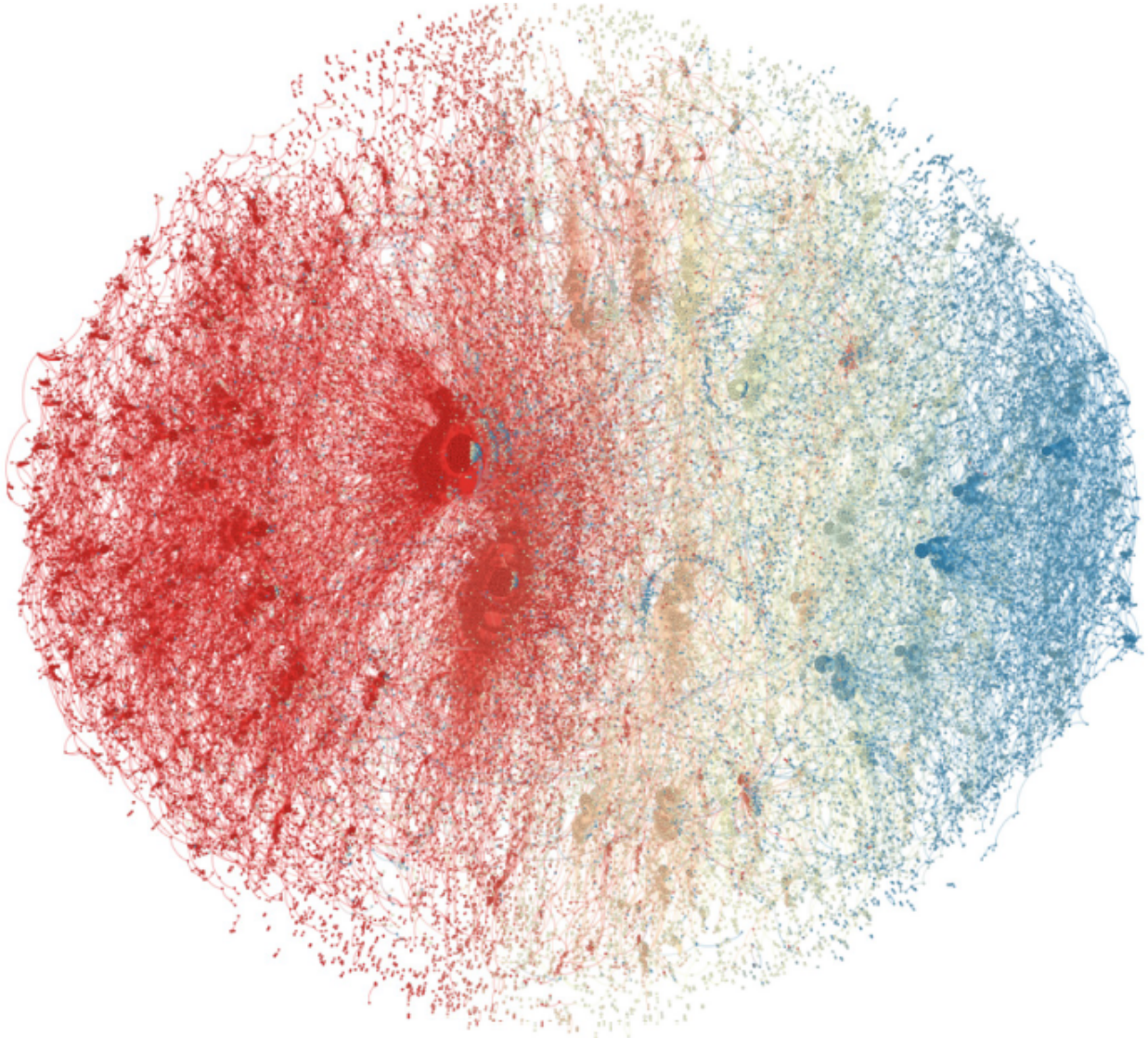
Graph Representation Learning

Jure Leskovec

CHAN ZUCKERBERG
BIOHUB

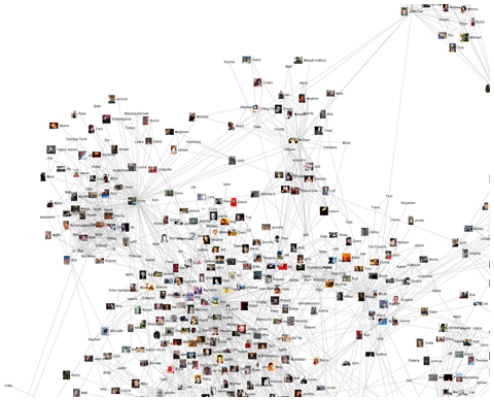
Networks



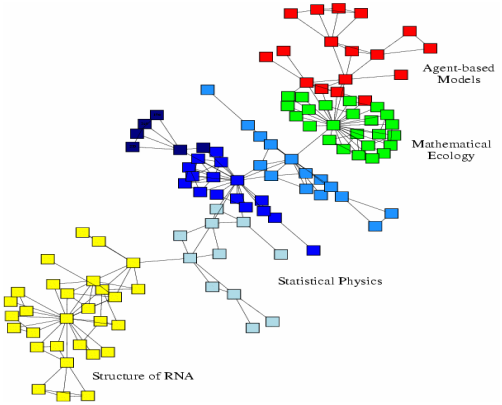
Why Networks?

