Practical Machine Learning on Graphs

Pantelis Elinas

November 2019

www.data61.csiro.au

Course Agenda

- Basic definitions
- Measures of centrality
- Traditional machine learning approach (feature engineering)
- Node classification: Laplacian eigenmaps
- Node classification: DeepWalk and Node2Vec
- Link prediction: DeepWalk and Node2Vec
- Node classification: Graph Convolutional Networks
- Node classification and link prediction/unsup : GraphSAGE

Before we start…

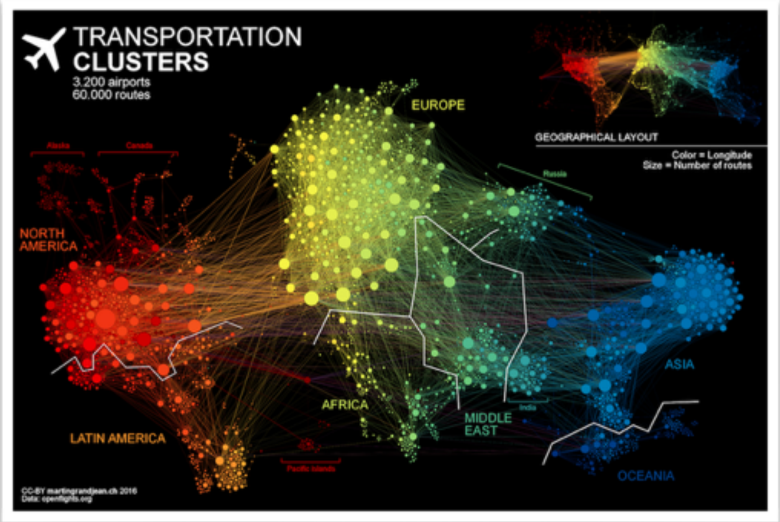
Navigate to `stellar-practical-ml-on-graphs` directory

```
git pull
```

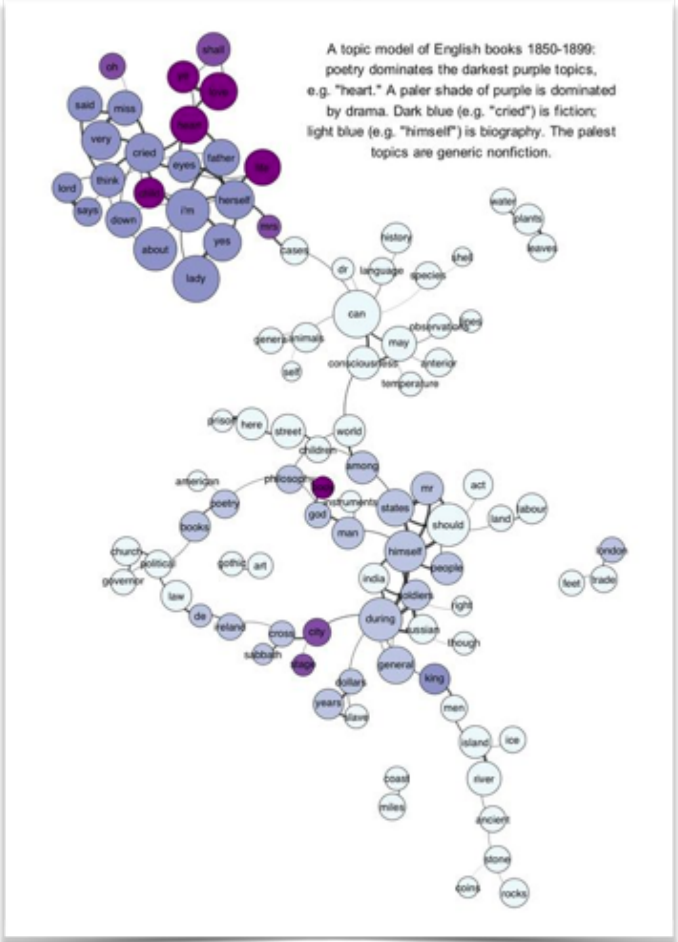
Networks

Social networks

Terrorist networks



Topic models

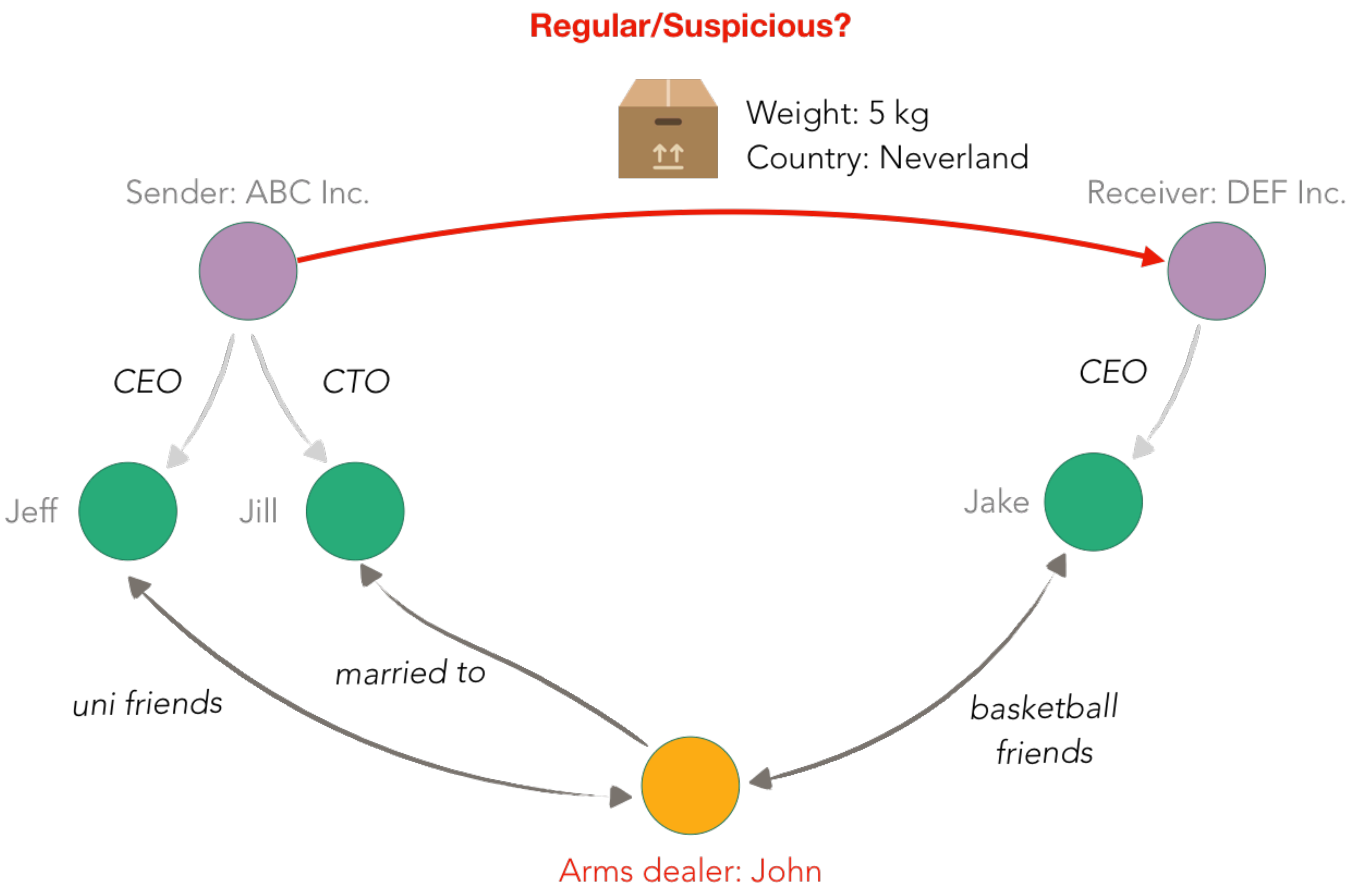


Example

Shipping + Customs: Which package should we inspect?

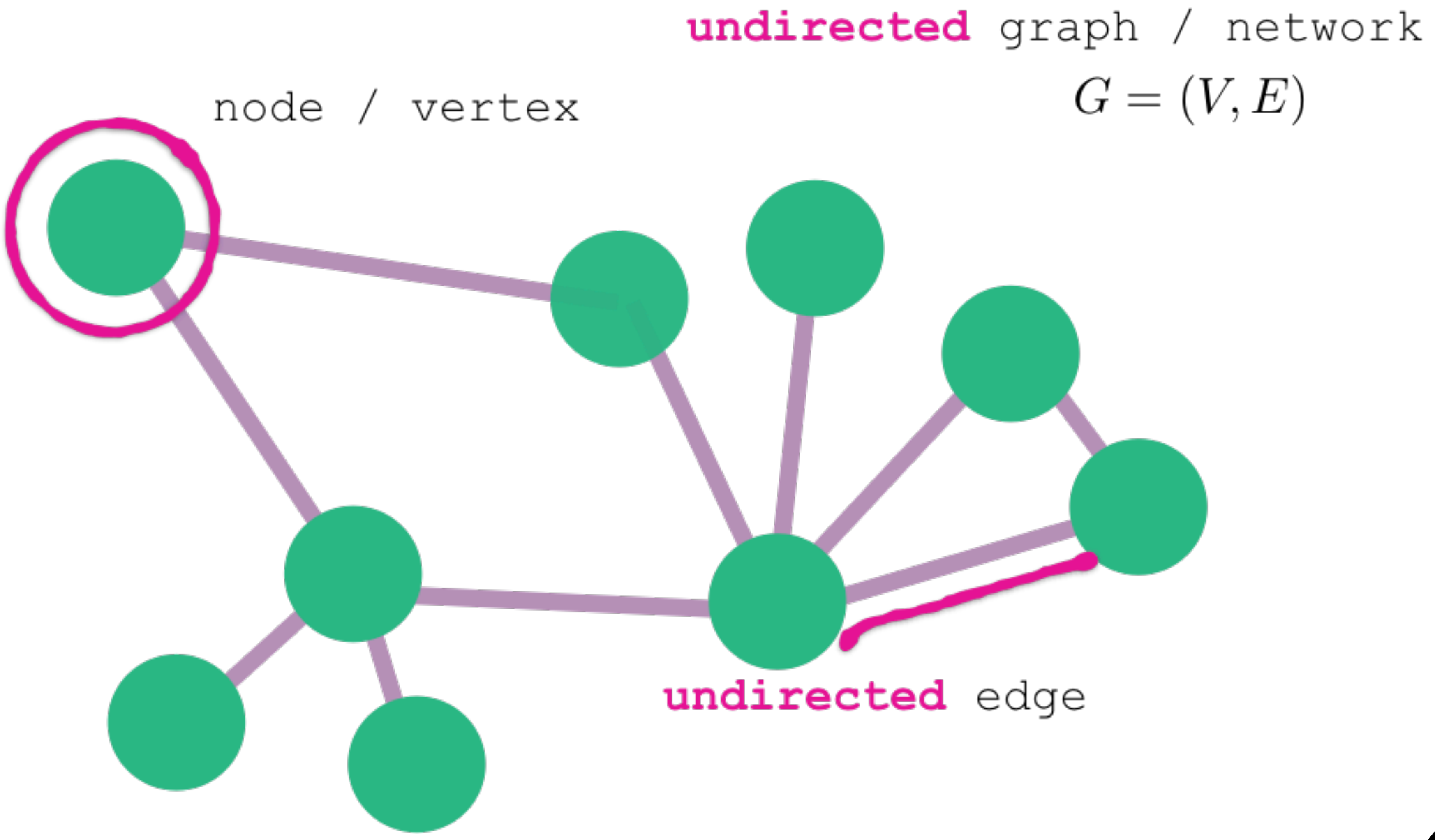
regular or suspicious?

Example

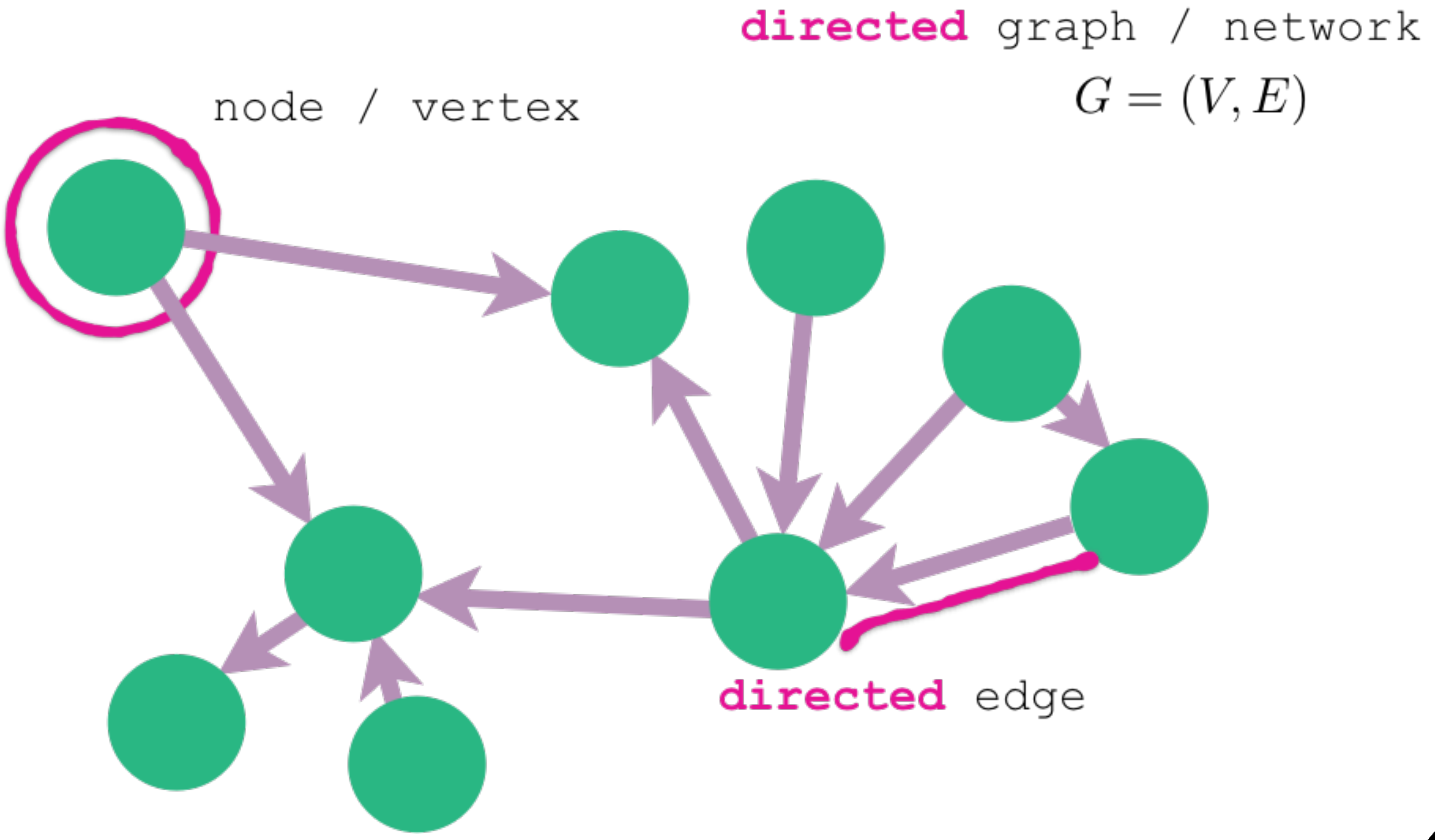


Regular/Suspicious?
Weight: 5 kg
Country: Neverland
Sender: ABC Inc.
Receiver: DEF Inc.
CEO
CTO
CEO
Jeff
Jill
Jake
uni friends
married to
basketball friends
Arms dealer: John

