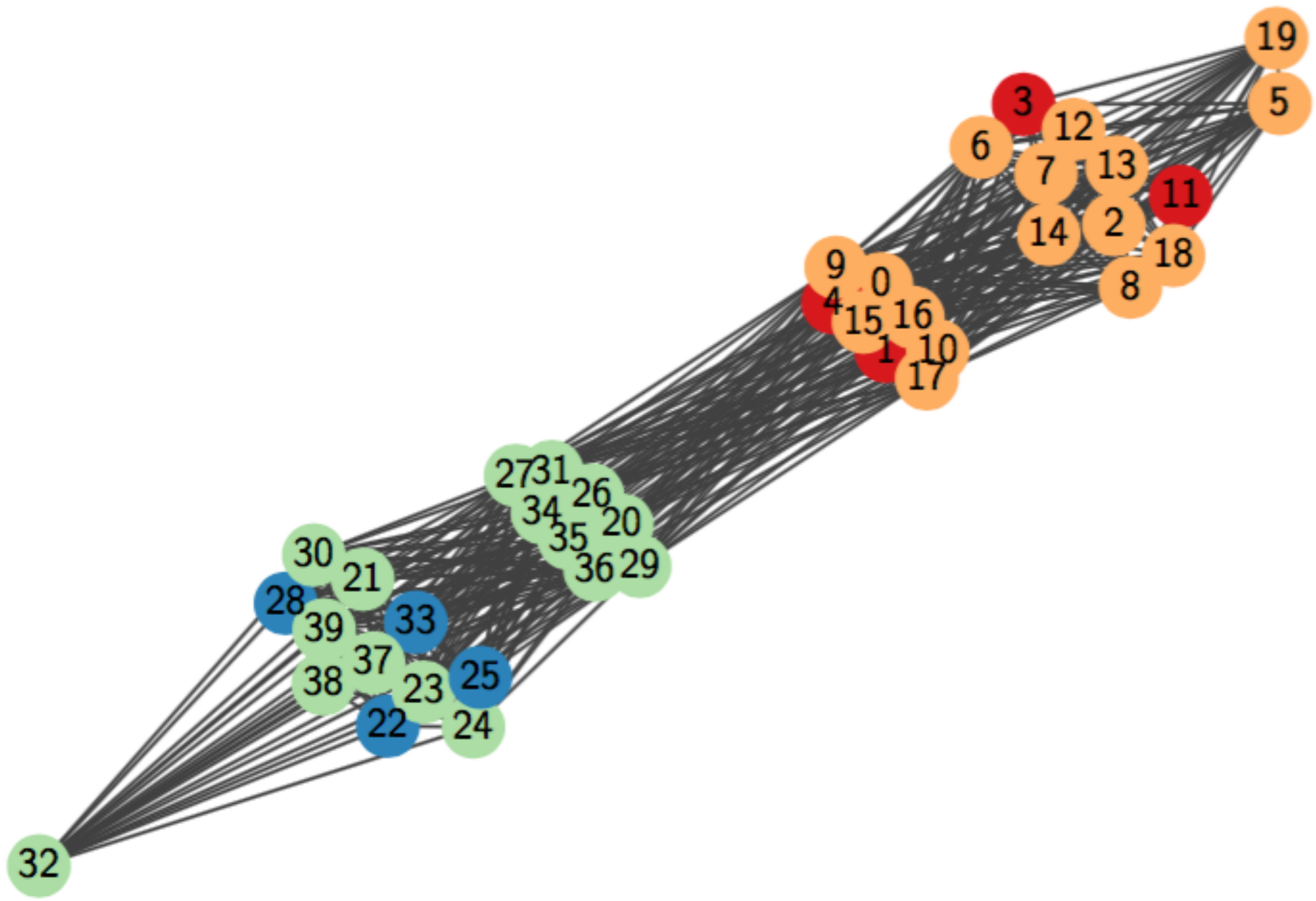
https://arxiv.org/abs/1708.08436

Spectral clustering algorithm for graphs. We use the Ng-Jordan-Weiss algorithm [39] to perform spectral clustering of graphs. Let n_0 be the number of vertices in a graph. Recall the *affinity matrix* $A \in \mathbb{R}^{n_0 \times n_0}$ is a matrix where A_{ij} (≥ 0) captures the affinity (i.e. measure of similarity) between vertex i and vertex j. In our setting, A_{ij} corresponds to the weight of edge e_{ij} in the diagonal edge weight matrix W_1. The spectral clustering algorithm in [39] can be summarized as follows:

1. Compute the diagonal matrix $\Delta \in \mathbb{R}^{n_0 \times n_0}$ with diagonal elements Δ_{ii} being the sum of A's i-th row, that is, $\Delta_{ii} = \sum_j A_{ij}$.
2. Construct the matrix $M = \Delta^{-1/2} A \Delta^{-1/2}$.
3. Find $u_1, u_2, \cdots, u_k$, the eigenvectors of M corresponding to the k largest eigenvalues (chosen to be orthogonal to each other in the case of repeated eigenvalues), and form the matrix $X = [u_1 u_2 \cdots u_k] \in \mathbb{R}^{n_0 \times k}$ by stacking the eigenvectors in columns.
4. Form the matrix Y from X by re-normalizing each of X's rows to have unit length, that is, $Y_{ij} = X_{ij} / \left(\sum_j X_{ij}^2\right)^{1/2}$.
5. Treating each row of Y as a point in $\mathbb{R}^k$, cluster them into k clusters via the k-means algorithm.
6. Finally, assign the original vertex v_i to cluster j if and only if row i of the matrix Y is assigned to cluster j.

Label Propagation

Label Propagation



https://arxiv.org/abs/1708.08436

Label propagation on graphs. We implement a simple version of the iterative label propagation algorithm [60] based on the notion of stochastic matrix (i.e. random walk matrix) $P = \Delta^{-1}A$, where A is the affinity matrix and Δ is the diagonal matrix with diagonal elements $\Delta_{ii} = \sum_j a_{ij}$ (as defined in Section 5.1).

The matrix P represents the probability of label transition. Given P and an initial label vector $\mathbf{y}$, we iteratively multiply the label vector $\mathbf{y}$ by P. If the graph is *label-connected* (i.e. we can always reach a labeled vertex from any unlabeled one), then P^t converges to a stationary distribution, that is, $P^t\mathbf{x} = \mathbf{x}$ for a large enough t.

Suppose there are two label classes $\{+1, -1\}$. Without loss of generality, assume that first l of the n vertices are assigned labels initially, represented as a length-l vector $\mathbf{y}_l$. Given a graph $G(V, E)$ and labels $\mathbf{y}_l$, the algorithm is given as:

1. Compute A, Δ, and $P = \Delta^{-1}A$.
2. Initialize $\mathbf{y}^{(0)} = (\mathbf{y}_l, \mathbf{0})$, $t = 0$.
3. Repeat until convergence:

$$\mathbf{y}^{(t+1)} = P\mathbf{y}^{(t)},$$

$$\mathbf{y}_l^{(t+1)} = \mathbf{y}_l^{(t)}.$$

4. Return $\mathrm{sgn}(\mathbf{y}^{(\mathbf{t})})$.

https://arxiv.org/abs/1708.08436

Consider P to be divided into blocks as follows:

$$P = \begin{pmatrix} P_{ll} & P_{lu} \\ P_{ul} & P_{uu}, \end{pmatrix}$$

where l and u index the labeled and unlabeled vertices with the number of vertices $n_0 = l + u$. Let $\mathbf{y} = (\mathbf{y}_l, \mathbf{y}_u)$ be the labels at convergence, then $\mathbf{y}_u$ is given by :

$$\mathbf{y}_u = (I - P_{uu})^{-1} P_{ul} \mathbf{y}_\mathbf{l}$$

As long as our graph is connected, it is also label-connected and $(I - P_{uu})$ is non-singular. So we can directly compute the labels at convergence without going through the iterative process described above.

https://arxiv.org/abs/1708.08436

Additional Reading

Graph-based Semi-supervised Learning

Zoubin Ghahramani

Department of Engineering
University of Cambridge, UK

zoubin@eng.cam.ac.uk
http://learning.eng.cam.ac.uk/zoubin/

MLSS 2012
La Palma

http://mlg.eng.cam.ac.uk/zoubin/talks/lect3ssl.pdf

Thanks!

Any questions?

You can find me at: **beiwang@sci.utah.edu**

CREDITS

Special thanks to all people who made and share these awesome resources for free:

- Presentation template designed by Slidesmash
- Photographs by unsplash.com and pexels.com
- Vector Icons by Matthew Skiles

Presentation Design

This presentation uses the following typographies and colors:

Free Fonts used:

http://www.1001fonts.com/oswald-font.html

https://www.fontsquirrel.com/fonts/open-sans

Colors used