- Universal language for describing complex data
  - Networks from science, nature, and technology are more similar than one would expect
- Shared vocabulary between fields
  - Computer Science, Social science, Physics, Statistics, Biology
- Data availability (+computational challenges)
  - Web/mobile, bio, health, and medical
- Impact!
  - Social networking, Social media, Drug design

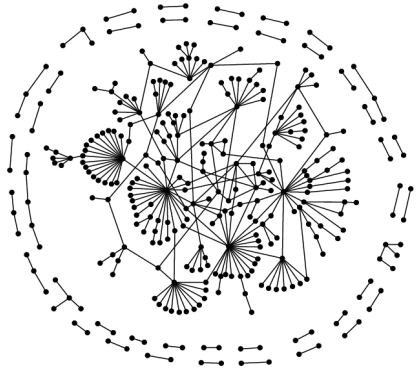
Many Data are Networks



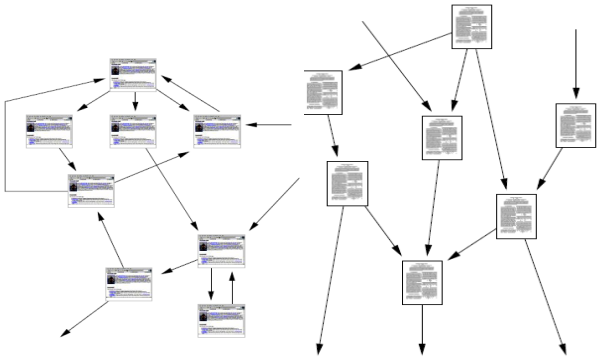