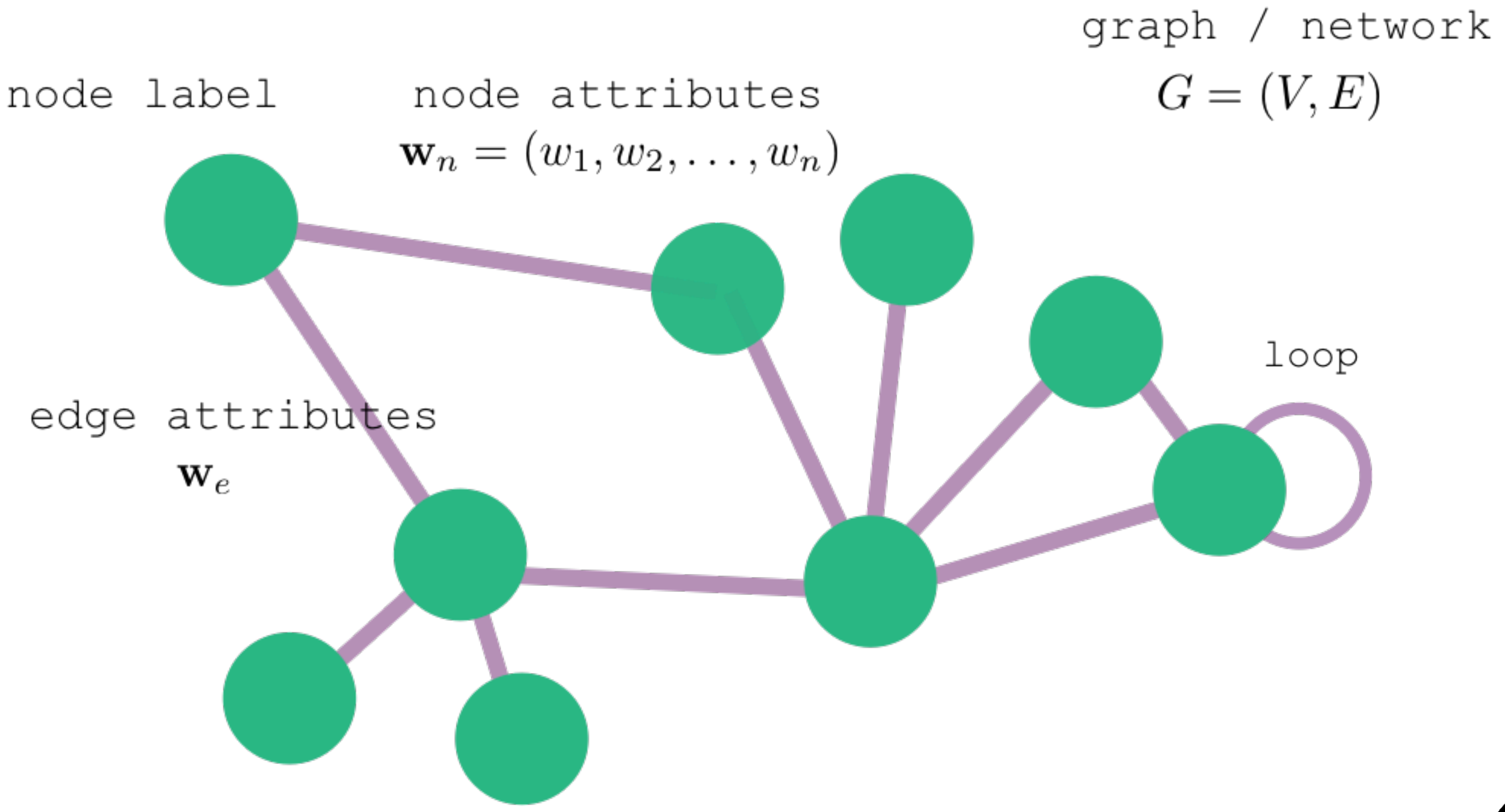
Basic definitions



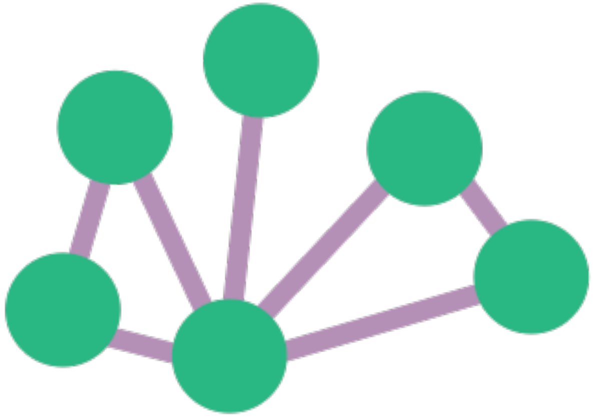
Basic definitions



Basic definitions

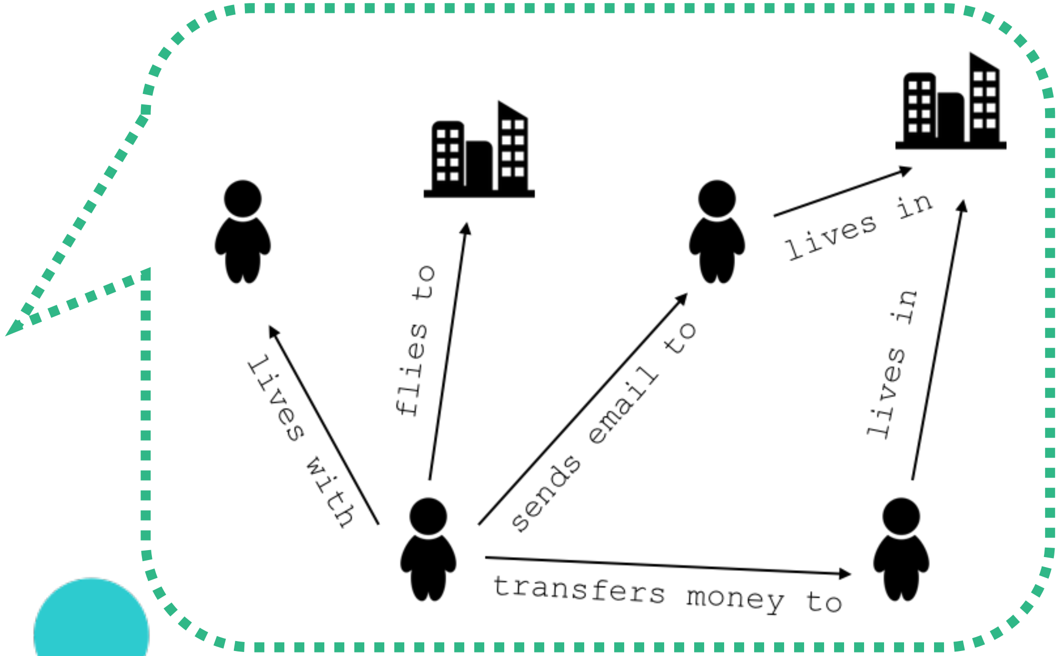


Types of networks

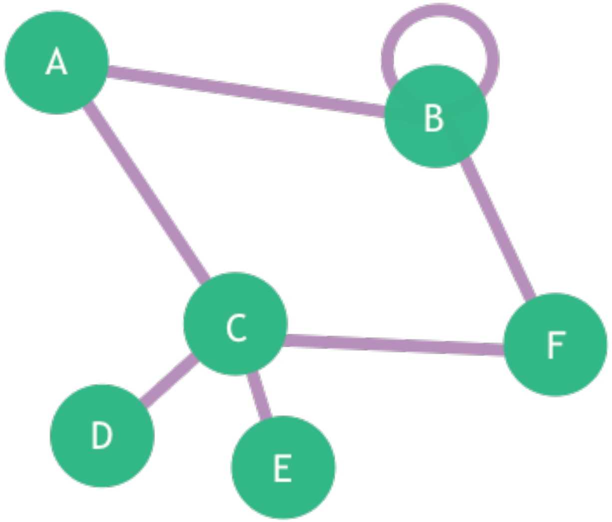


homogeneous

heterogeneous



Graph Representations



Adjacency matrix

	A	B	C	D	E	F
A	0	1	1	0	0	0
B	1	1	0	0	0	1
C	1	0	0	1	1	1
D	0	0	1	0	0	0
E	0	0	1	0	0	0
F	0	1	1	0	0	0

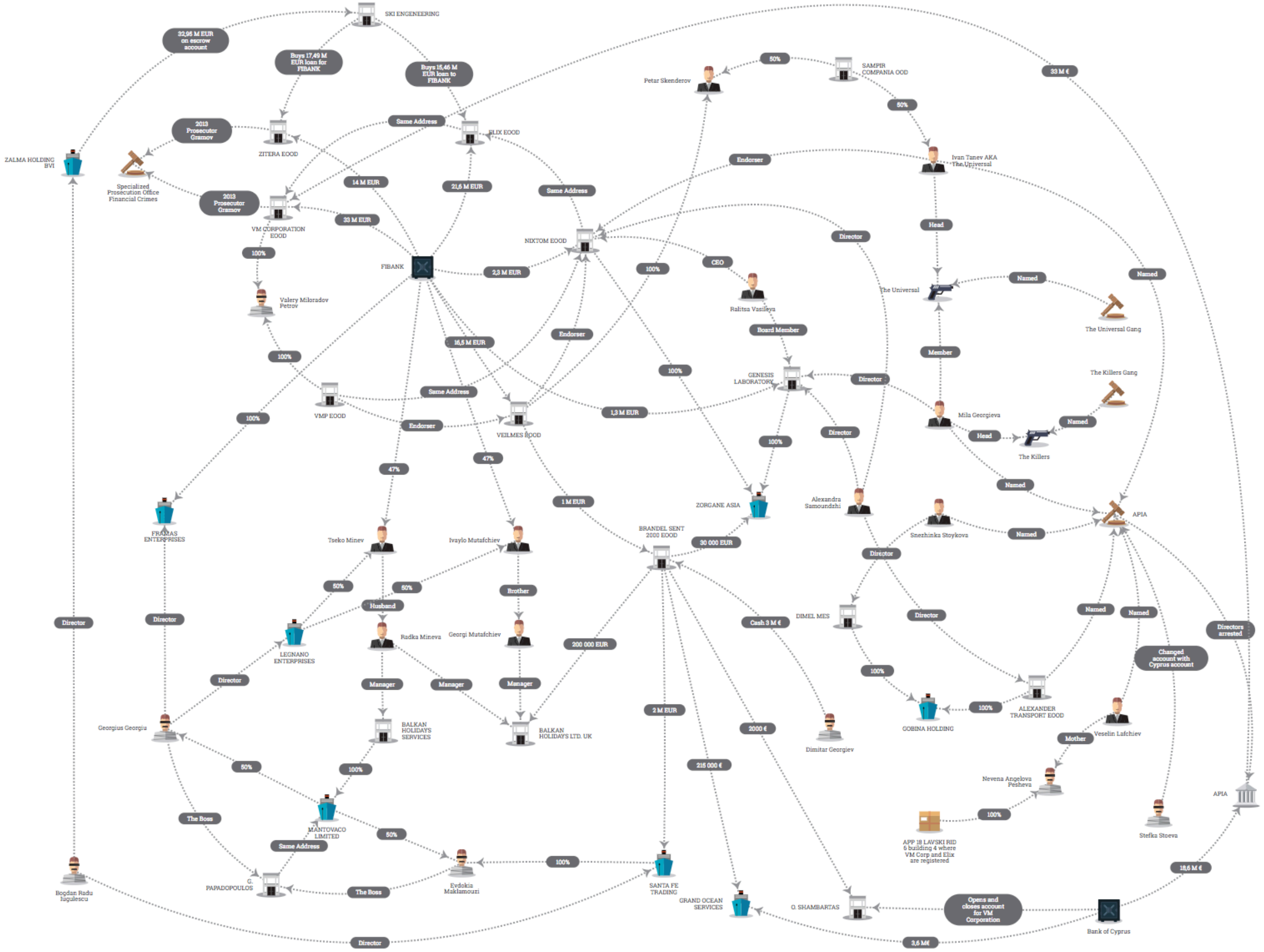
Edge list

```
B,A
B,B
B,F
C,A
C,F
D,C
E,C
```

Adjacency list

```
B: A,B,F
C: A,F,D,E
D: C
E: C
```

Measures of centrality



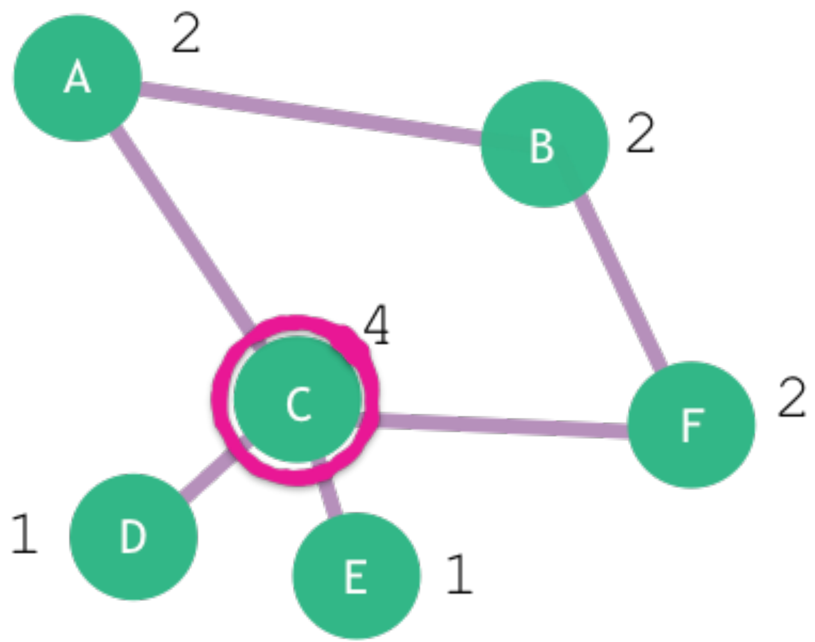
Who to investigate? →
Who is important in the network?

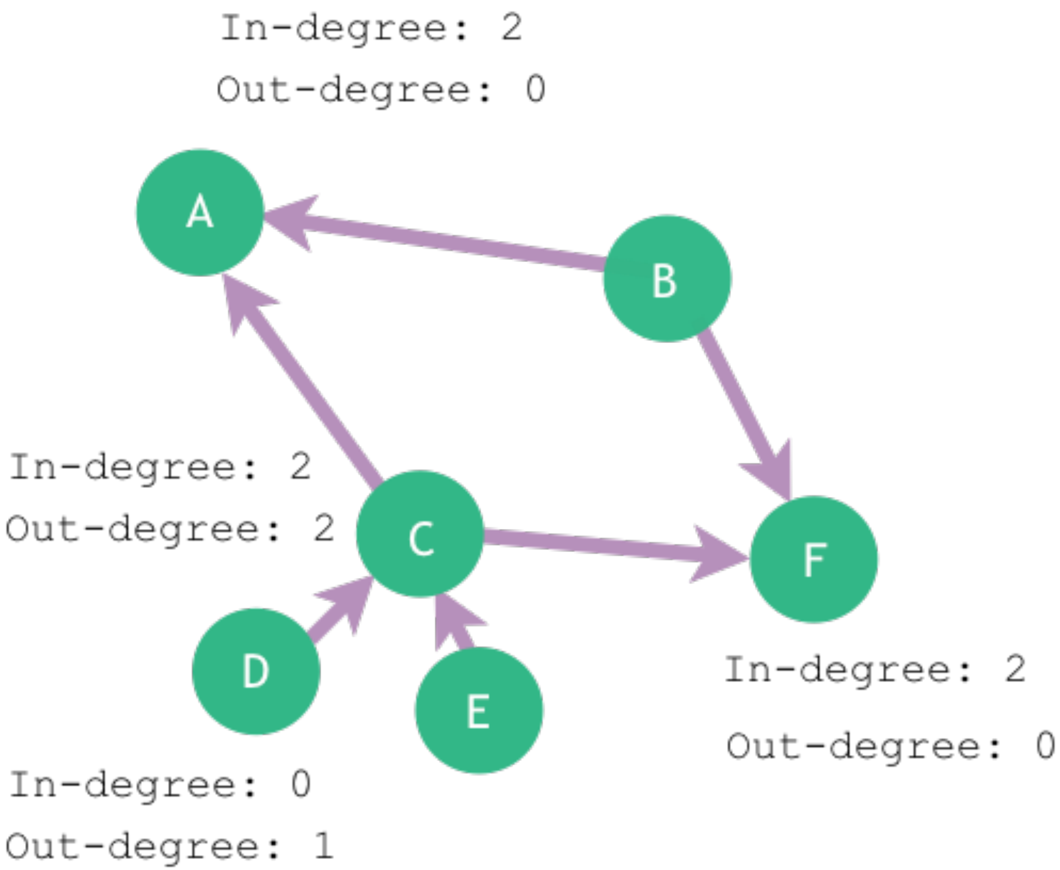
Define **importance**!

vis.occrp.org First Investment Bank Offshore Entities and Romanian APIA connection

Degree

Degree of a node:
number of neighbours of a node

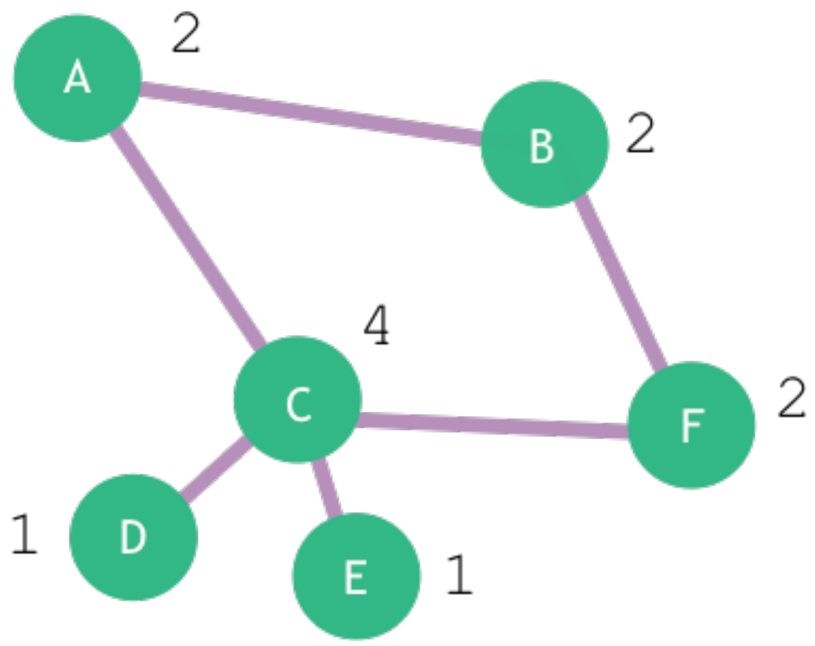


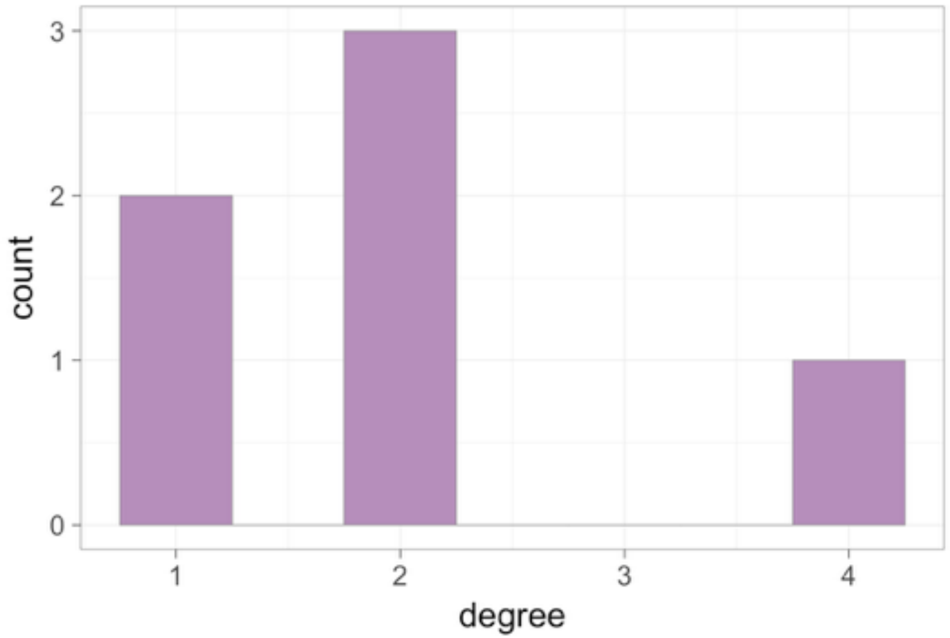


High degree nodes called **hubs**

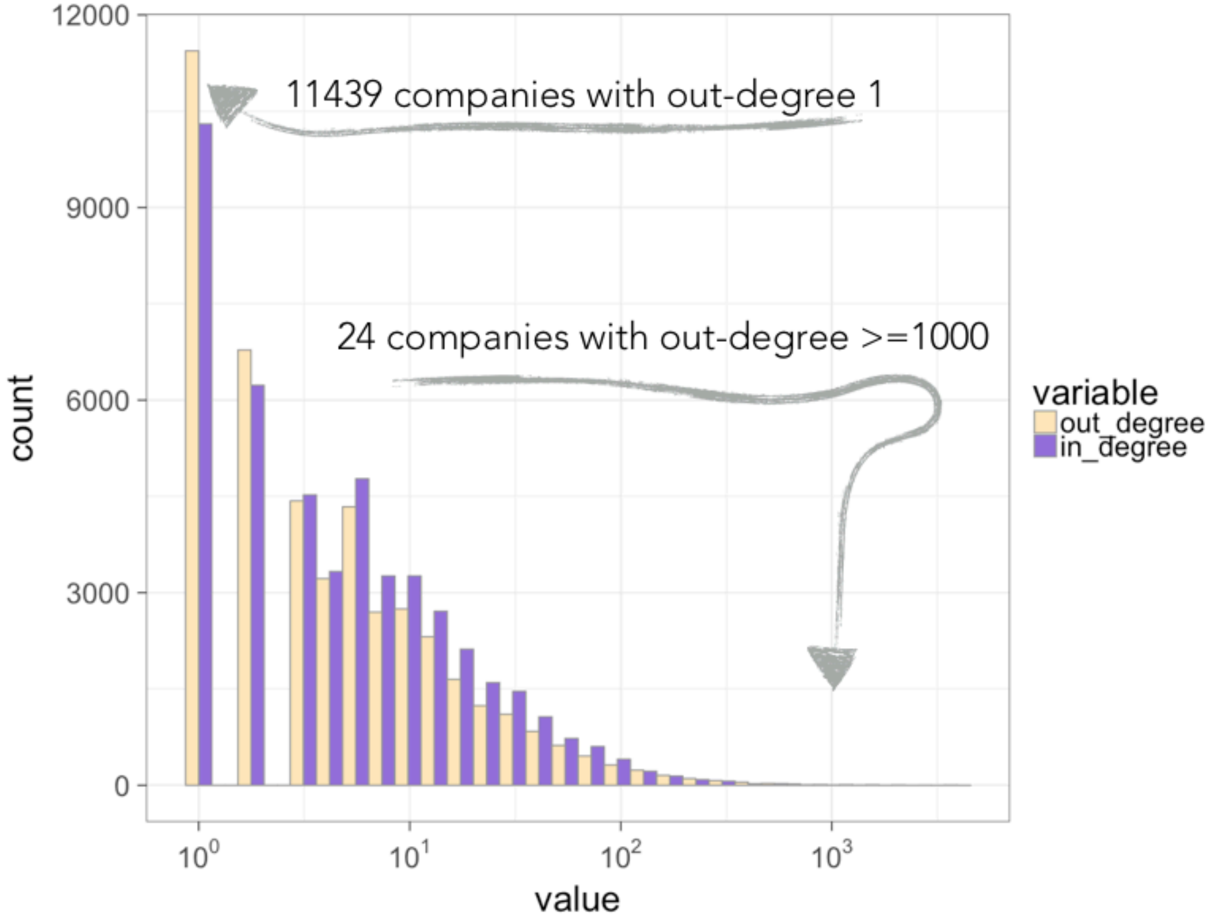
Degree distribution

Probability distribution of the degrees over the entire network



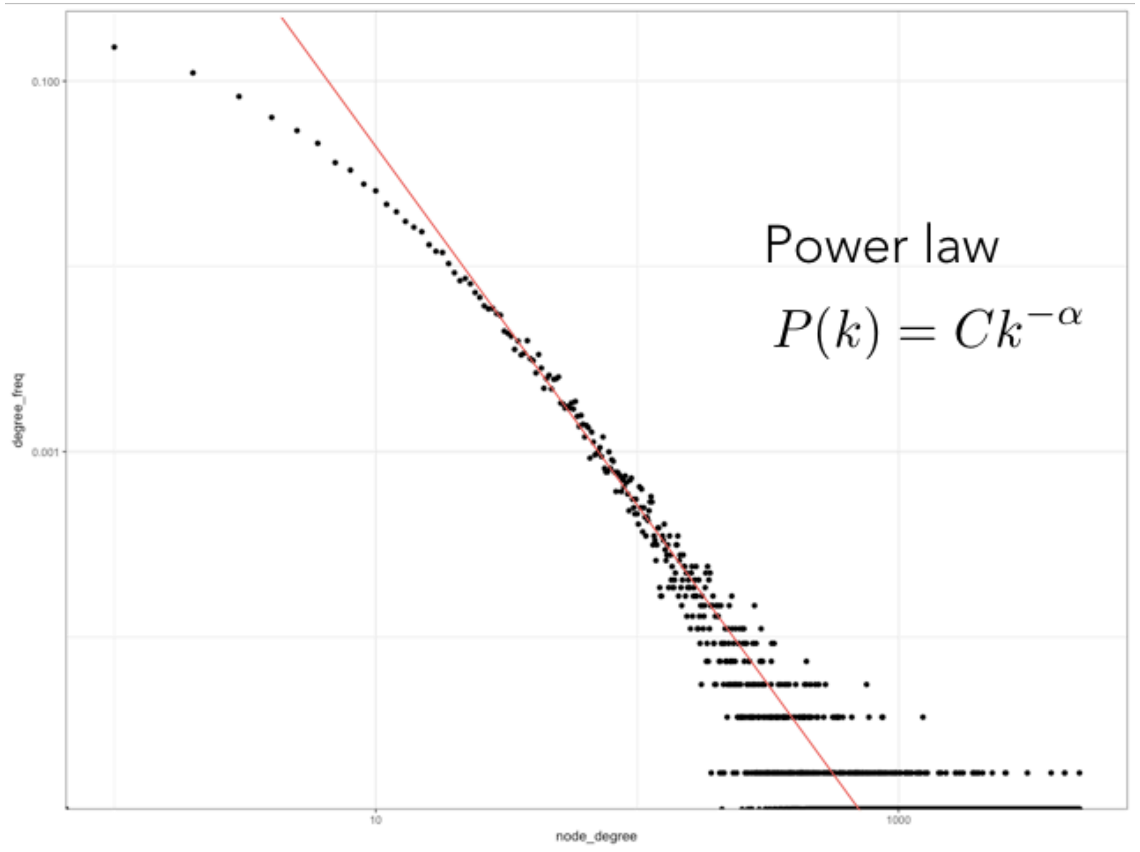


Degree distribution of transactional data between companies

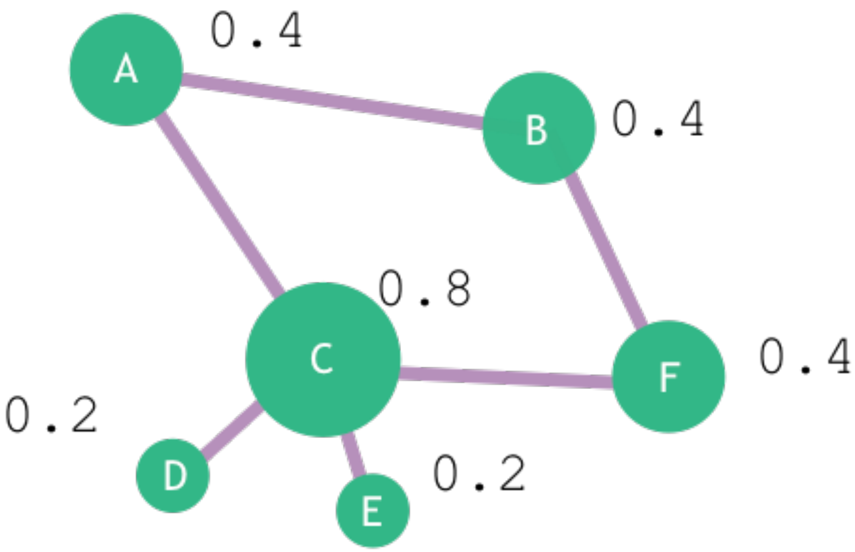


Scale-free networks

log-log scale



Degree centrality

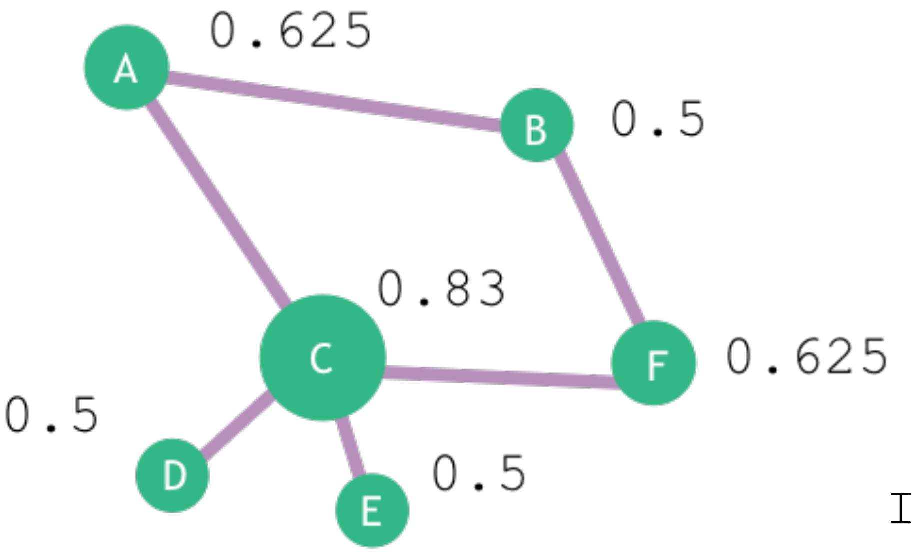


degree of node v

$$C_{deg}(v) = \frac{d_v}{n-1}$$

Important nodes **have many connections**

Closeness centrality



shortest path

number of nodes

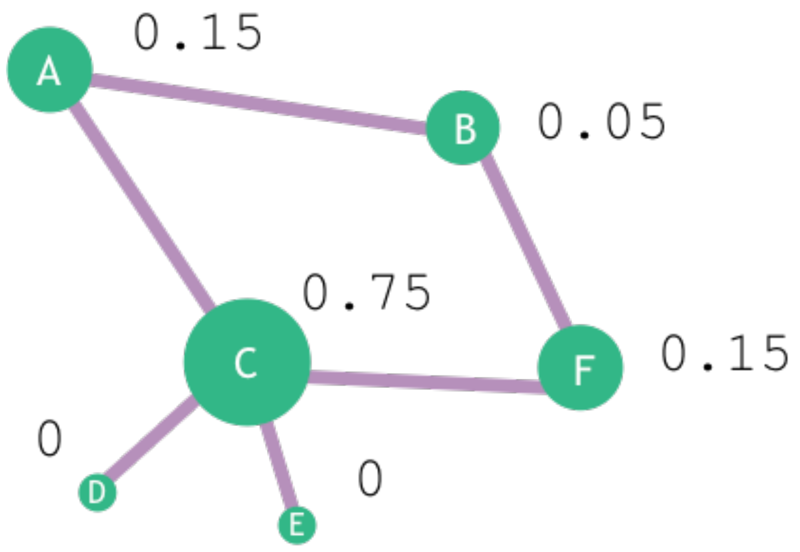
$$C_{closeness}(s) = \frac{n-1}{\sum_{t \in N \setminus \{s\}} d(s,t)}$$

set of nodes

Important nodes are **close to other nodes**

minimal number of
steps to reach others

Betweenness centrality



Many shortest paths go through a high betweenness node

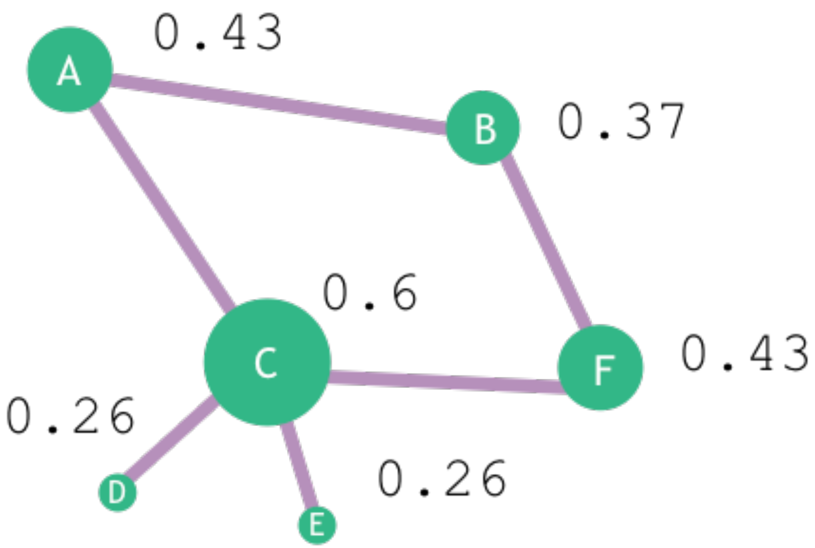
number of shortest paths between s and t that go through v

$$C_{btw}(v) = \sum_{s,t \in N} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

number of shortest paths between s and t

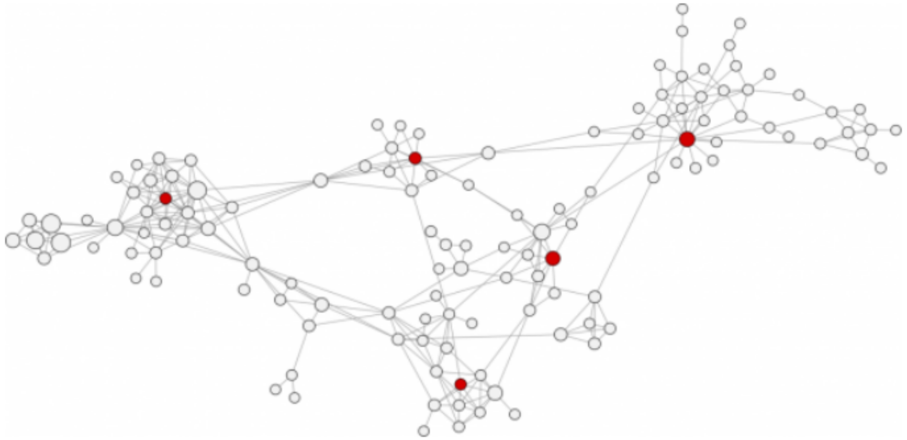
Important nodes **connect other nodes**

Eigenvector centrality

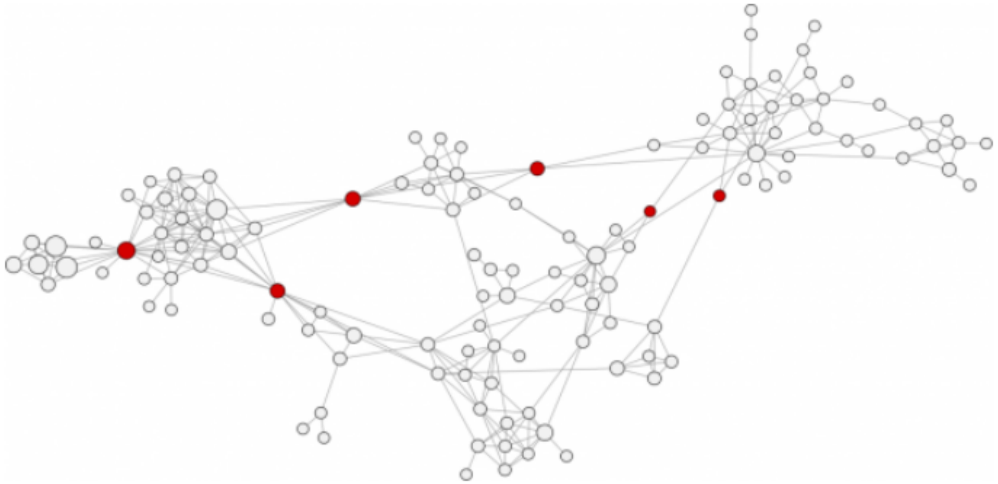


Important nodes have many connections to **other important nodes (recursive)**

Closeness centrality



Betweenness centrality



Eigenvector centrality

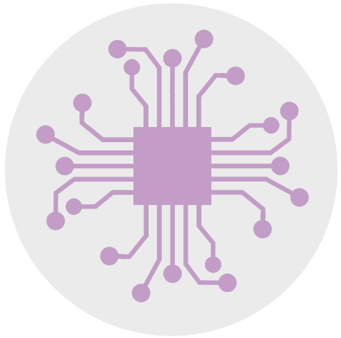
Machine Learning in Practice

A collection of computational methods for automatically detecting patterns in data that can also be used to predict future data.[1]

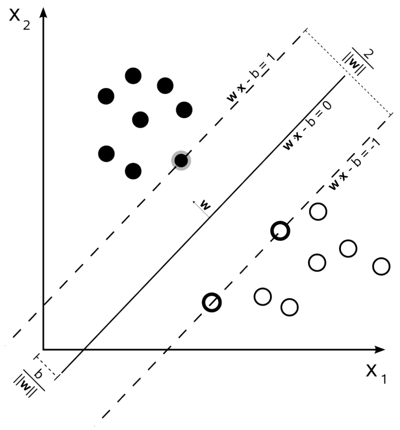
SQL

First	Last	Sex	Income	Age
Homer	Simpson	M	85000	39
Marge	Simpson	F	0	36
Bart	Simpson	M	?	10
Lisa	Simpson	F	0	8
Carl	Carlson	M	90000	40
Fat	Tony	M	15000	45
Ned	Flanders	M	72400	38

Machine Learning



Model



[1] Machine Learning: A Probabilistic Perspective, Kevin P. Murphy, The MIT Press, 2012

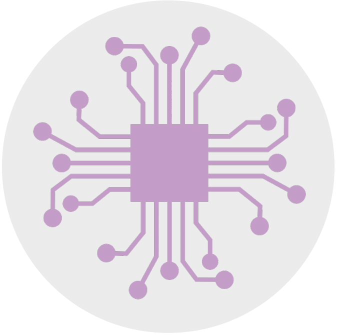
Machine Learning in Practice

A collection of computational methods for automatically detecting patterns in data that can also be used to predict future data.[1]

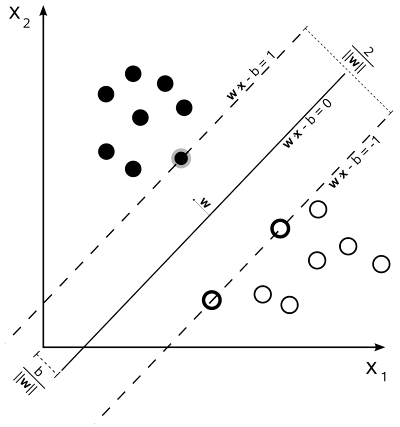
SQL

First	Last	Degree	Betweenness	Age
Homer	Simpson	3	1.2	39
Marge	Simpson	5	0.3	36
Bart	Simpson	4	0.1	10
Lisa	Simpson	18	0.001	8
Carl	Carlson	1	0.8	40
Fat	Tony	3	0.54	45
Ned	Flanders	2	0.001	38

Machine Learning

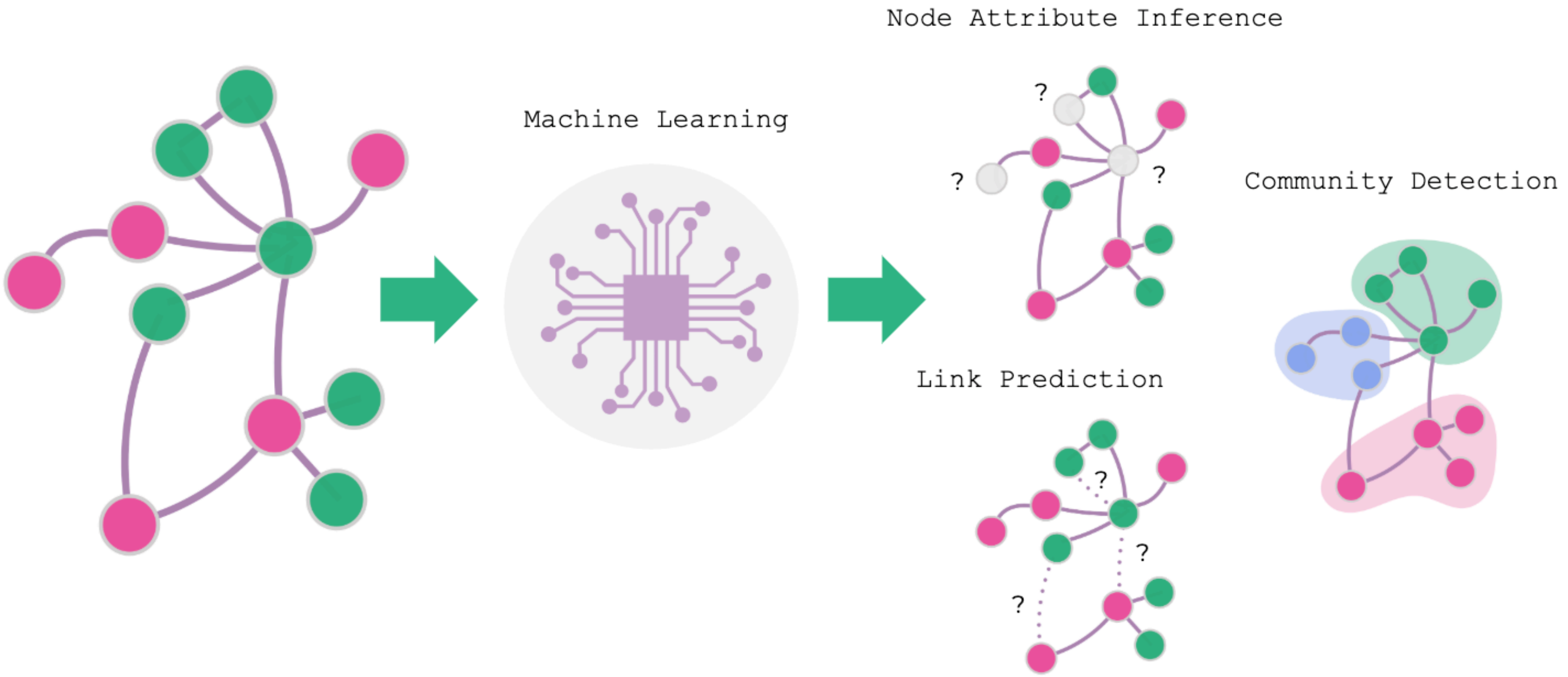


Model

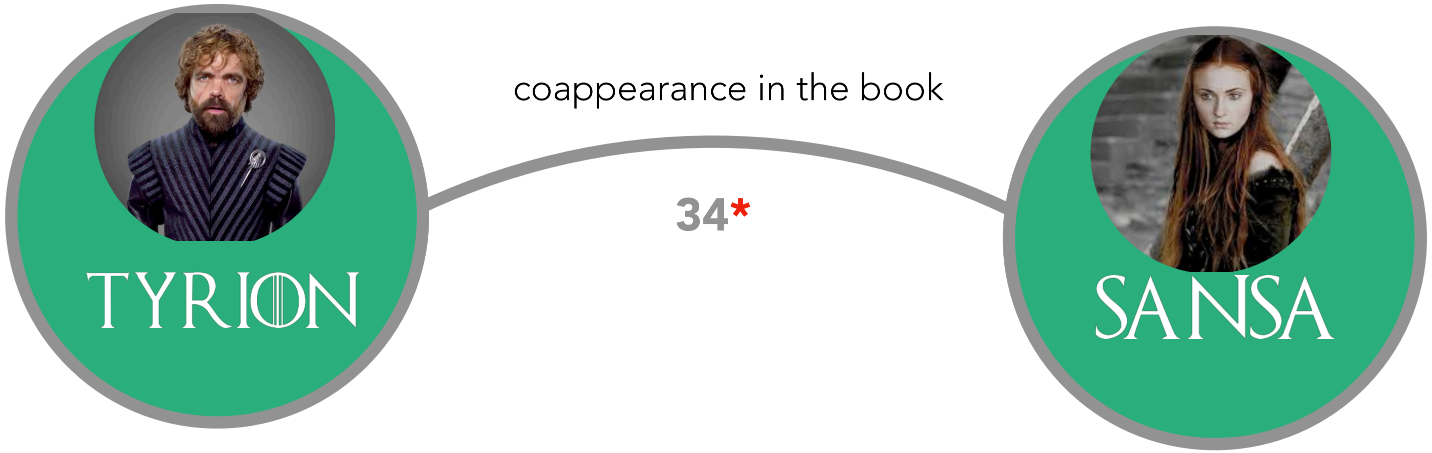


[1] Machine Learning: A Probabilistic Perspective, Kevin P. Murphy, The MIT Press, 2012

Common tasks on graphs

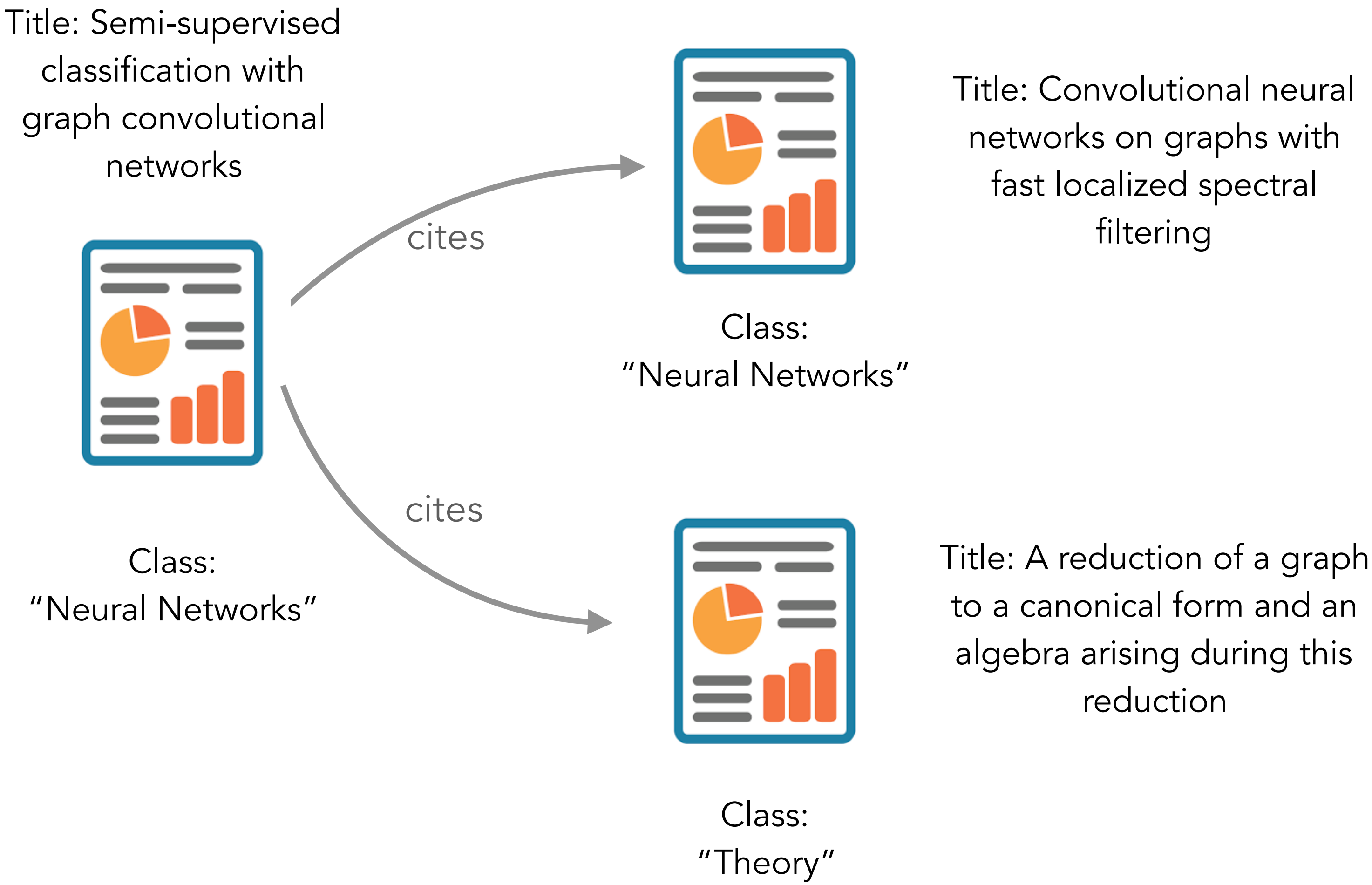


Dataset description: Game of Thrones



* weight: number of times the two characters' names appeared within 15 words of each other in the text

Dataset description: CORA network



Dataset description: CORA features

ABSTRACT

We present a scalable approach for semi-supervised learning on graph-structured data that is based on an efficient variant of convolutional neural networks which operate directly on graphs. We motivate the choice of our convolutional architecture via a localized first-order approximation of spectral graph convolutions. Our model scales linearly in the number of graph edges and learns hidden layer representations that encode both local graph structure and features of nodes. In a number of experiments on citation networks and on a knowledge graph dataset we demonstrate that our approach outperforms related methods by a significant margin.