Social networks



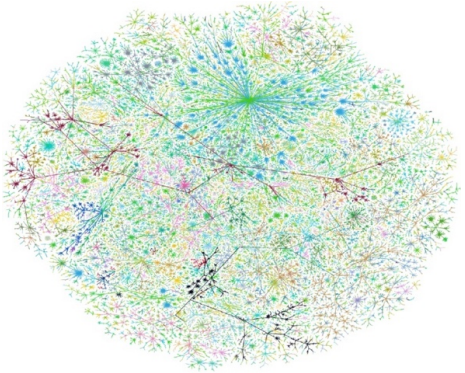
Economic networks



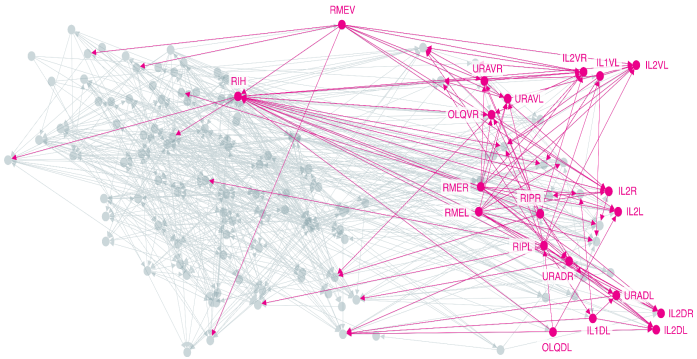
Biomedical networks



Information networks:
Web & citations

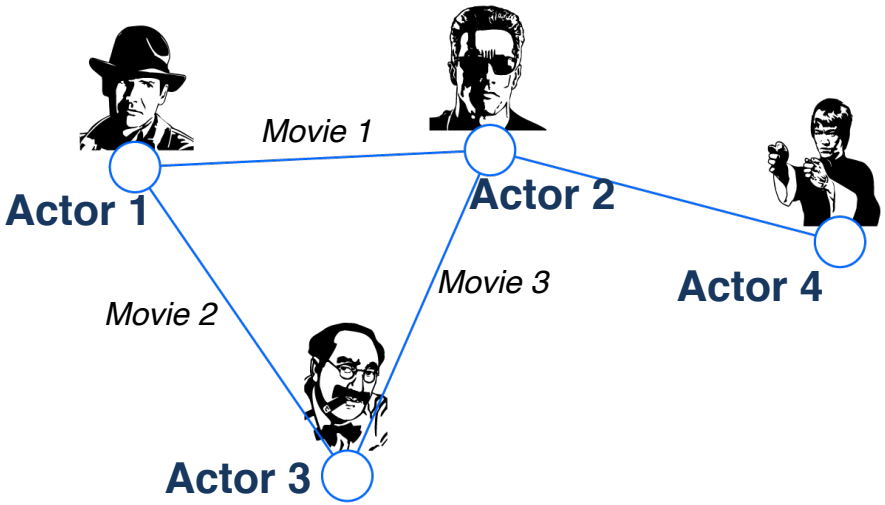


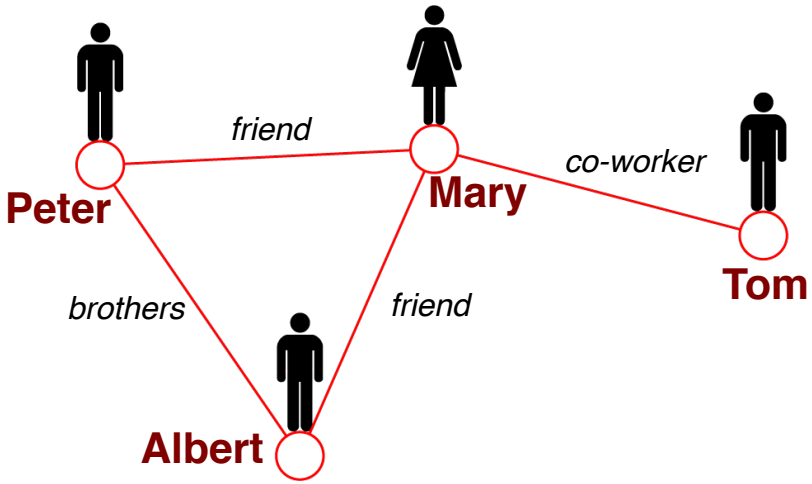
Internet

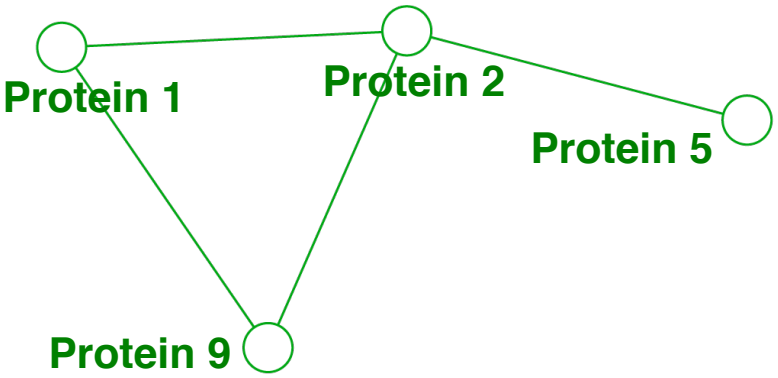


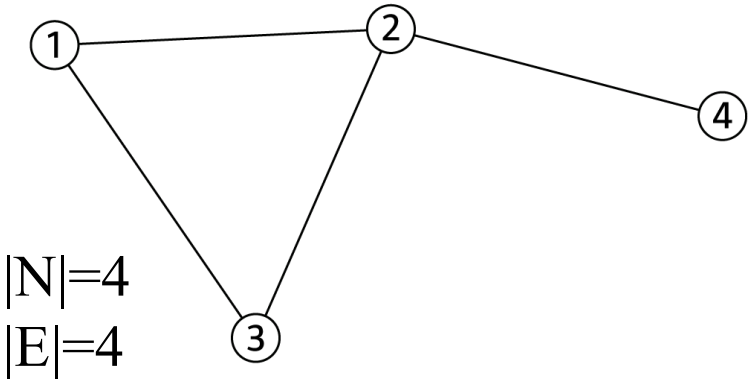
Networks of neurons

Networks: Common Language







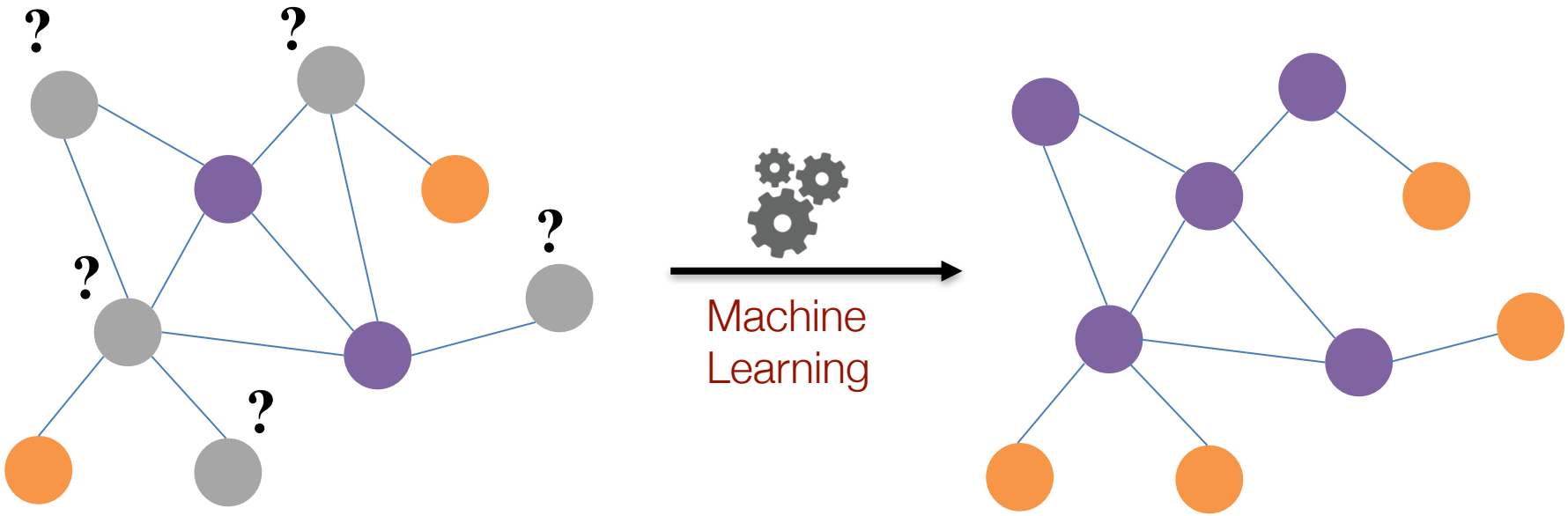


Tasks on Networks

Classical ML tasks in networks:

- Node classification
  - Predict a type of a given node
- Link prediction
  - Predict whether two nodes are linked
- Community detection
  - Identify densely linked clusters of nodes
- Network similarity
  - How similar are two (sub)networks

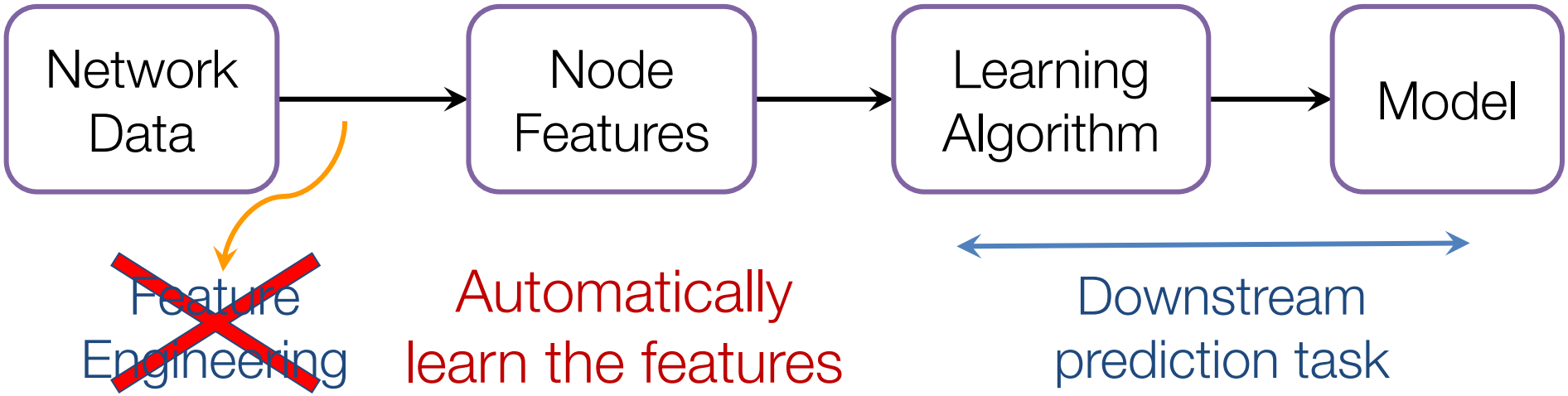
Example: Node Classification



Many possible ways to create node features:

- Node degree, PageRank score, motifs, …
- Degree of neighbors, PageRank of neighbors, …