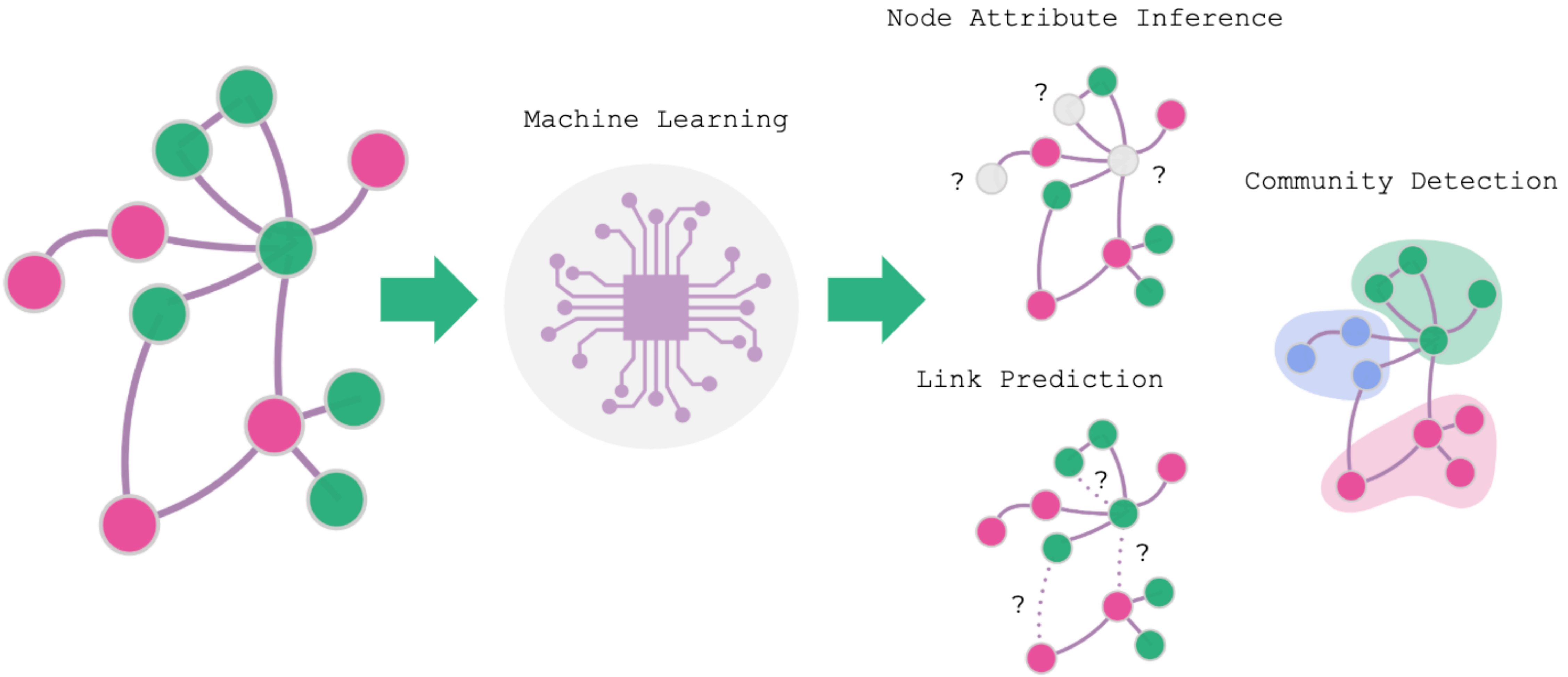
Title: Semi-supervised classification with graph convolutional networks

Class: "Neural Networks"

	w_1: learning	w_2: genetic	w_3: model	…
Semi-supervised classification ..	1	0	1	…
Convolutional neural networks…	1	0	0	…
…				…

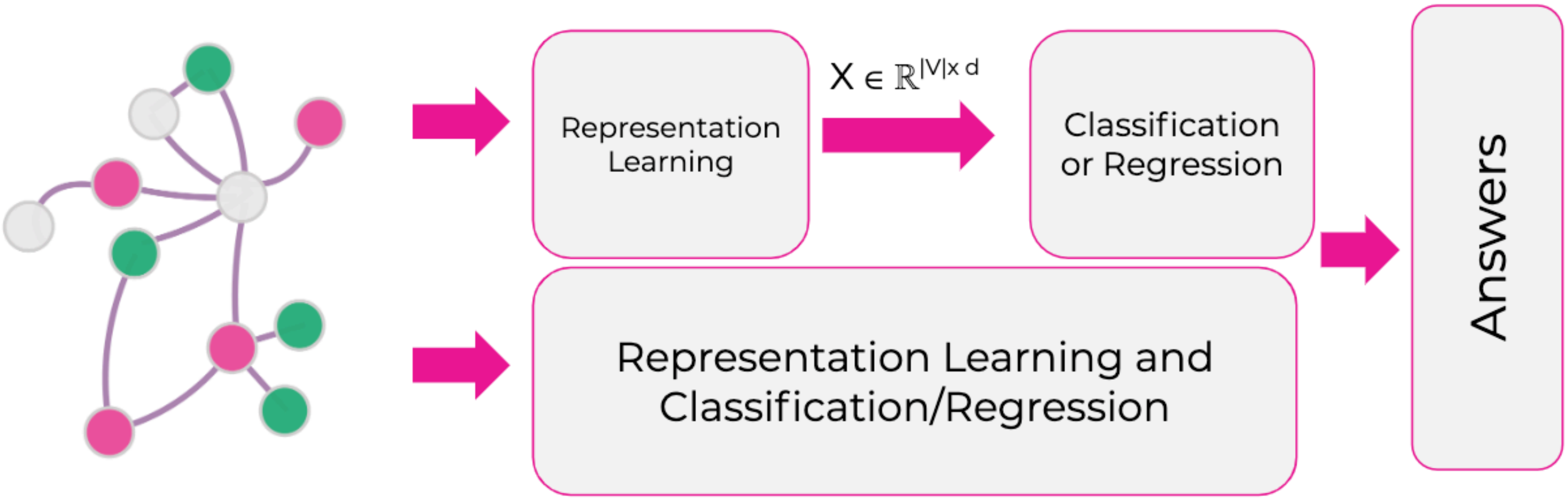
Traditional SNA notebook

Common tasks on graphs



Representation Learning

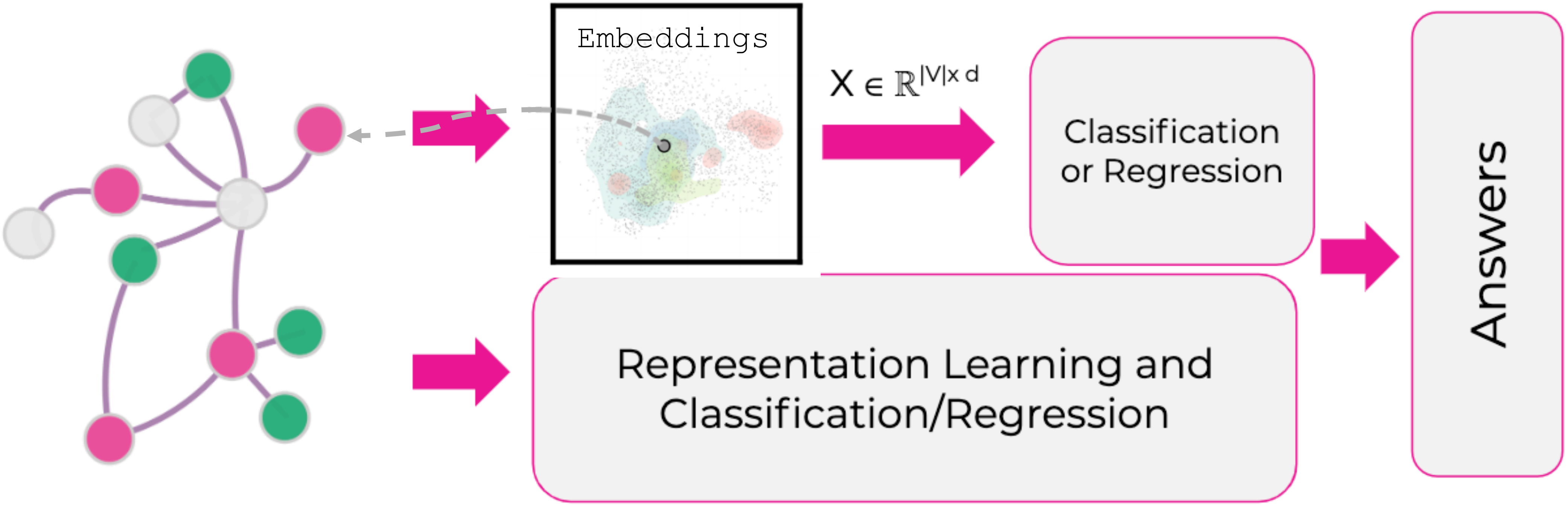
Recall: Graph $G = (V, E)$ where $v_i \in V$ and $e_i \in E$



Properties for $x_i \in X$:
"Similar" nodes should have similar feature vectors

Representation Learning

Recall: Graph $G = (V, E)$ where $v_i \in V$ and $e_i \in E$



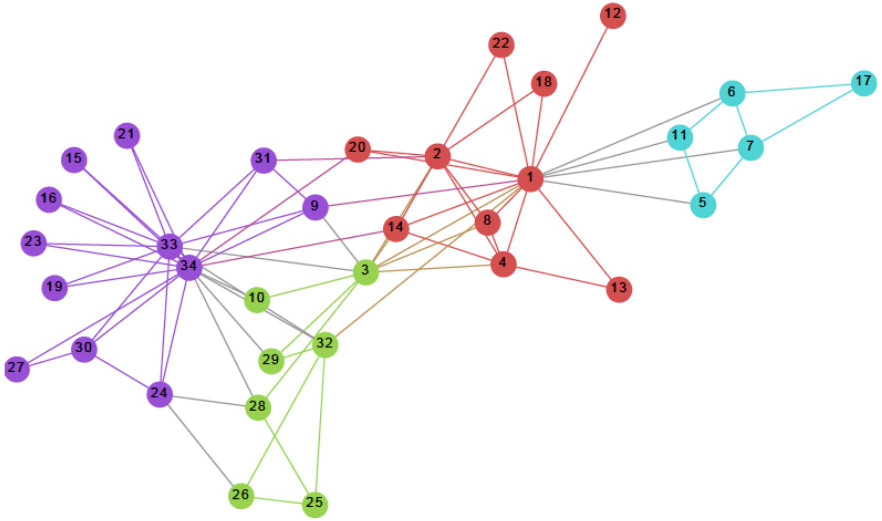
Properties for $x_i \in X$:
"Similar" nodes should have similar feature vectors

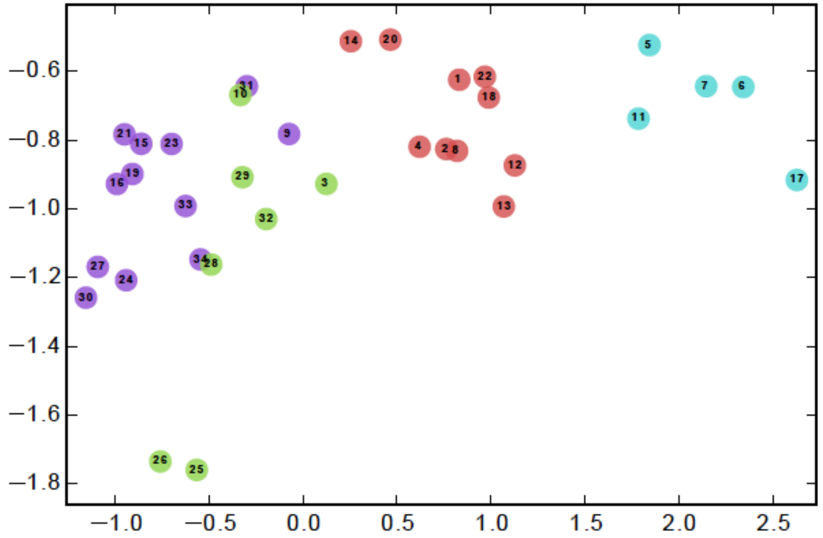
Representation Learning Example

- Zachary's Karate Club Network

Input: Graph

Output: 2D node feature vectors





DeepWalk: Online Learning of Social Representations, B. Perozzi, R. Al-Rfou, and S. Skiena, KDD 2014

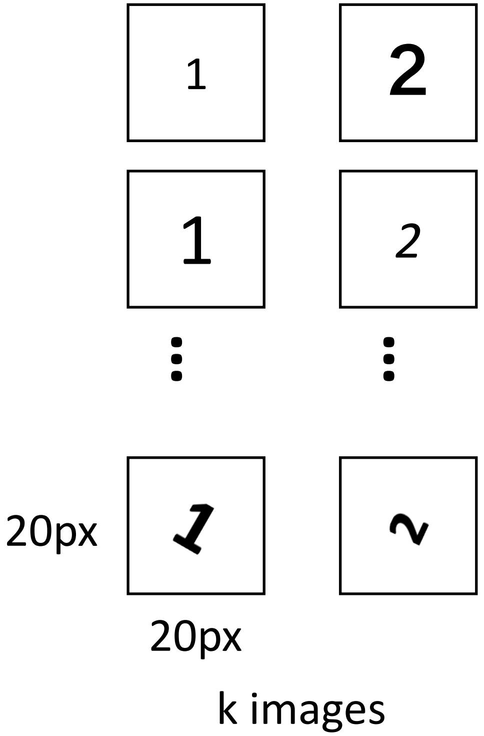
Transductive vs Inductive

- Transductive
  - Graph is static
- Inductive
- Graph is dynamic
  - Given a new node in the graph how can we quickly determine a feature vector for it?
- Transfer Learning
  - Train model on one graph and use learned model to answer questions on another (but similar) graph

`Laplacian Eigenmaps` → `DeepWalk` → `Graph Convolutional Networks (GCN)`

Laplacian Eigenmaps

- Representation learning using the geometry of the data
- Given are N data $\mathbf{X_i} \in R^F$ and labels $\mathbf{l}$.



$$A = \begin{cases} 1 & \text{if } A_{ij} \text{ edge} \\ 0 & \text{otherwise} \end{cases}$$

$$D \text{ such that } D_{ii} = \sum_j A_{ji}$$

$$L = D - A$$

$$\text{eigenvectors of L } \mathbf{y_0}, \mathbf{y_1}, \ldots, \mathbf{y_{k-1}}$$

$$\text{image of i-th point is } (\mathbf{y_1}(\mathbf{i}), \ldots, \mathbf{y_m}(\mathbf{i}))$$

Laplacian Eigenmaps on Cora

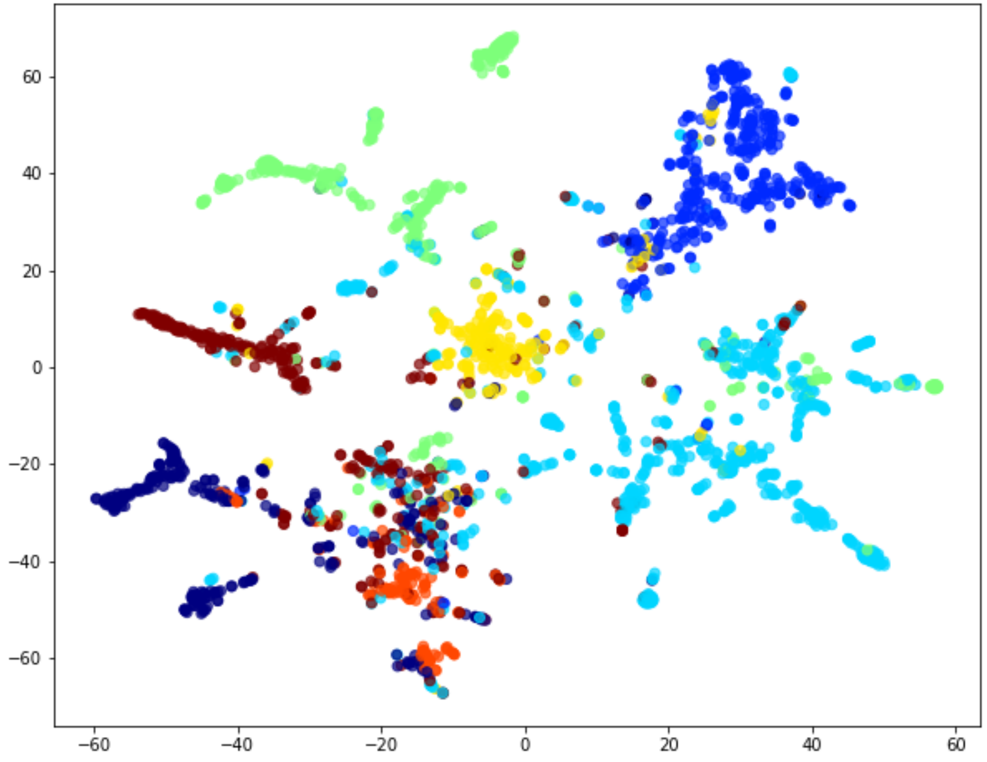
- Let's apply this method on the Cora citation network dataset
- Cora dataset
  - nodes are papers (2485), and
  - edges are citation links between papers (5069)

Exercise

- Can you improve the results?
- Number of components to keep, 16, 32, 128,…
- Different Laplacian,
    - Normalized Laplacian: $I-D^{-1/2}AD^{-1/2}$
    - Random Walk Laplacian: $I-D^{-1}A$
- Different classifier, e.g., Logistic Regression, ensemble?
- Utilize node features

Laplacian Eigenmaps: Cora Example

- Cora citation dataset: nodes are papers (2485) and edges are citation links between papers (5069)



Accuracy of Random Forest and 32 dimensions and 140 training examples prediction is ~73%

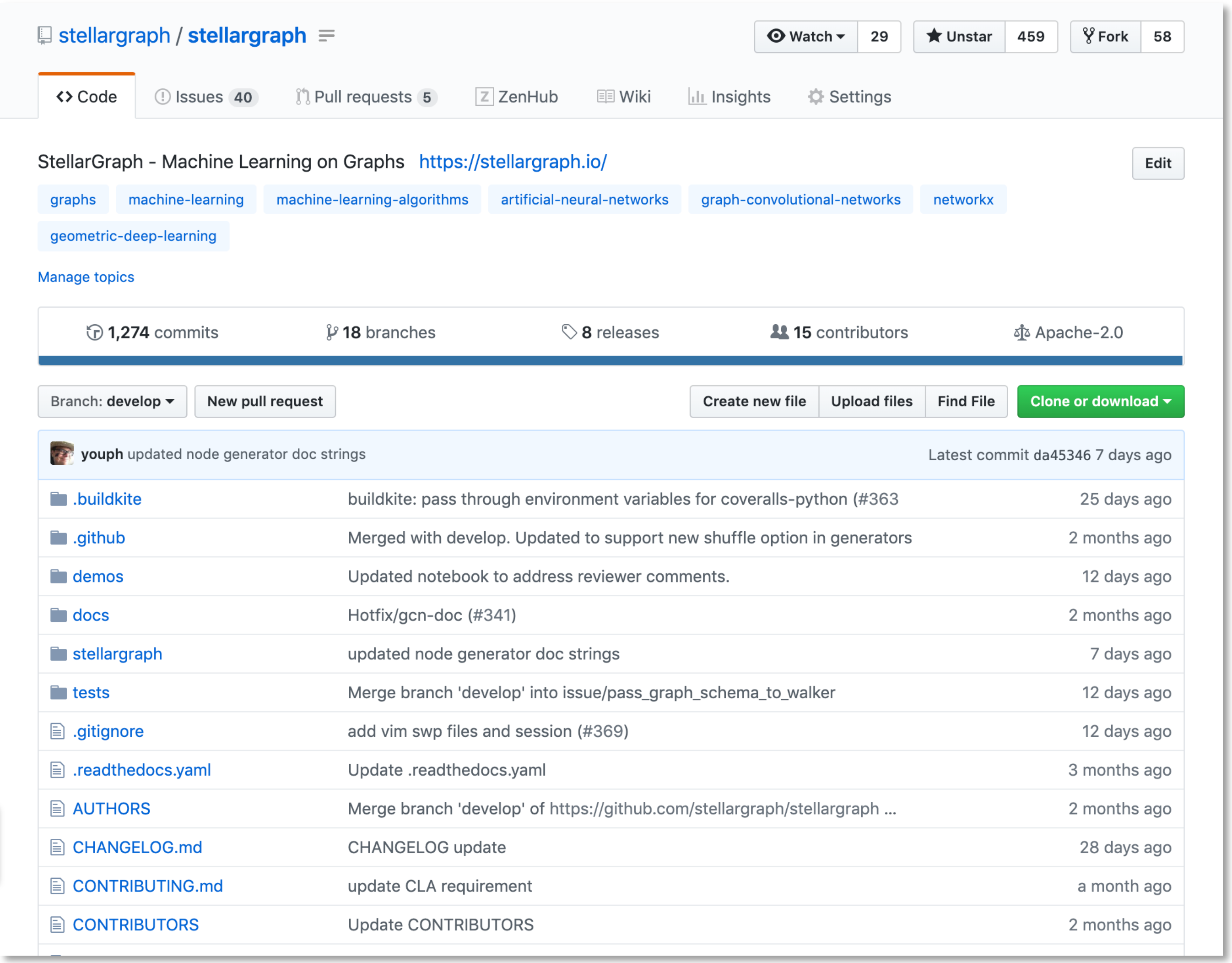
StellarGraph

- Python Graph ML library
- Open source Apache 2.0 License

https://github.com/stellargraph/stellargraph

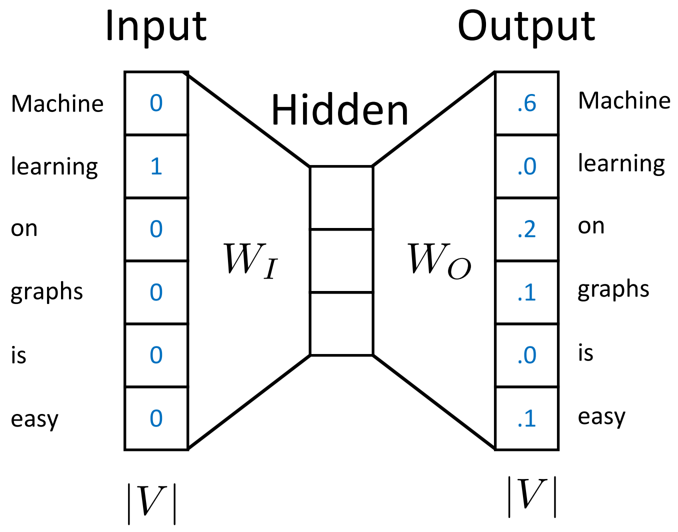
https://stellargraph.readthedocs.io/en/stable

https://community.stellargraph.io



DeepWalk: Core Idea

- The skip-gram model in Natural Language Processing (Word2Vec)
- Consider the sentence: **Machine learning on graphs is easy**
  - Window size is 2, and dictionary size |V| = 6
  - Data: [learning: (Machine, on)], [on: (learning, graphs)], [graphs: (on, is)],…
  - Skip-gram model: Predict context from word, e.g., predict (*Machine, on*) from *learning*



Important

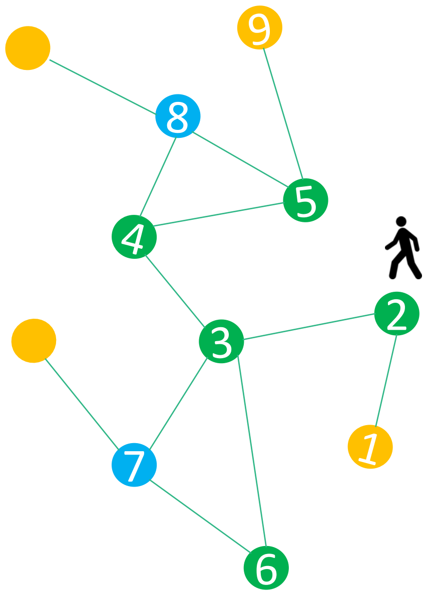
- Hidden layer is linear
- Two projections matrices: $\mathbf{W_I}$ (from input to hidden) and $\mathbf{W_O}$ (from hidden to output)

Output is Softmax: $p(w_O|w_I) = \frac{e^{{v'}_{w_O}^{T} v_{w_I}}}{\sum_{w=1}^{|V|} e^{{v'}_w^{T} v_{w_I}}}$

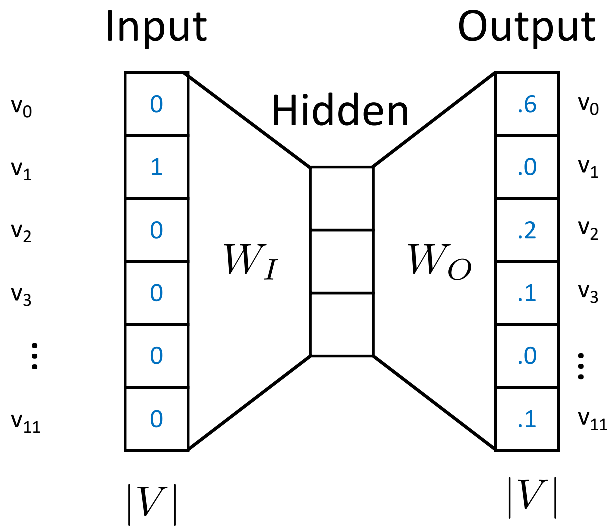
Objective: $max \frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-c \leq j \leq c \\ j \neq 0}} logp(w_{t+j}|w_t)$

DeepWalk: From words to graphs

- What is a sentence for a graph?
- Idea: Generate sentences by random walks

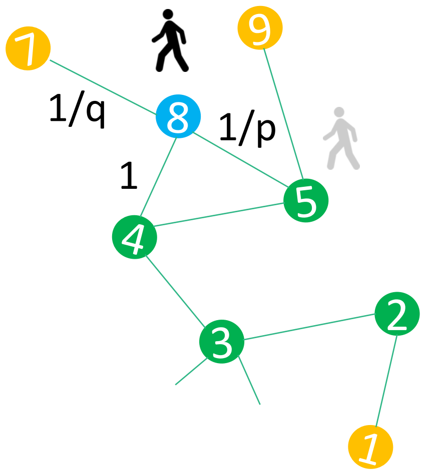


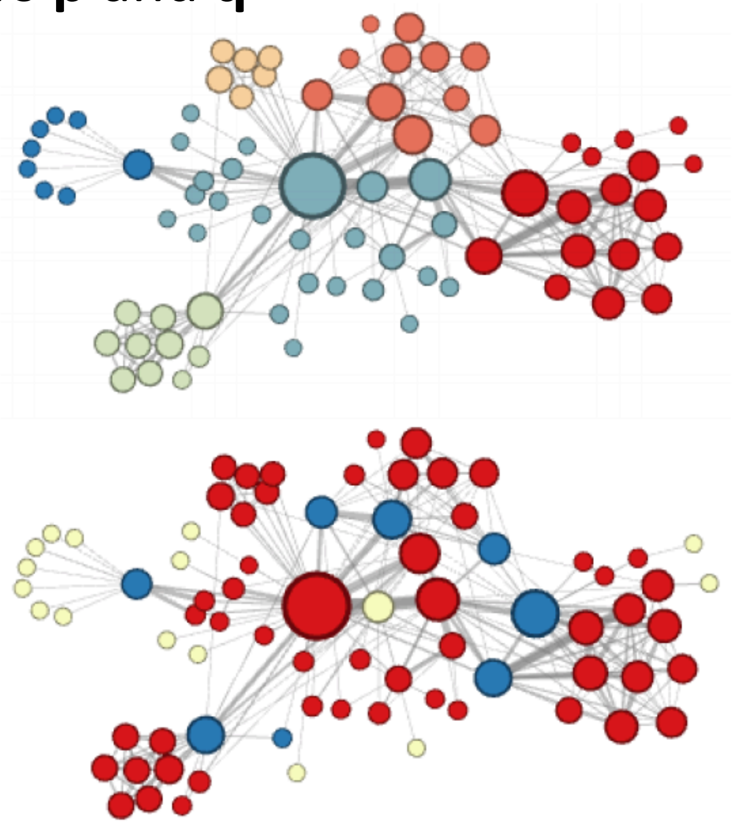
Random walk properties
- Sequence of node IDs: [2,3,4,8,…]
- Next node sampled uniformly at random
- Walk length is fixed
- Start from every node in graph
- Make several passes over graph



Node2Vec

- DeepWalk with biased, second order, random walks
- Node2Vec introduces parameters **p** and **q**





Homophily
(p=1, q = 0.5)

Structural
Equivalence
(p=1, q=2)

Node2Vec: Scalable Feature Learning for Networks, A. Grover and J. Leskovec, KDD 2016

DeepWalk and Node2Vec on Cora

- Notebook: Node Attribute Inference

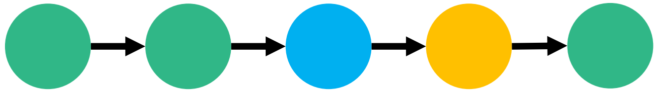
Exercise

- Can you improve the results?
- Use different values for p, q to bias the random walker
- Use different values for number of walks per node and walk length
- Use different values for Word2Vec parameters
- Use different downstream classifier, e.g., Random Forest, ensemble, etc.

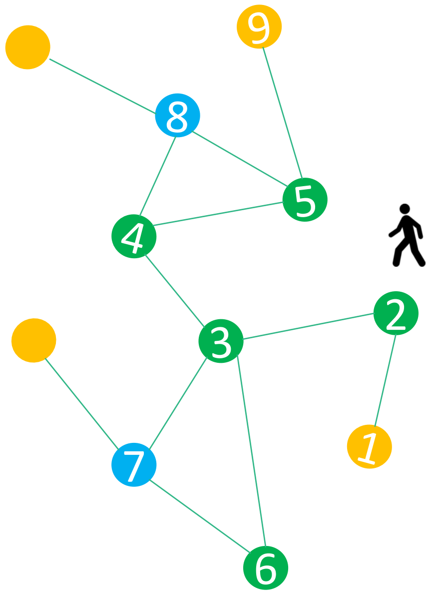
Metapath-driven random walks

- Metapath2vec: *Scalable representation learning for heterogeneous networks*, Y. Dong, N. V. Chawla, and A. Swami, KDD 2017

Metapath:

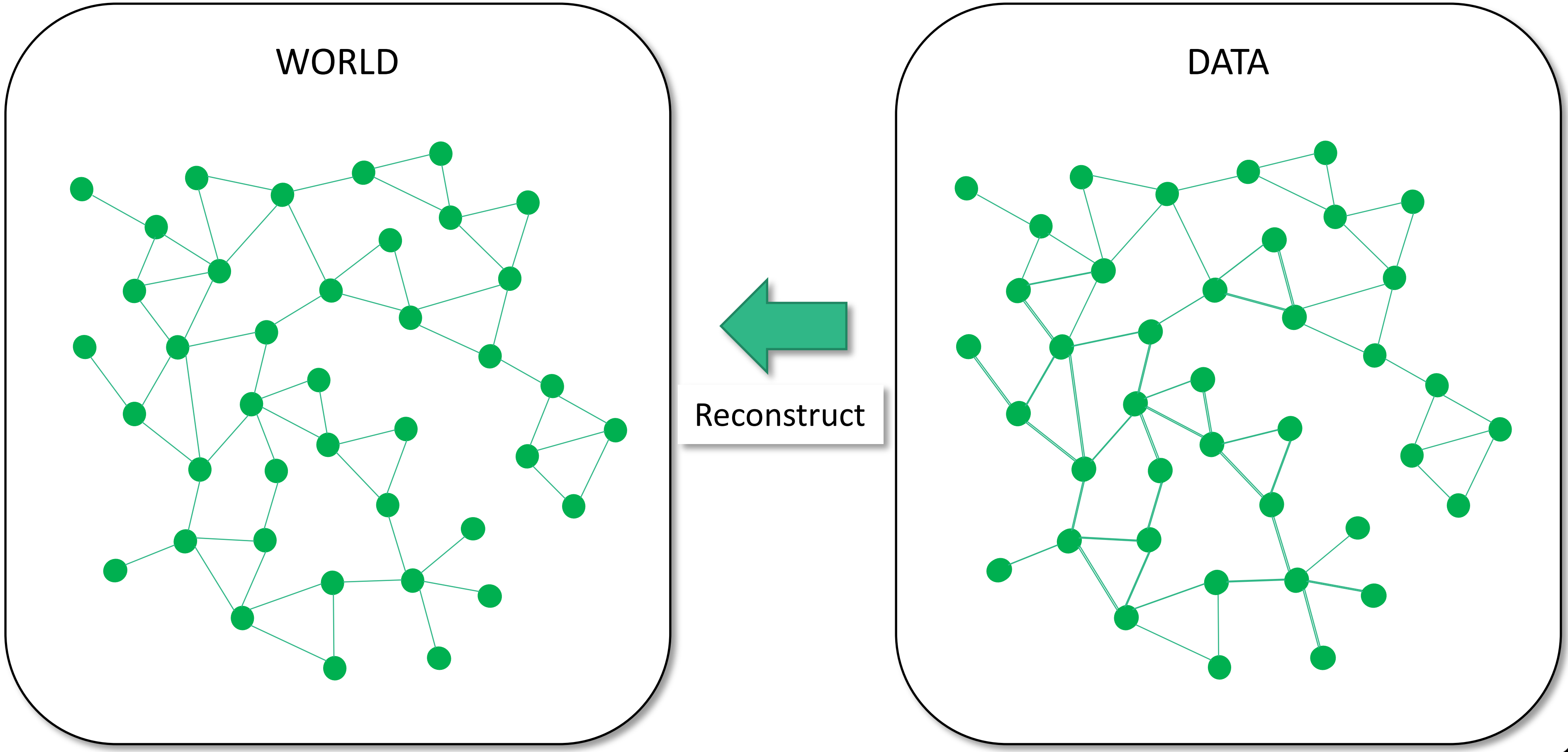


Color denotes node type



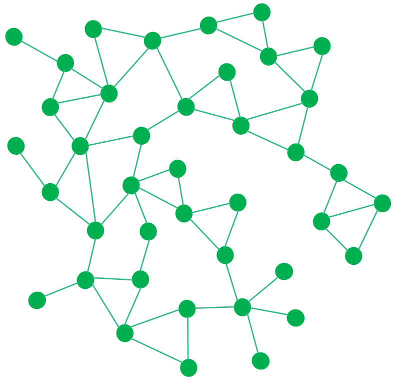
Link Prediction Problem Overview

Data is a noisy and partial observation of the world!



Link Prediction Big Picture

Data



● [0.1, -0.2, 0.03, 0.01, 0.9,…]

● [0.2, 0.2, -0.05, -0.1, 0.9,…]

● [0.1, -0.2, -0.5, 0.1, -0.5,…]

⋮

Representation Learning (Nodes)

Edge Features

f(●, ●) = [0.1, 0.9, 0.1, -0.5,…]

Solve a classification problem

Edge data
- Positive
- Negative

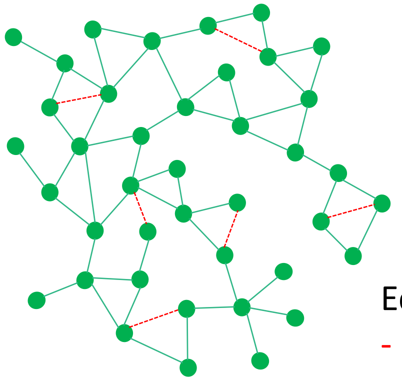
\+

Edge features

f(●, ●)

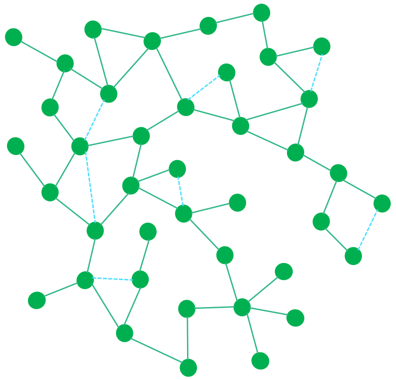
Classifier

Test



Edge data
- Positive
- Negative

Train

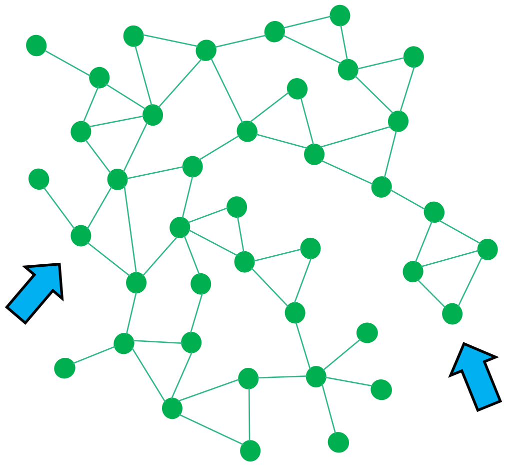


Edge data
- Positive
- Negative

Data Splitting

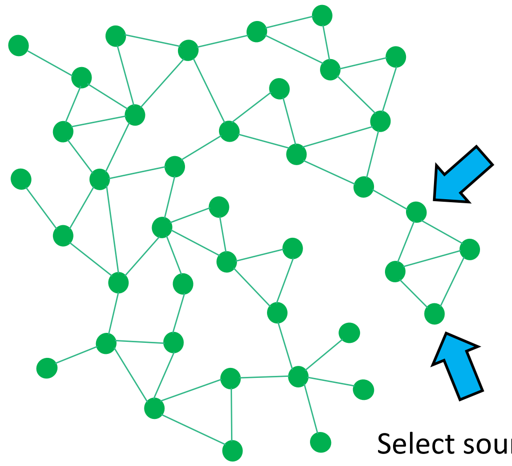
- Positive edge samples = Easy
- Negative edge samples = Not so easy!
  - 2 approaches: Global vs Local

Global



Connected by edge?

Local



Select source node

Sample distance to target node, e.g., 2 hops

Connected by edge?

DeepWalk/Node2Vec Link Prediction

- What are link/edge representations/embeddings?
  - An edge connects 2 nodes.
  - The edge embedding can be a function of the node embeddings
  - Operators:
    - Dot product
    - Mean
    - Cosine distance
    - Concatenation
    - etc.
- How do we sample positive and negative examples?
  - Stellargraph has ***stellargraph.data.EdgeSplitter*** class
  - https://stellargraph.readthedocs.io/en/stable/api.html#module-stellargraph.data

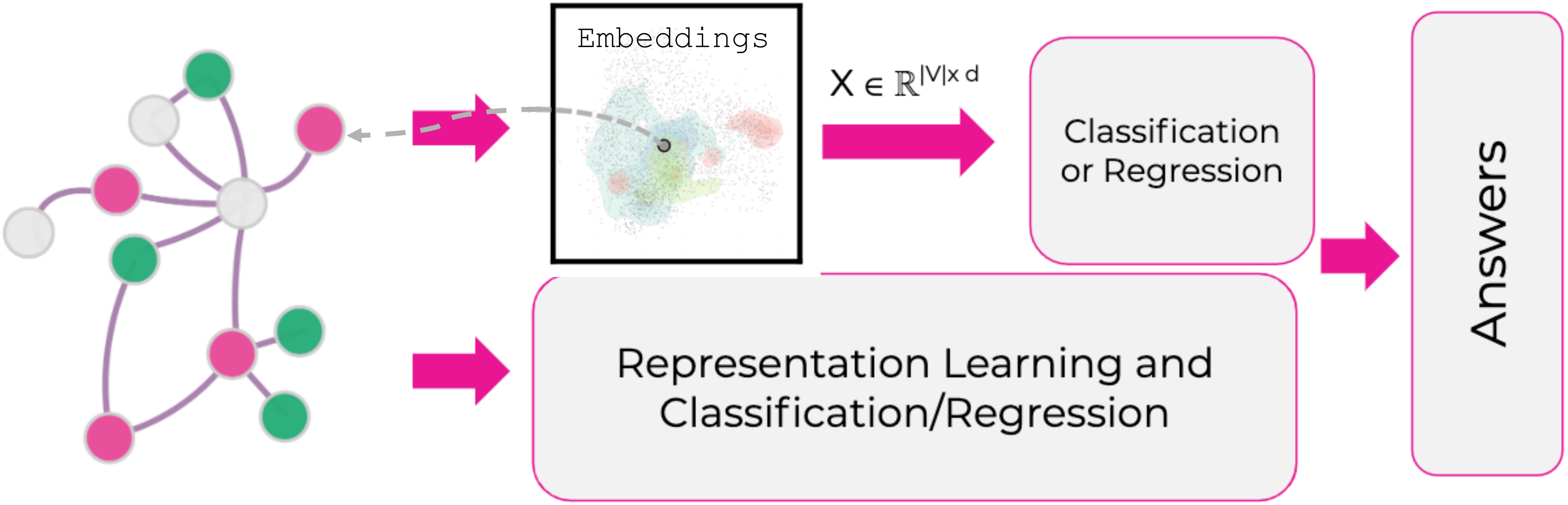
Link Prediction on Cora

Exercise

- Can you do better?
- Different operator for edge embeddings
- Different parameters for Node2Vec
- Local vs Global edge sampling
  - https://stellargraph.readthedocs.io/en/stable/api.html#module-stellargraph.data
- Different binary classifier