Machine Learning Lifecycle

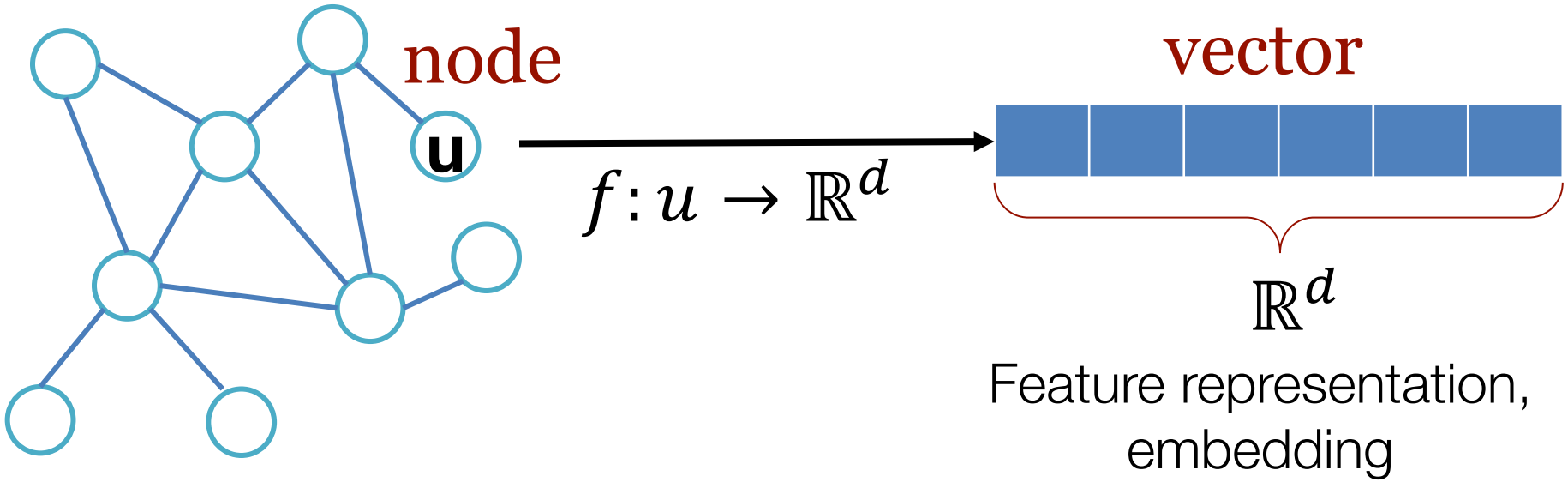


(Supervised) Machine Learning Lifecycle: This feature, that feature.

Every single time!

Feature Learning in Graphs

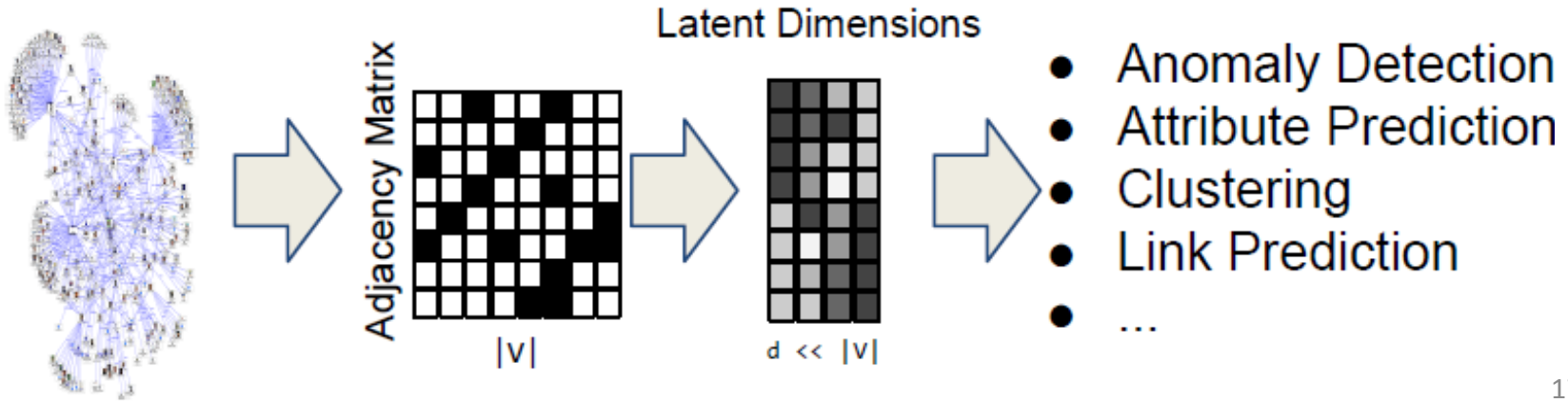
This talk: Feature learning for networks!



Why Learn Embeddings?

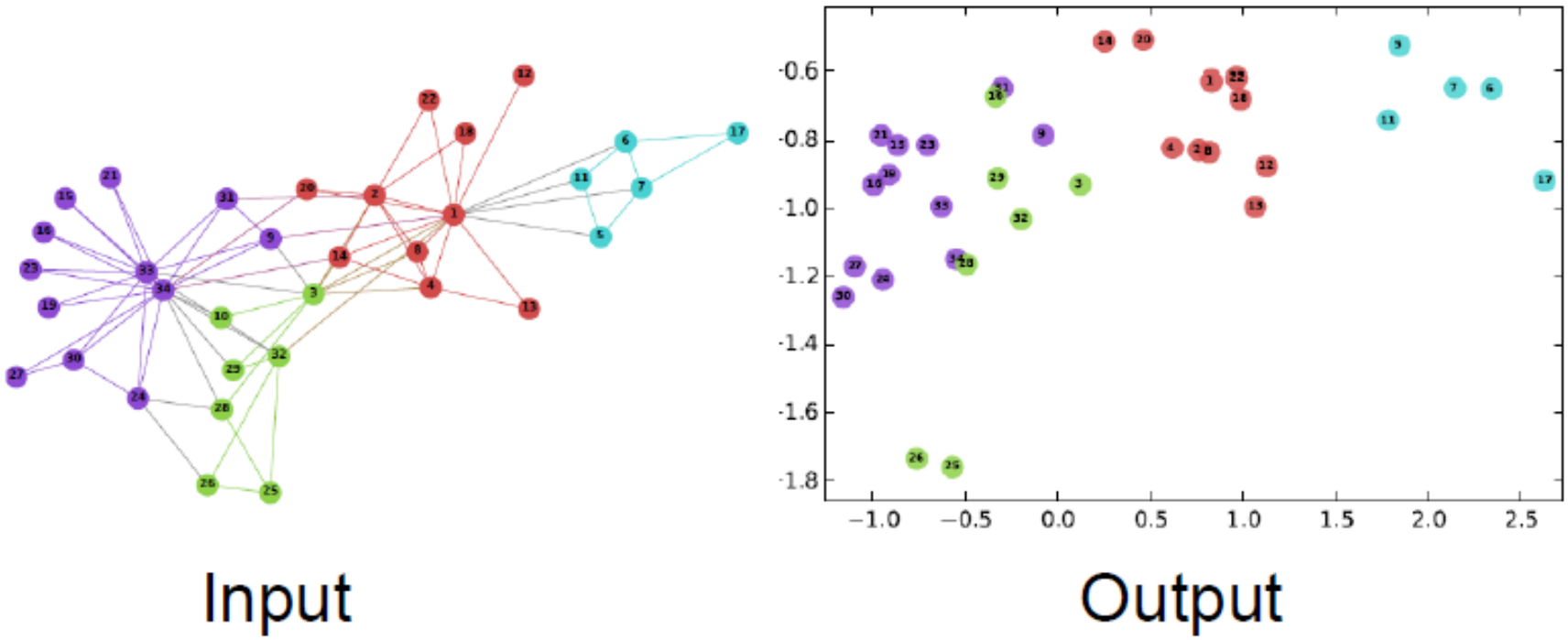
The goal is to map each node into a low-dimensional space

- Distributed representation for nodes
- Similarity between nodes indicates link strength
- Encodes network information and generate node representation



Example

- Zachary's Karate Club network:

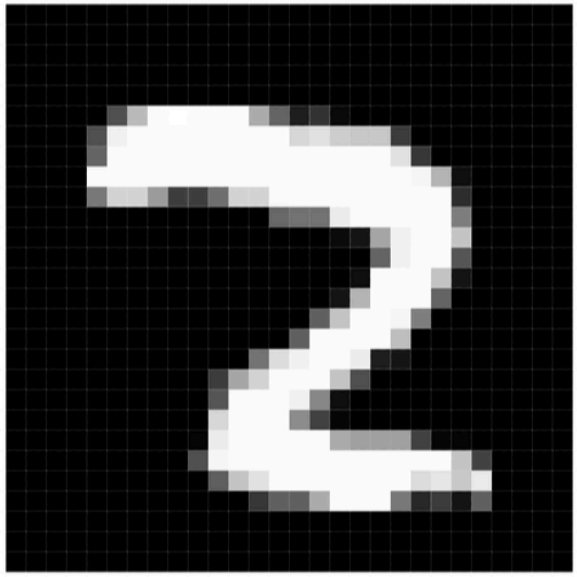


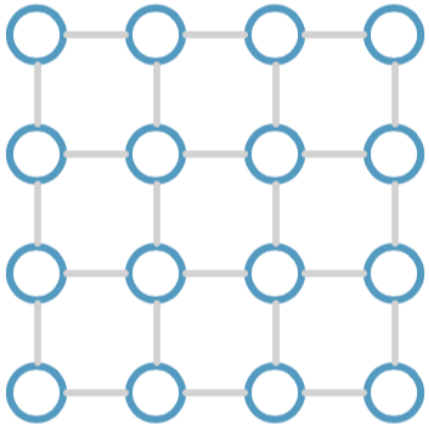
Input

Output

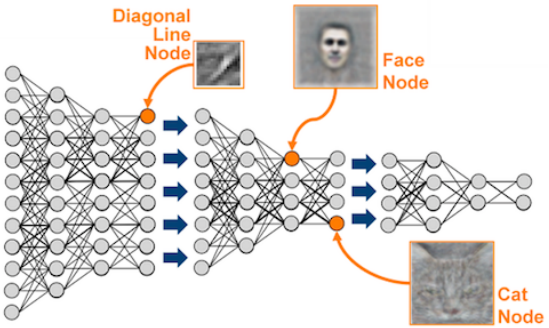
[image by DeepWalk]

Why Is It Hard?