Representation Learning

Recall: Graph $G = (V, E)$ where $v_i \in V$ and $e_i \in E$



Properties for $x_i \in X$:
"Similar" nodes should have similar feature vectors

Convolutional Neural Networks 101

- Powerful models for structured data

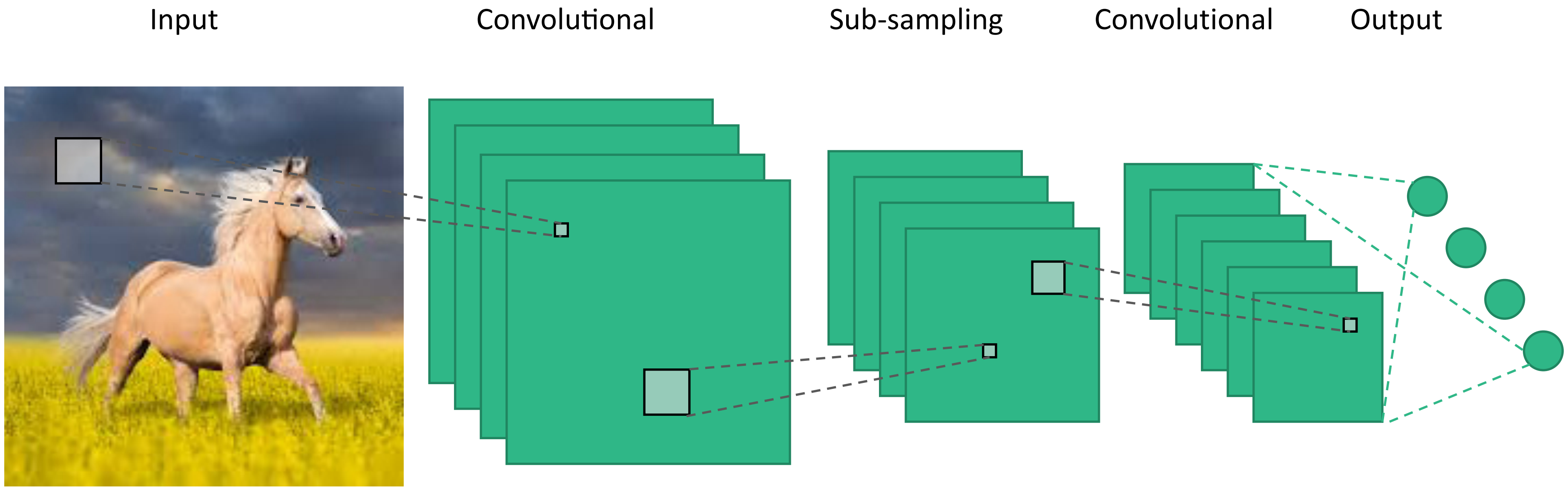
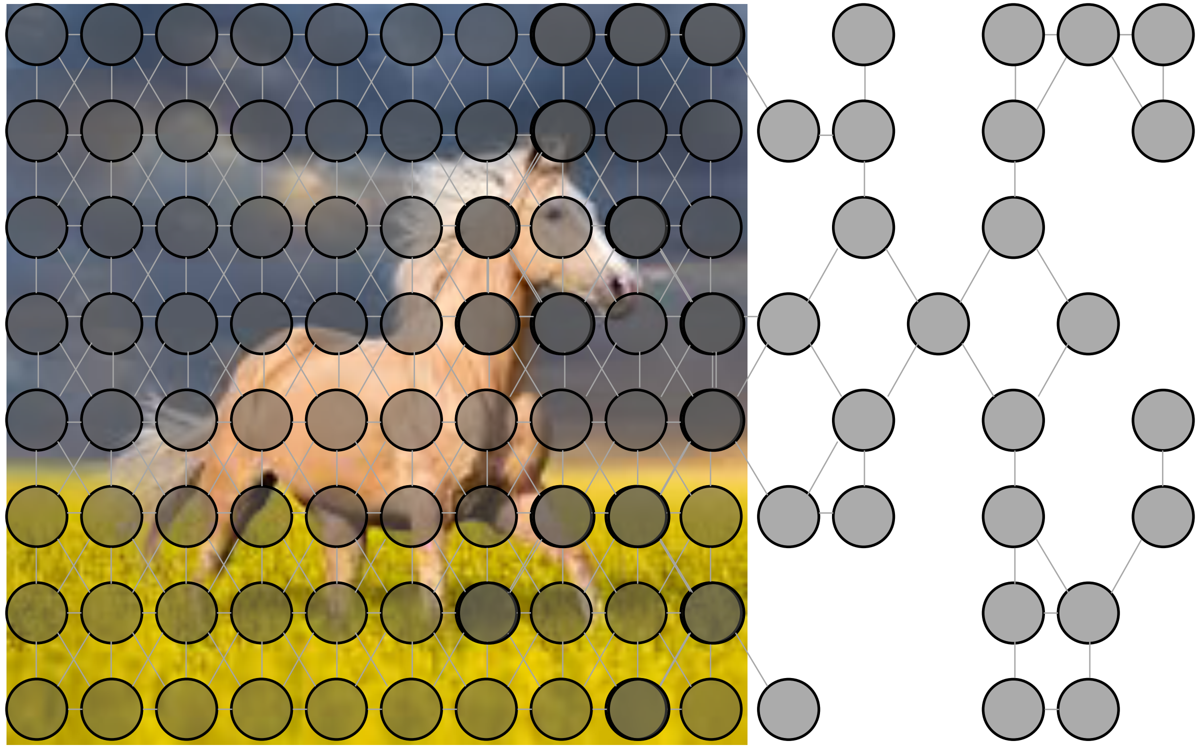


Image Filter Output

$$\begin{matrix} 0 & 0 & 4 & 0 & 3 & 0 \\ 3 & 1 & 1 & 2 & 0 & 0 \\ 0 & 2 & 5 & 0 & 2 & 1 \\ 0 & 2 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 4 & 0 & 2 \end{matrix} \quad * \quad \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix} \quad = \quad \begin{matrix} 7 & -1 & -5 & -1 \\ 3 & \boxed{-2} & -4 & -2 \\ 5 & -1 & -3 & 0 \end{matrix}$$

Images are graphs

- An image is a grid and a grid is a (nice) graph



Graph Convolutional Network

Graph Convolutional Networks (GCN): Semi-Supervised Classification with Graph Convolutional Networks. Thomas N. Kipf, Max Welling. International Conference on Learning Representations (ICLR), 2017

Model (single layer)

graph structure

node features

$$Z = \sigma(A' F W)$$

layer output

trainable weights

non-linearity e.g., ReLU

Graph Convolutional Network

Graph Convolutional Networks (GCN): Semi-Supervised Classification with Graph Convolutional Networks. Thomas N. Kipf, Max Welling. International Conference on Learning Representations (ICLR), 2017

A (5x5)

0	1	0	0	1
1	0	0	0	1
0	0	0	1	0
0	0	1	0	1
1	1	0	1	0

X

F (5x3)

0.5	0.25	0.25
0.0	0.5	0.5
0.24	0.10	0.66
0.9	0.1	0.0
0.1	0.0	0.9

$$A' = D^{-1/2}(A + I)D^{-1/2}$$

Model (single layer)

graph structure

node features

$$\boldsymbol{Z} = \sigma(\boldsymbol{A'}\,\boldsymbol{F}\,\boldsymbol{W})$$

layer output

trainable weights

non-linearity e.g., ReLU

GCN on Cora

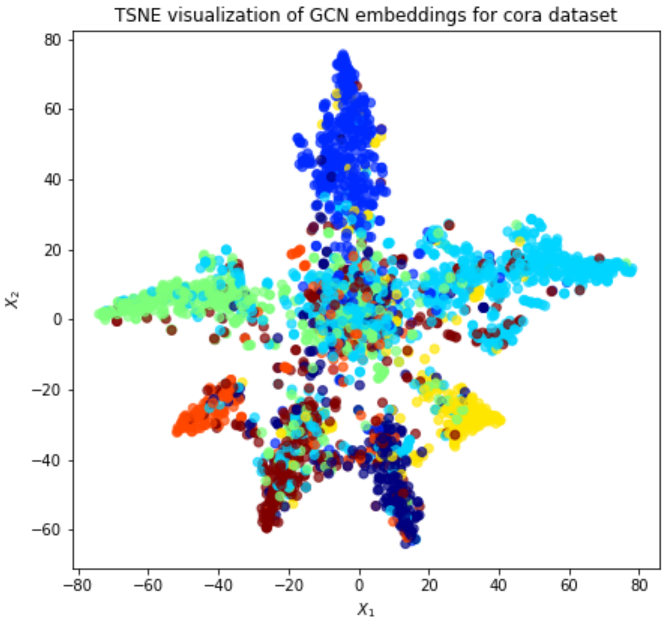
- Let’s predict node labels using GCN

Exercise

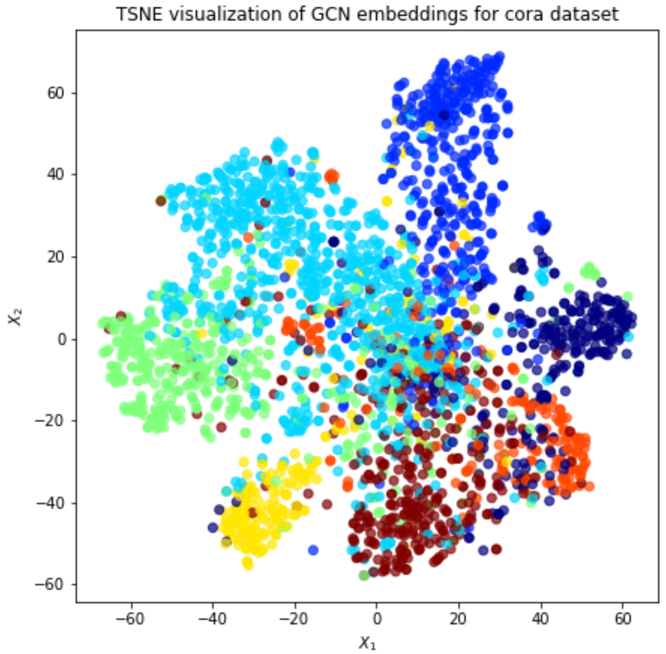
- Can you do better?
- Different number of layers?
- Different size layers?
- What if we add a dense layer at the end?
- What about GCN with no node features?

GCN: Example

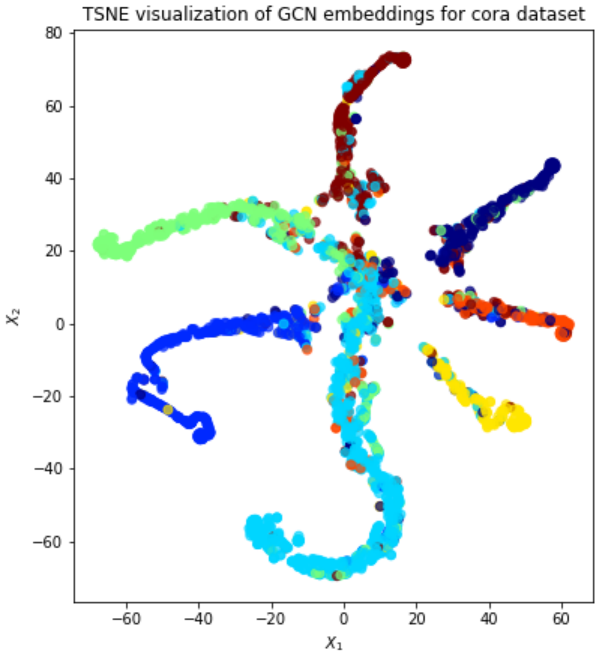
- Attribute inference on Cora, with and without attributes



GCN no node attributes
accuracy ~ 0.65
Output of last GCN layer



GCN + node attributes
accuracy ~ 0.76,
Output of first GCN layer



GCN + node attributes
accuracy ~ 0.76
Output of last GCN layer

GCN Limitations

- GCN does not scale well
- Full batch
  - Memory requirements grow linearly with dataset size
- Undirected graphs
- Homogeneous graphs
- Transductive only
  - What if we add just a single new node to the graph?
  - What if we want to train on one graph and make predictions with another graph?
- Can we have GCN power without the above limitations?

GraphSAGE model

- Use sampling and aggregation over node neighbors!

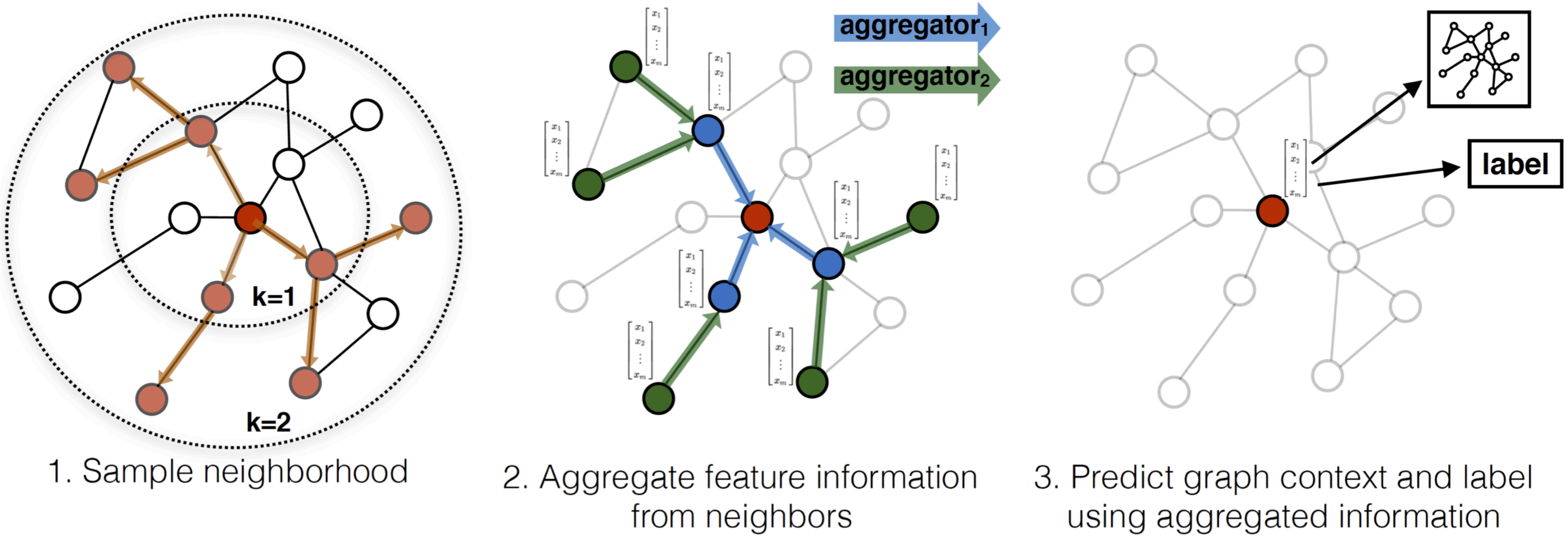
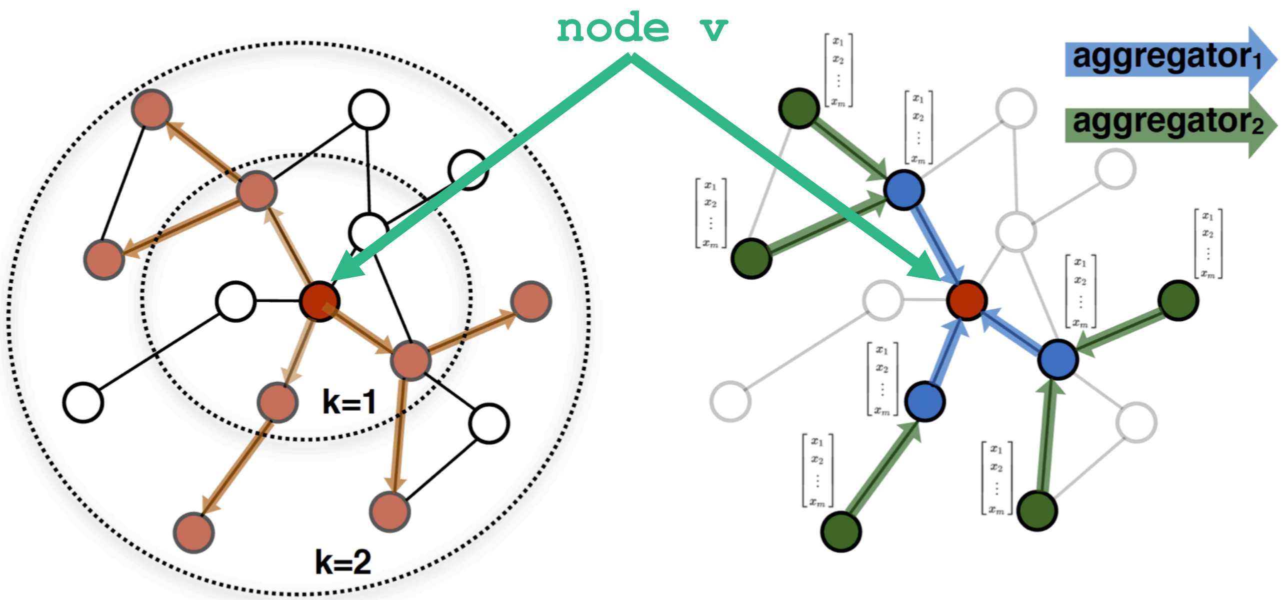


Image from http://snap.stanford.edu/graphsage/

GraphSAGE model

- Use sampling and aggregation over node neighbors!



1. Sample neighborhood

2. Aggregate feature information from neighbors

Model (one hop neighborhood)

$$\overrightarrow{f_{N_i}} = AGGREGATE_{j \in N_i}(\overrightarrow{f_j})$$

graph structure

node features

layer output

$$\overrightarrow{z_i} = \sigma(\boldsymbol{W} \times CONCAT(\overrightarrow{f_i}, \overrightarrow{f_{N_i}}))$$

non-linearity e.g., ReLU

trainable weights

Image from http://snap.stanford.edu/graphsage/

GraphSAGE on Cora (Node Classification)

- Let's predict node labels using GraphSAGE

Exercise

- Can you improve the results?
- Different number of neighbors per depth
- Deeper network, sample more than K=2 hops away
- Wider network
- Different aggregator functions?
  - https://stellargraph.readthedocs.io/en/stable/api.html#module-stellargraph.layer.graphsage

GraphSAGE on Cora (Link Prediction)

- Let's try link prediction with GraphSAGE

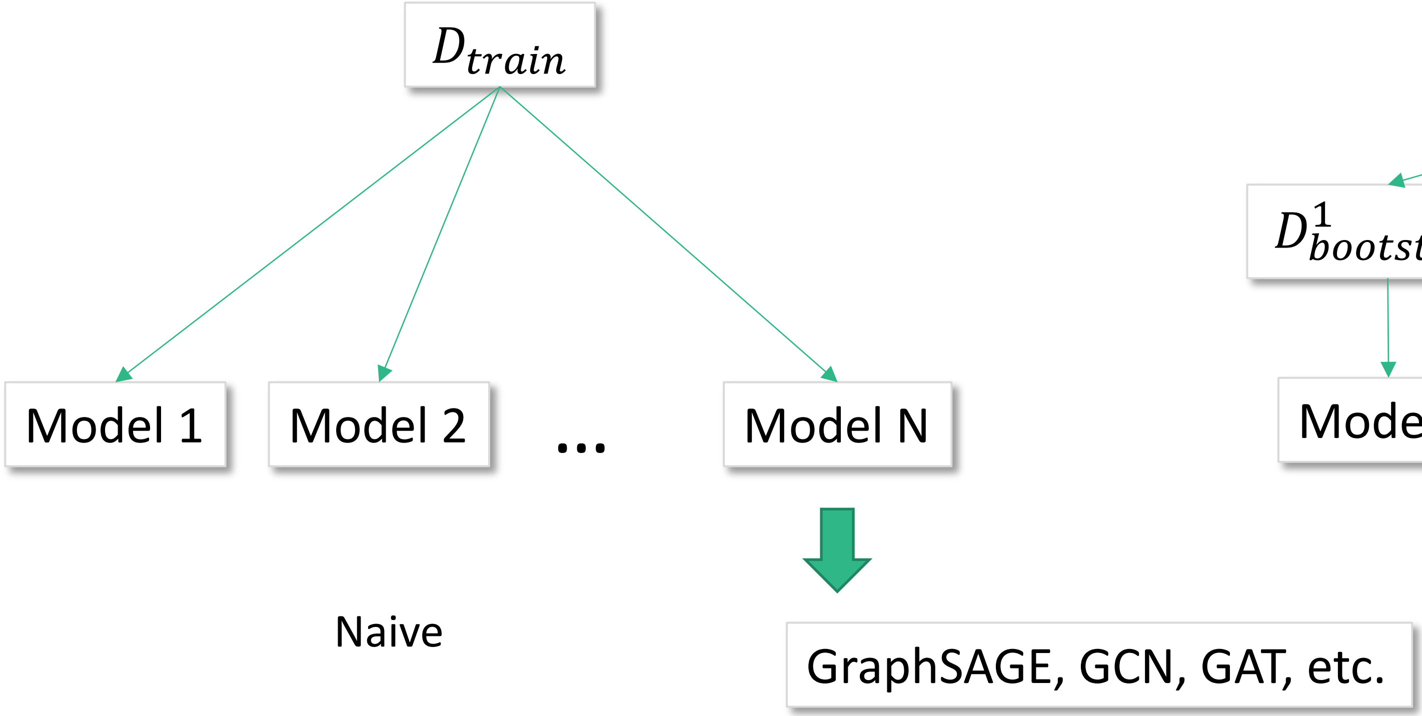
Model Calibration

- Consider the output of a classification model
- What does this output represent?
- Can we trust it?

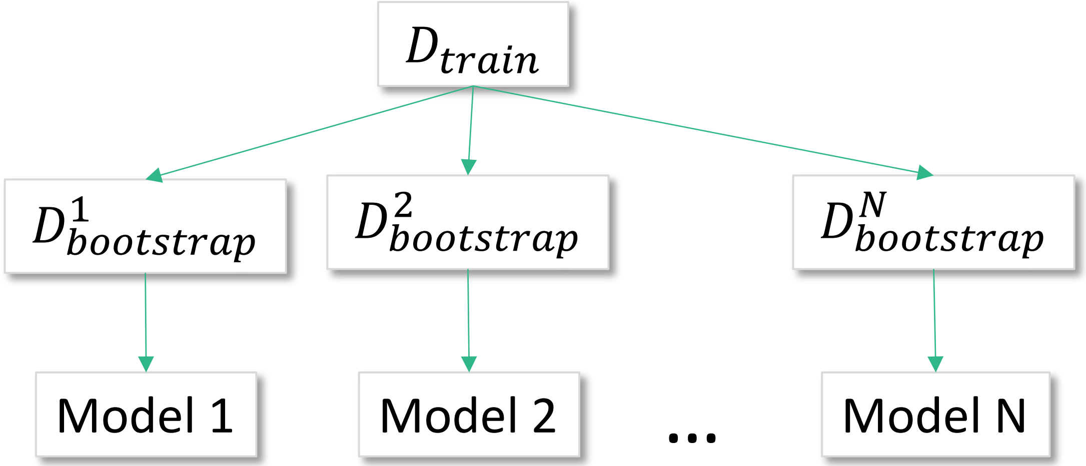
GNN Ensembles

Ensembles (Training)

- Need diversity
  - Randomized weight initialization (Naive)
  - Sampling of training data (Bagging)

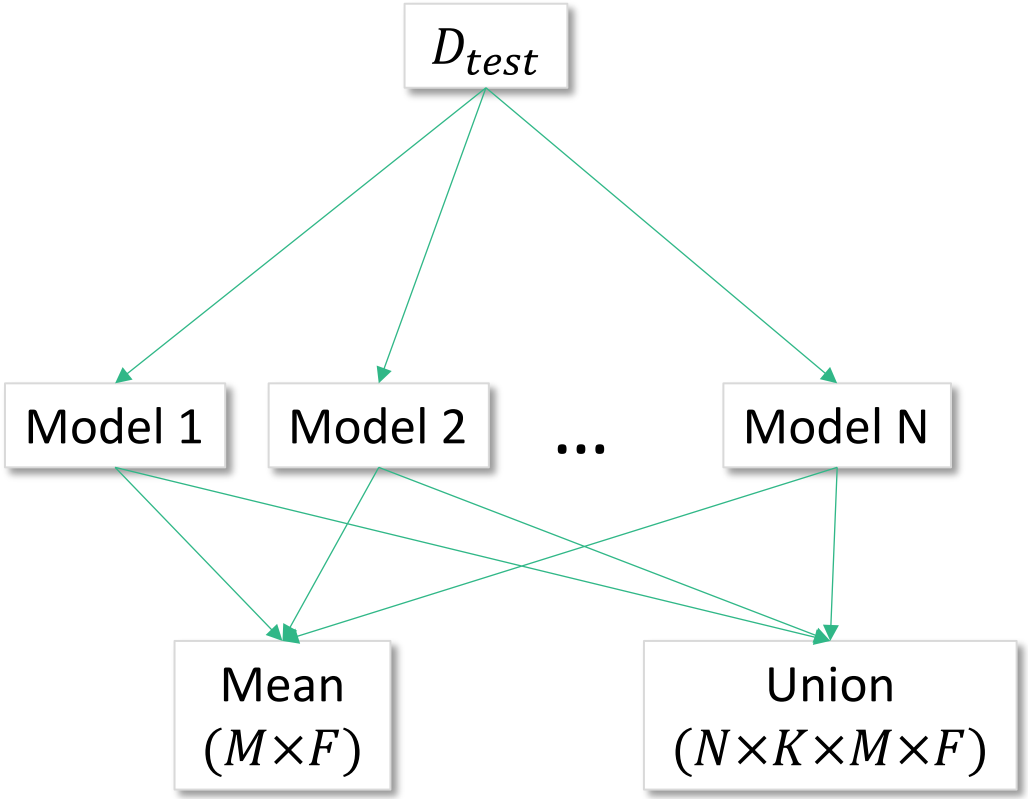


Naive



Bagging

Ensembles (Predicting)



N: Number of models
K: Number of predictions per query point
M: Number of query points
F: Output dimensionality

Ensembles for classification

But There is More...

Link Prediction
GraphSAGE
Ensembles
Metapath2Vec
Random Walks Demos
Calibration
GAT HinSAGE
SGC Node2Vec
Unsupervised Methods
Probabilistic Predictions
Node Classification

https://github.com/stellargraph/stellargraph

STELLAR GRAPH

CSIRO DATA 61

Other Topics

- Weighted Random Walks
- Graph Attention Networks
  - GAT
  - GraphSAGE with GAT aggregator
- Unsupervised convolutional neural network methods
  - GraphSAGE
- Heterogeneous Networks
  - Metapath2Vec
  - HinSAGE
- Model Calibration
- Model Ensembles

Other Approaches

- **FastGCN**: *Fast learning with Graph Convolutional Networks via importance sampling*, J. Chen, T. Ma, and C. Xiao, ICLR 2018
- **Metapath2Vec**: *Scalable representation learning for heterogeneous networks*, Y. Dong, N. V. Chawla, and A. Swami, KDD 2017
- **GraphSAGE**: *Inductive Representation Learning on Large Graphs, W. Hamilton*, R. Ying, and J. Leskovec, NIPS 2017
- **GAT**: *Graph Attention Networks*, P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, ICLR 2018
- **Decagon**: *Modeling polypharmacy side effects with Graph Convolutional Networks*, M. Zitnik, M. Agrawal, and J. Leskovec

References

1. *Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering*, M. Belkin and P.Niyogi, NIPS 2002
2. *Distributed Representations of Words and Phrases and their Compositionality*, T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, NIPS 2013
3. *DeepWalk: Online Learning of Social Representations*, B. Perozzi, R. Al-Rfou, and S. Skiena, KDD 2014
4. *Node2Vec: Scalable Feature Learning for Networks*, A. Grover and J. Leskovec, KDD 2016
5. *Semi-Supervised Classification with Graph Convolutional Networks*, T. N. Kipf and M. Welling, ICLR 2017
6. *Inductive Representation Learning on Large Graphs*, W. L. Hamilton, R. Ying, and J. Leskovec, NIPS 2017
7. *Representation Learning on Graphs: Methods and Applications*, W. L. Hamilton, R. Ying, and J. Leskovec, arXiv preprint arXiv:1709.05584 2017

Open Source Implementations

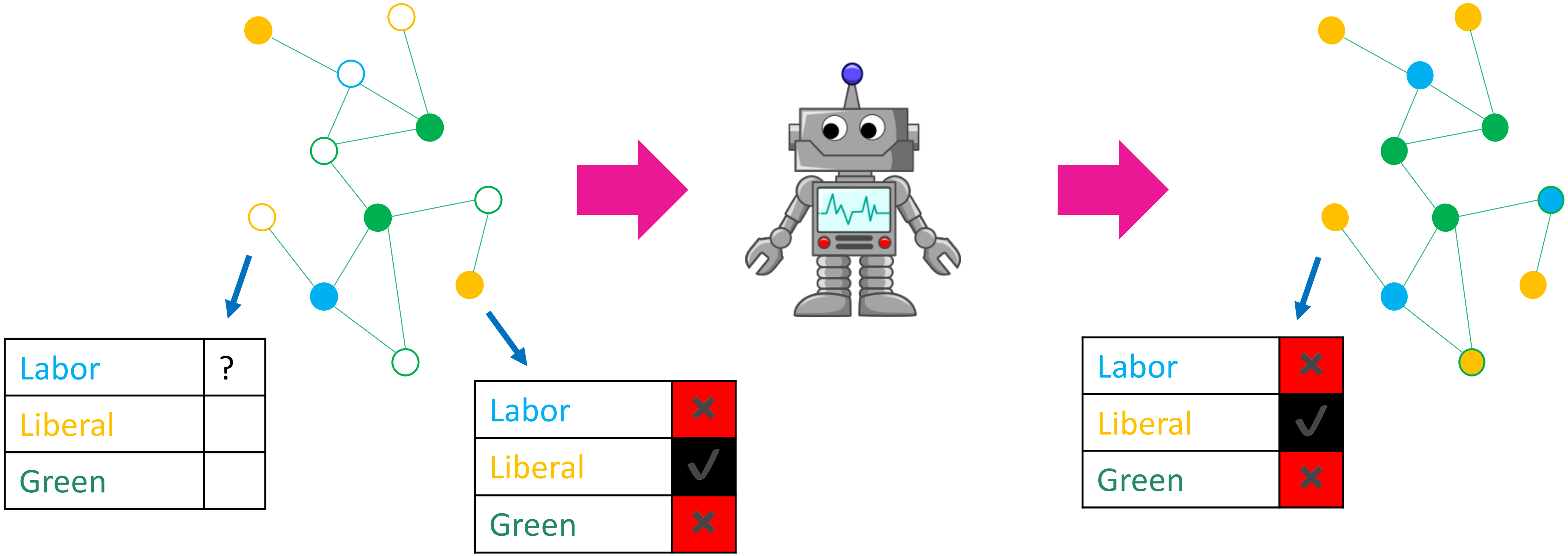
- **StellarGraph:** https://github.com/stellargraph/stellargraph
  - Implementations for DeepWalk, Node2Vec, Metapath2Vec, GCN, GAT, GraphSAGE, and HINSAGE.
- DeepWalk: https://github.com/phanein/deepwalk
- Node2Vec: http://snap.stanford.edu/node2vec/
- Metapath2Vec: https://ericdongyx.github.io/metapath2vec/m2v.html
- GCN: https://github.com/tkipf/gcn
- FastGCN: https://github.com/matenure/FastGCN
- GraphSAGE: http://snap.stanford.edu/graphsage/
- GAT: https://github.com/topics/graph-attention-networks

Thank you

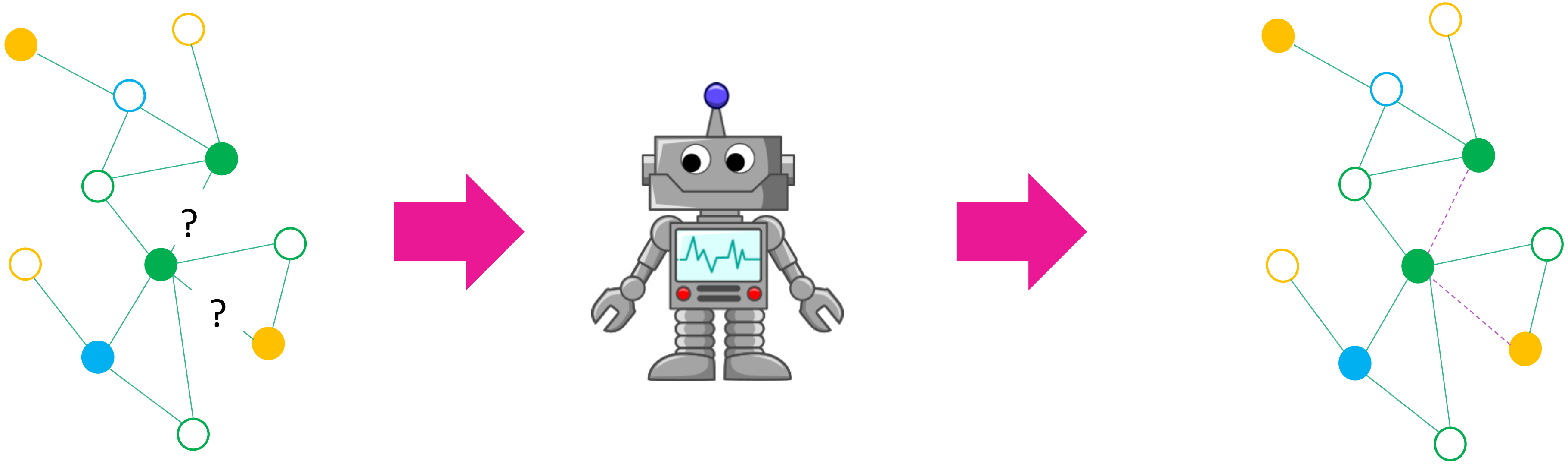
Engineering and Design
Pantelis Elinas/Anna Leontjeva
Senior Research Engineer/Senior Data Scientist

e pantelis.elinas@data61.csiro.au/anna.leontjeva@data61.csiro.au

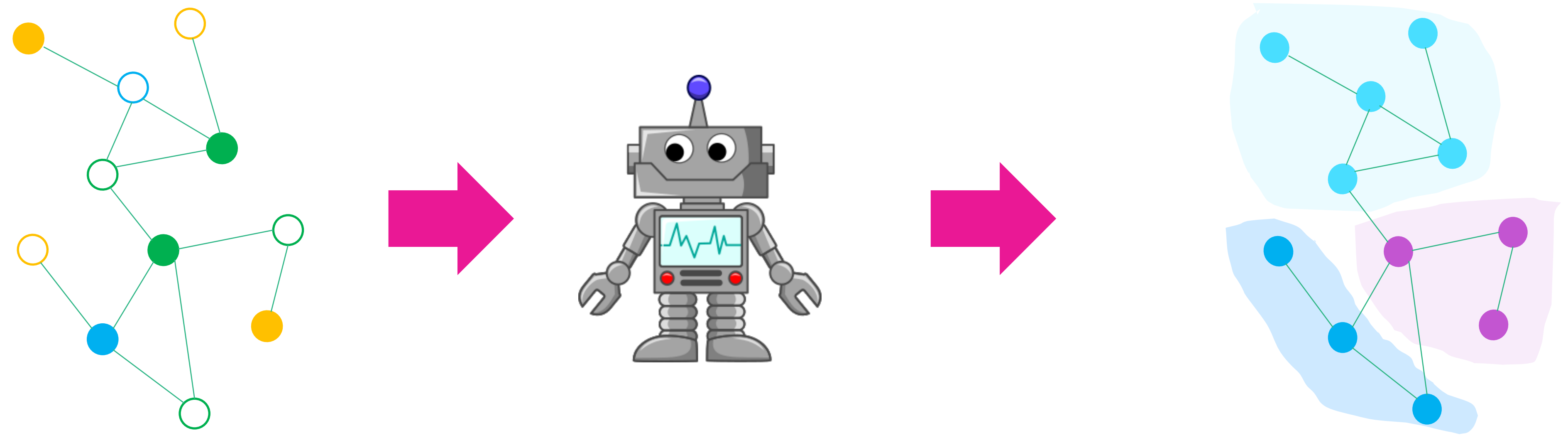
Graph ML Problem: Node Classification



Graph ML Problem: Link Prediction

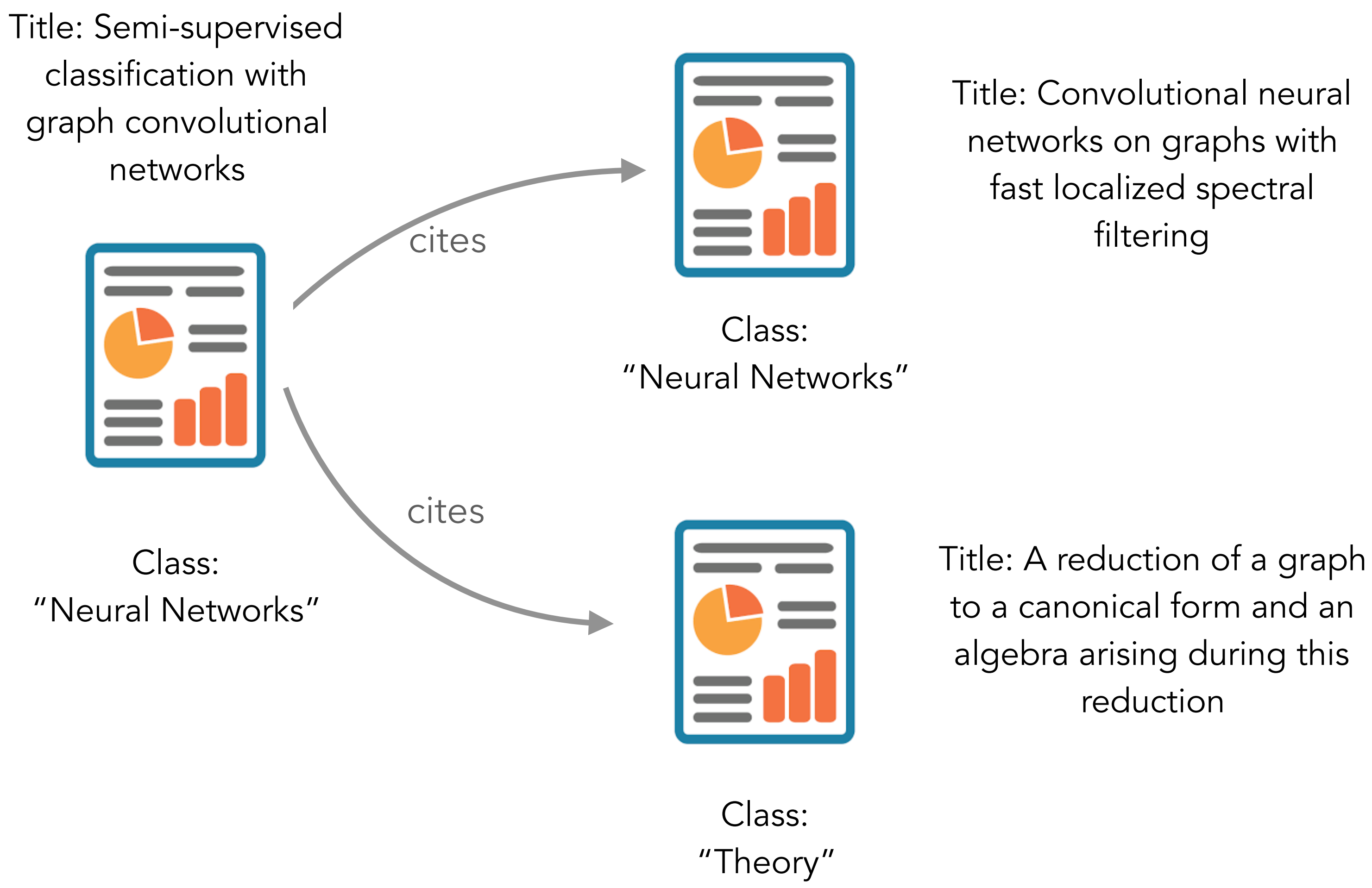


Graph ML Problem: Community Detection



Datasets

CORA citation network



CORA node features

ABSTRACT

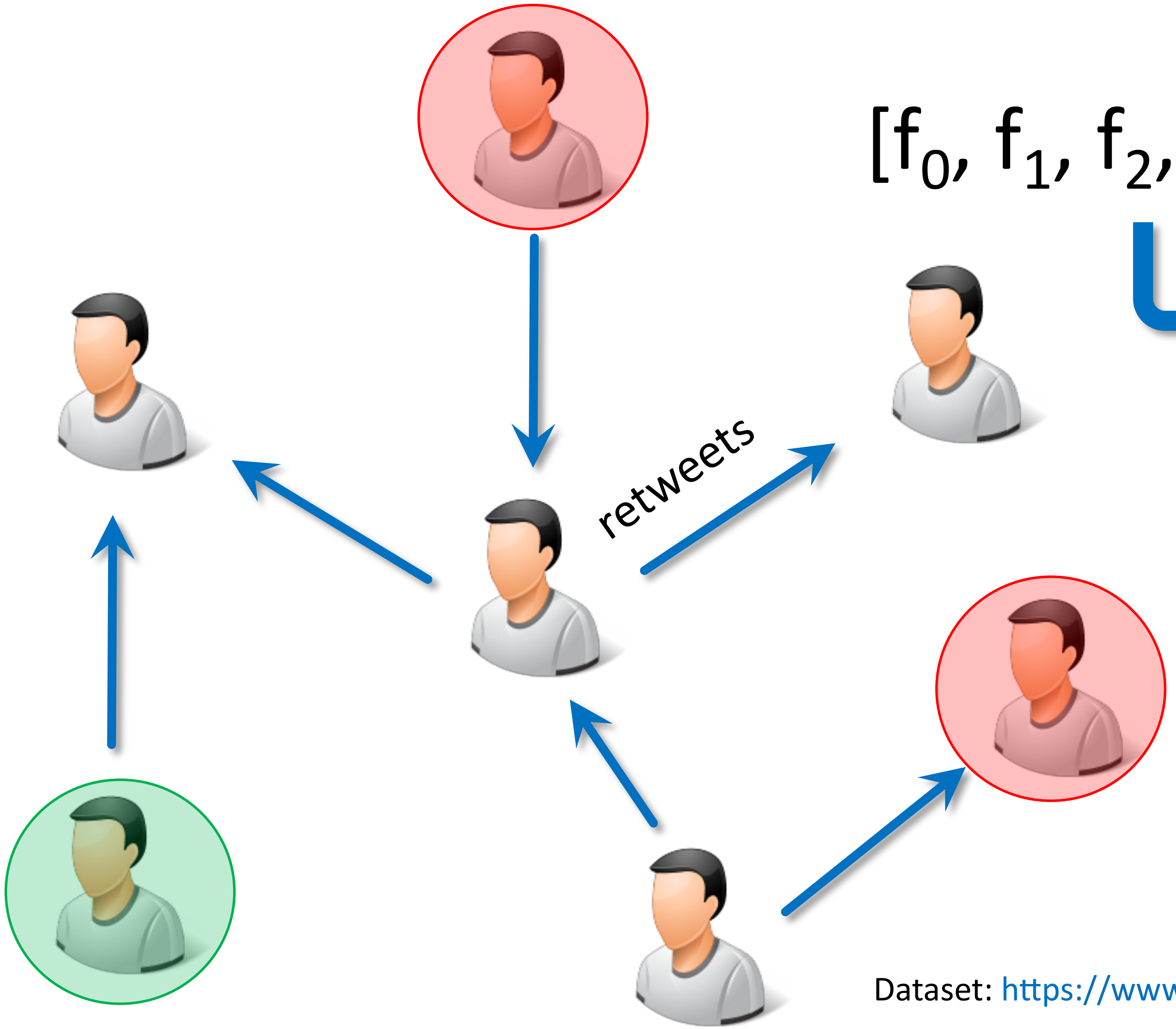
We present a scalable approach for semi-supervised learning on graph-structured data that is based on an efficient variant of convolutional neural networks which operate directly on graphs. We motivate the choice of our convolutional architecture via a localized first-order approximation of spectral graph convolutions. Our model scales linearly in the number of graph edges and learns hidden layer representations that encode both local graph structure and features of nodes. In a number of experiments on citation networks and on a knowledge graph dataset we demonstrate that our approach outperforms related methods by a significant margin.