2D grid



Images have fixed 2D structure

- Can define convolutions (CNNs)

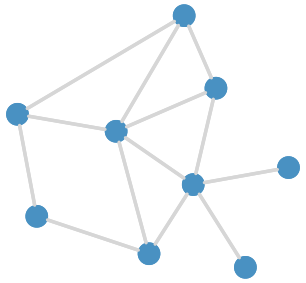
Why Is It Hard?

Text and Speech have linear 1D structure

- Can define sliding windows

But graphs are non-Euclidean!

- Node numbering is arbitrary (node isomorphism problem)
- Much more complex structure



This Talk: Outline

Feature Learning for networks:

1) "Linearizing" the graph

- Create a "sentence" for each node using random walks
  - node2vec

2) Graph convolution networks

- Propagate information between the nodes of the graph
  - GraphSAGE

node2vec: Unsupervised Feature Learning

node2vec: Scalable Feature Learning for Networks
A. Grover, J. Leskovec. KDD 2016.
Predicting multicellular function through multi-layer tissue networks.
M. Zitnik, J. Leskovec. Bioinformatics, 33 (14): i190-i198, 2017.

Unsupervised Feature Learning

- Intuition: Find embedding of nodes to d-dimensions that preserves similarity
- Idea: Learn node embedding such that nearby nodes are close together
- Given a node u, how do we define nearby nodes?
  - $N_S(u)$ … neighbourhood of u obtained by some strategy S

Feature Learning as Optimization

- Given $G = (V, E)$
- Goal is to learn $f: u \rightarrow \mathbb{R}^d$
  - where f is a table lookup
    - We directly "learn" coordinates $f(u)$ of u
- Given node u, we want to learn feature representation $f(u)$ that is predictive of nodes in u's neighborhood $N_S(u)$

$$\max_f \sum_{u \in V} \log \Pr(N_S(u) | f(u))$$

Unsupervised Feature Learning

Goal: Find embedding $f(u)$ that predicts nearby nodes $N_S(u)$:

$$\max_f \sum_{u \in V} \log Pr(N_S(u)|f(u))$$

Assume conditional likelihood factorizes:

$$Pr(N_S(u)|f(u)) = \prod_{n_i \in N_S(u)} Pr(n_i|f(u))$$

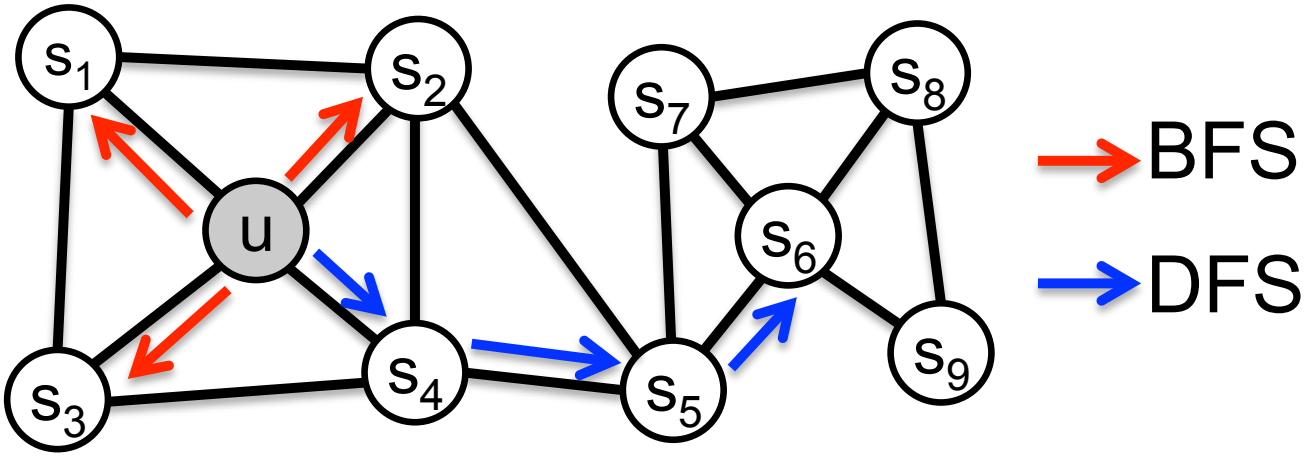
Then softmax:

$$Pr(n_i|f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}$$

Estimate $f(u)$ using stochastic gradient descent.