Title: Semi-supervised classification with graph convolutional networks

Class: "Neural Networks"

	w_1: learning	w_2: genetic	w_3: model	…
Semi-supervised classification ..	1	0	1	…
Convolutional neural networks…	1	0	0	…
…				…

Hateful twitter users network



$[f_0, f_1, f_2, \ldots, f_{204}]$

Features, f_i, are derived from user tweets and activity

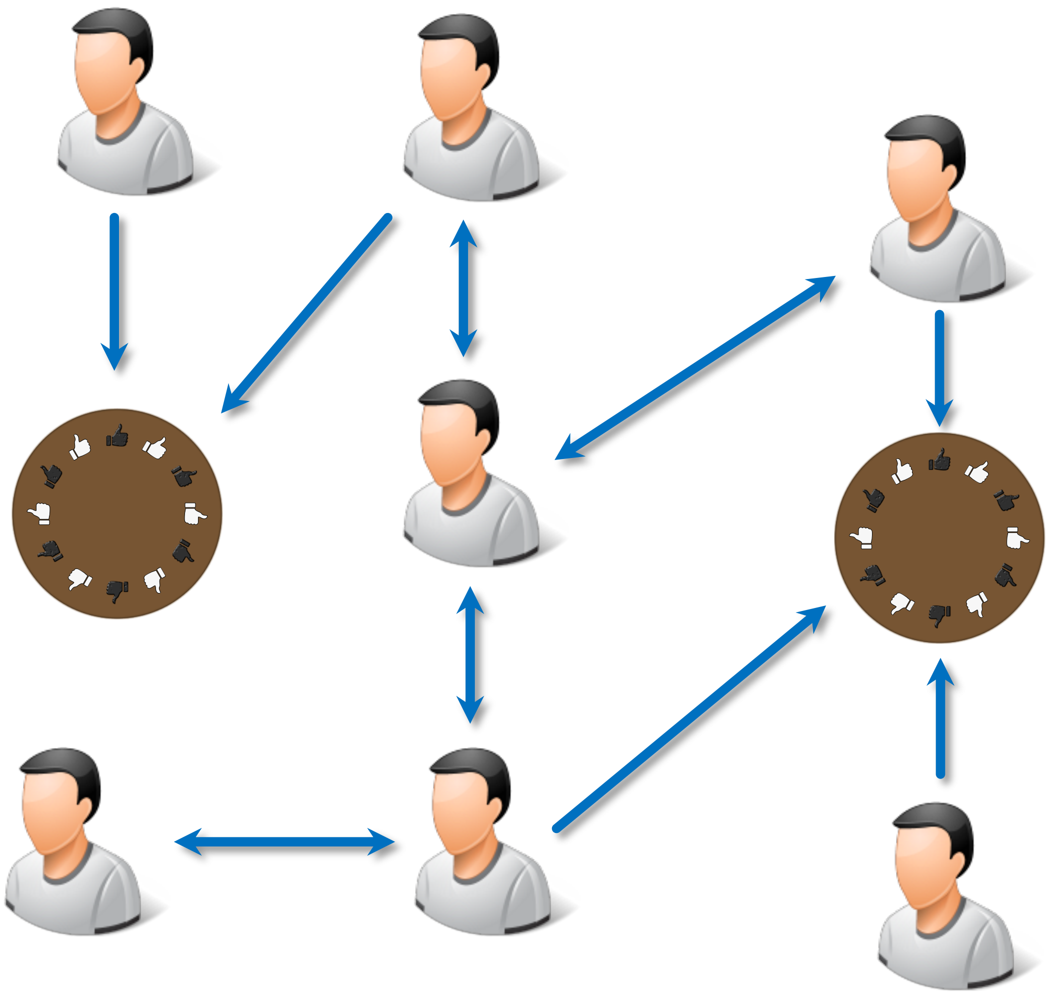
Users: 100K

Edges: 2200K

Annotated Users: 5K

Dataset: https://www.kaggle.com/manoelribeiro/hateful-users-on-twitter/home

BlogCatalog



- Heterogeneous network
- No attributes
- Statistics
  - Bloggers: 10,312
  - Groups: 39
  - Edges: ~333K
- Edges are between bloggers and between bloggers and groups

Traditional ML workflow

1 → 2 → 3

1 Prepare Data

2 Specify and Train Model

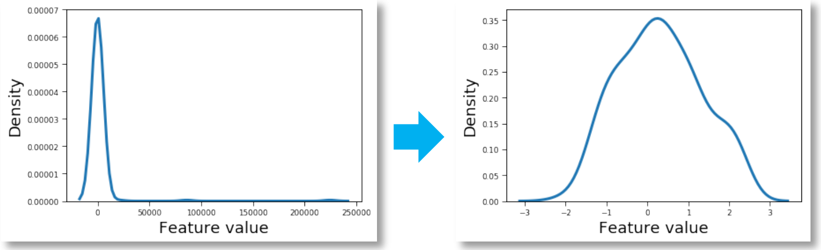
3 Evaluate Model and Visualize Results

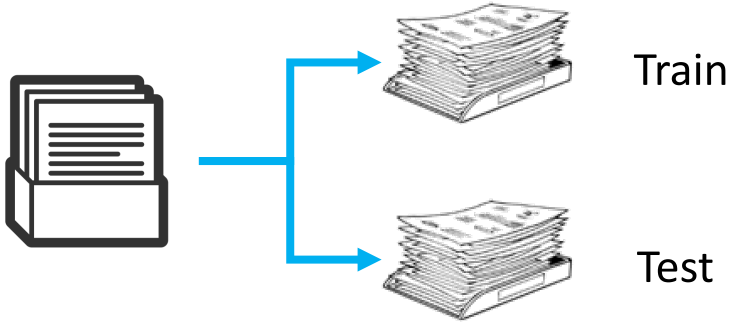
pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

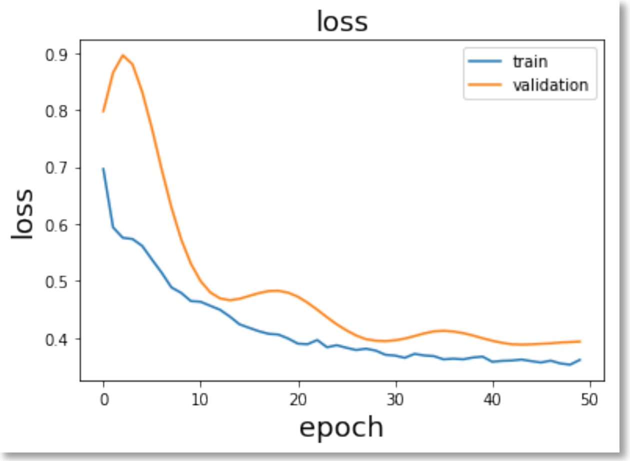
scikit learn

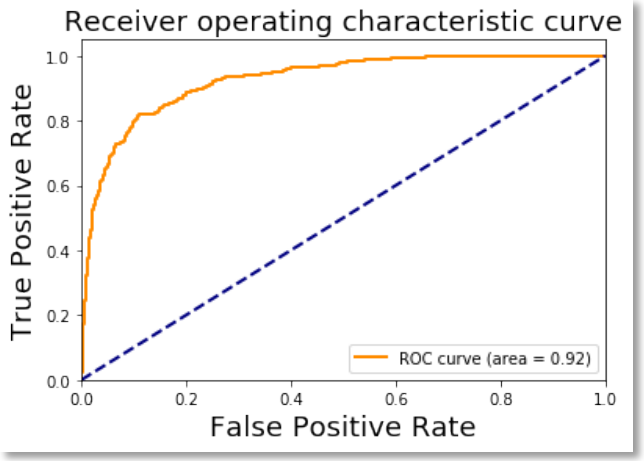
scikit learn

matplotlib



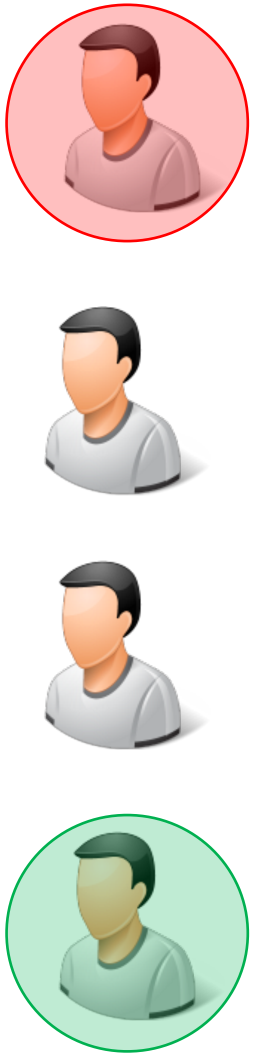






CSIRO DATA 61

Traditional ML for Twitter dataset

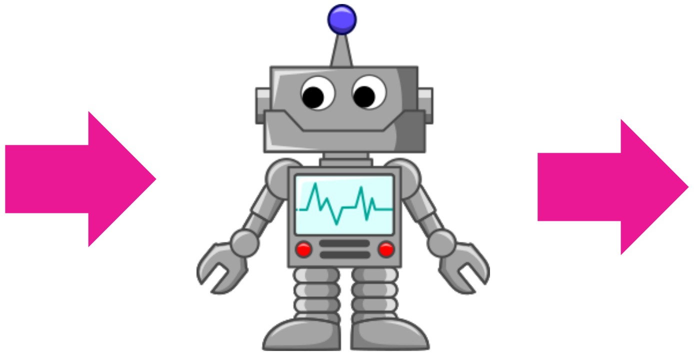


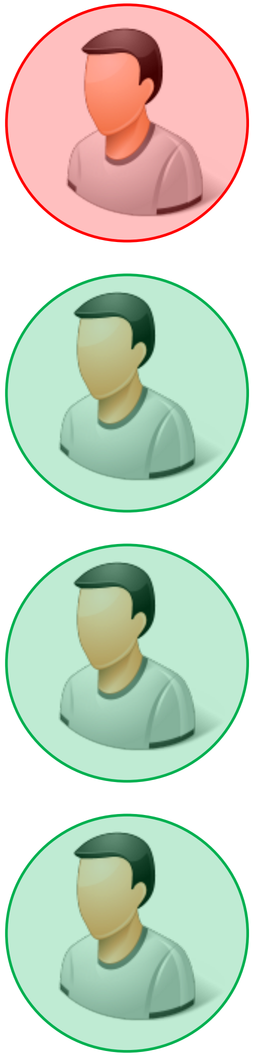
$[f_0, f_1, f_2, \ldots, f_{204}]$

$[f_0, f_1, f_2, \ldots, f_{204}]$

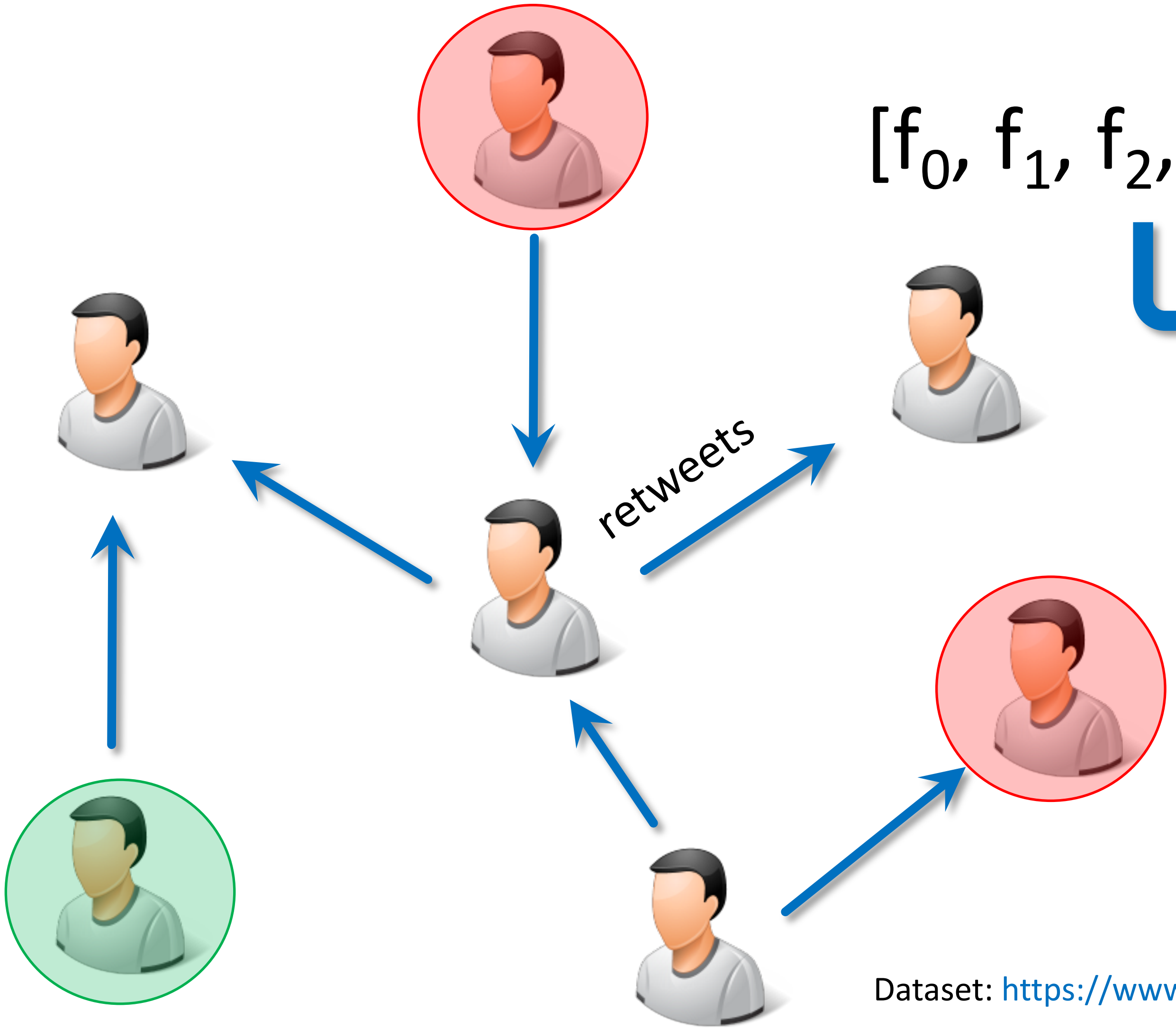
$[f_0, f_1, f_2, \ldots, f_{204}]$

$[f_0, f_1, f_2, \ldots, f_{204}]$





Hateful twitter users network



$[f_0, f_1, f_2, \ldots, f_{204}]$

Features, f_i, are derived from user tweets and activity

Users: 100K

Edges: 2200K

Annotated Users: 5K

Dataset: https://www.kaggle.com/manoelribeiro/hateful-users-on-twitter/home

StellarGraph

- Python Graph ML library
- Open source Apache 2.0 License

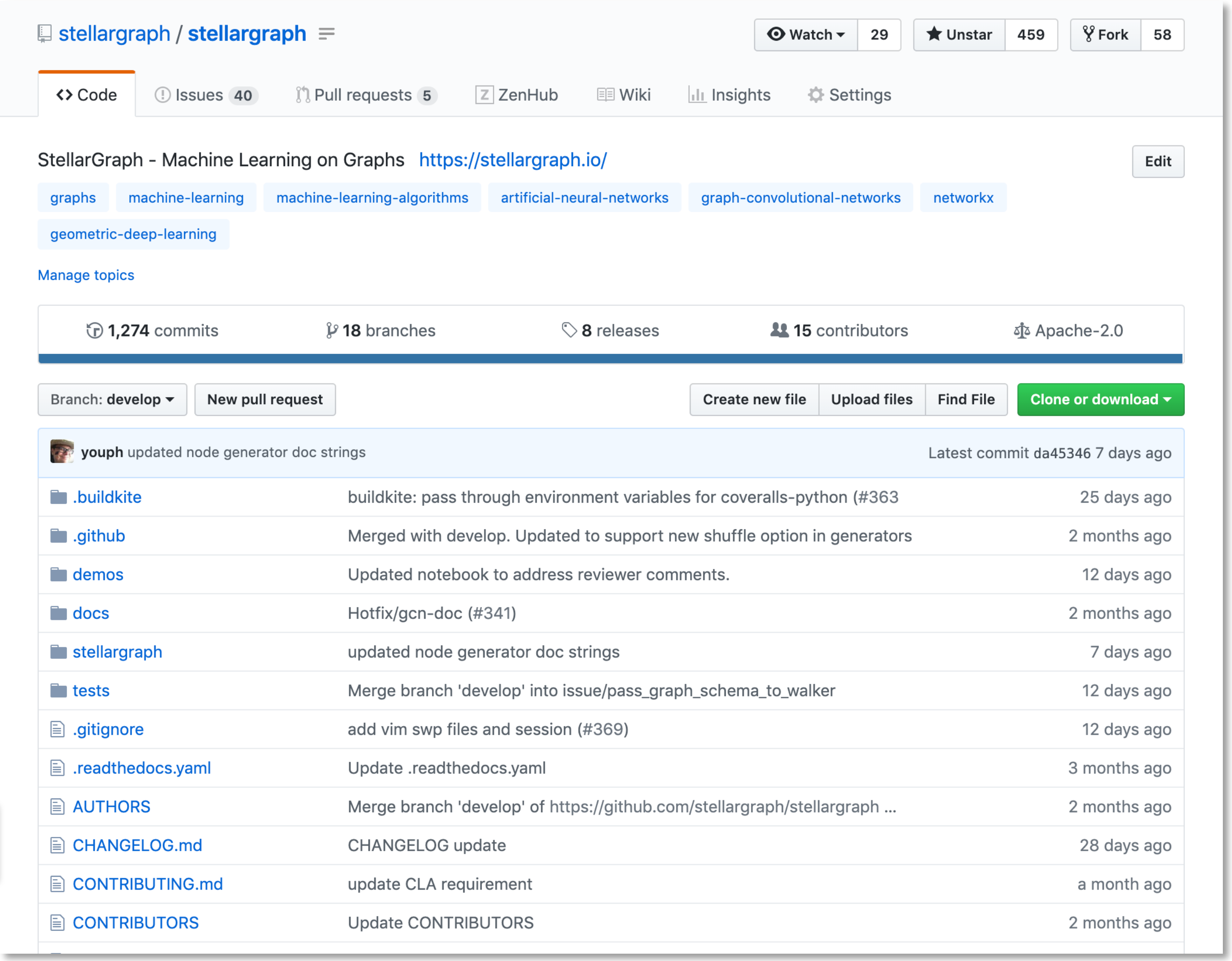
STELLAR
GRAPH

NetworkX

https://github.com/stellargraph/stellargraph

https://stellargraph.readthedocs.io/en/stable

https://community.stellargraph.io



Graph ML workflow

1 → 2 → 3

1 Prepare Data

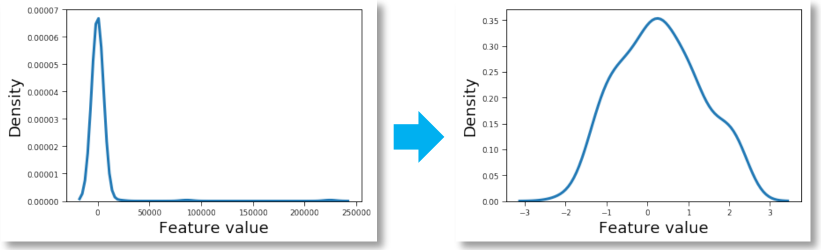
2 Specify and Train Model

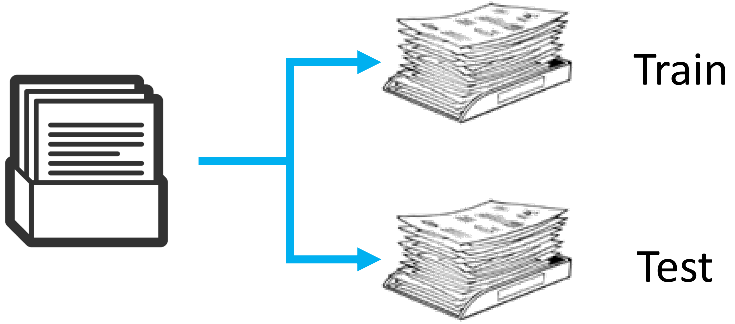
3 Evaluate Model and Visualize Results

pandas

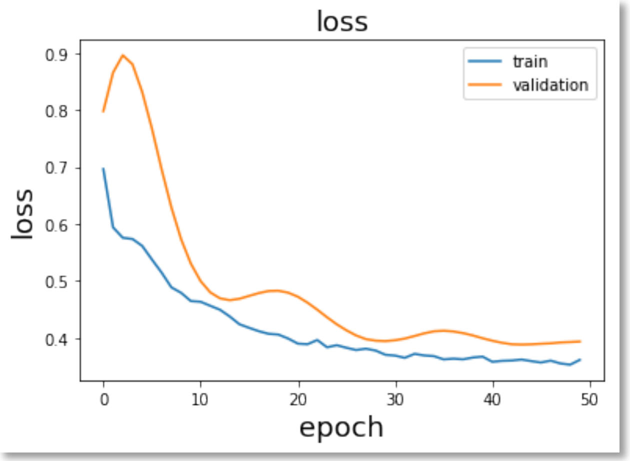
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

scikit learn

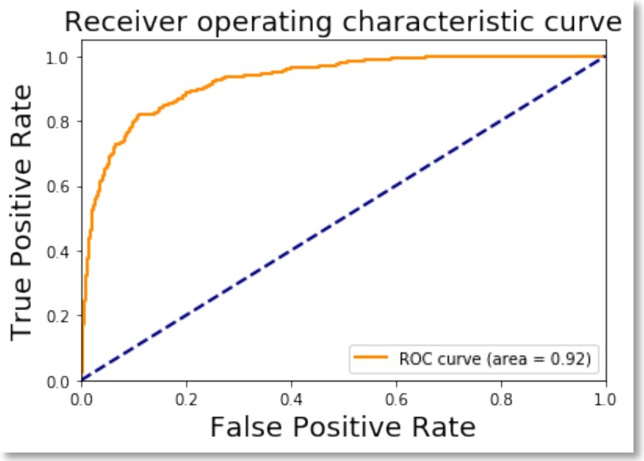




STELLAR
GRAPH



matplotlib



CSIRO DATA 61