How to determine $N_S(u)$

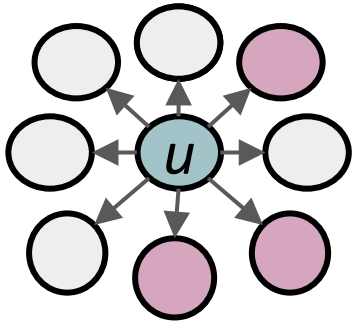
Two classic strategies to define a neighborhood $N_S(u)$ of a given node u:



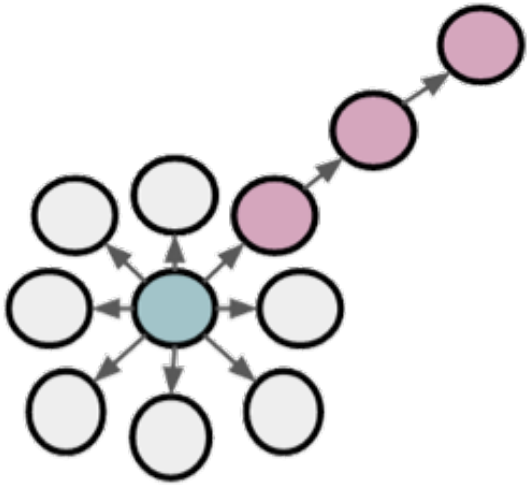
$N_{BFS}(u) = \{s_1, s_2, s_3\}$ Local microscopic view

$N_{DFS}(u) = \{s_4, s_5, s_6\}$ Global macroscopic view

BFS vs. DFS



BFS:
Micro-view of neighbourhood



DFS:
Macro-view of neighbourhood

Interpolating BFS and DFS

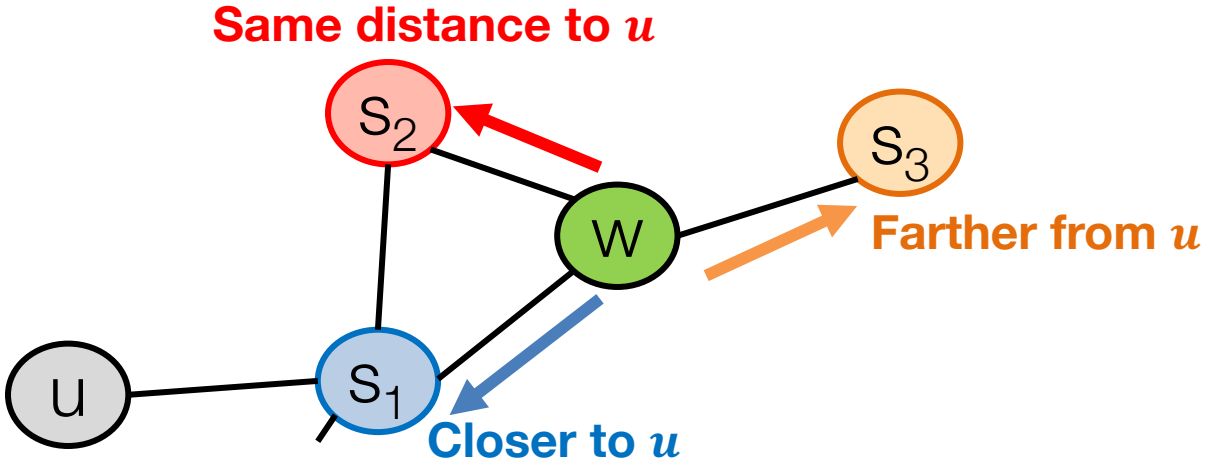
Biased random walk S that given a node u generates neighborhood $N_S(u)$

- Two parameters:
  - Return parameter p:
    - Return back to the previous node
  - In-out parameter q:
    - Moving outwards (DFS) vs. inwards (BFS)
    - Intuitively, q is the "ratio" of BFS vs. DFS

Biased Random Walks

Biased 2nd-order random walks explore network neighborhoods:

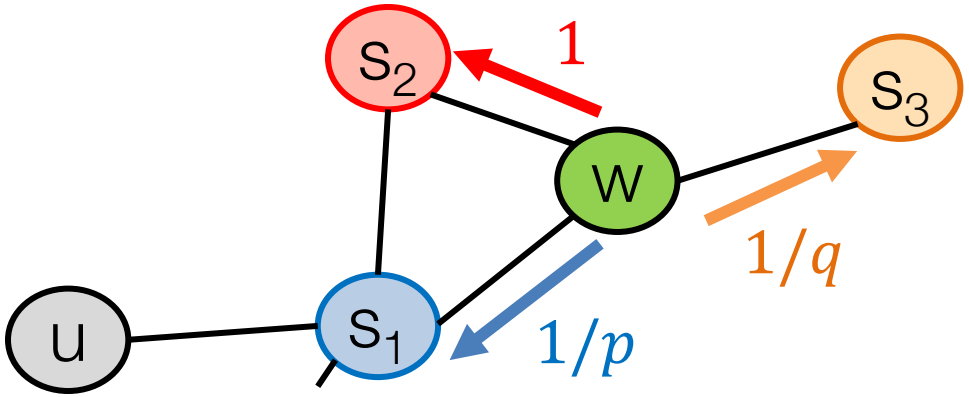
- Rnd. walk started at u and is now at w
- **Insight:** Neighbors of w can only be:



Idea: Remember where that walk came from

Biased Random Walks

- Walker is at $\mathbf{w}$. Where to go next?

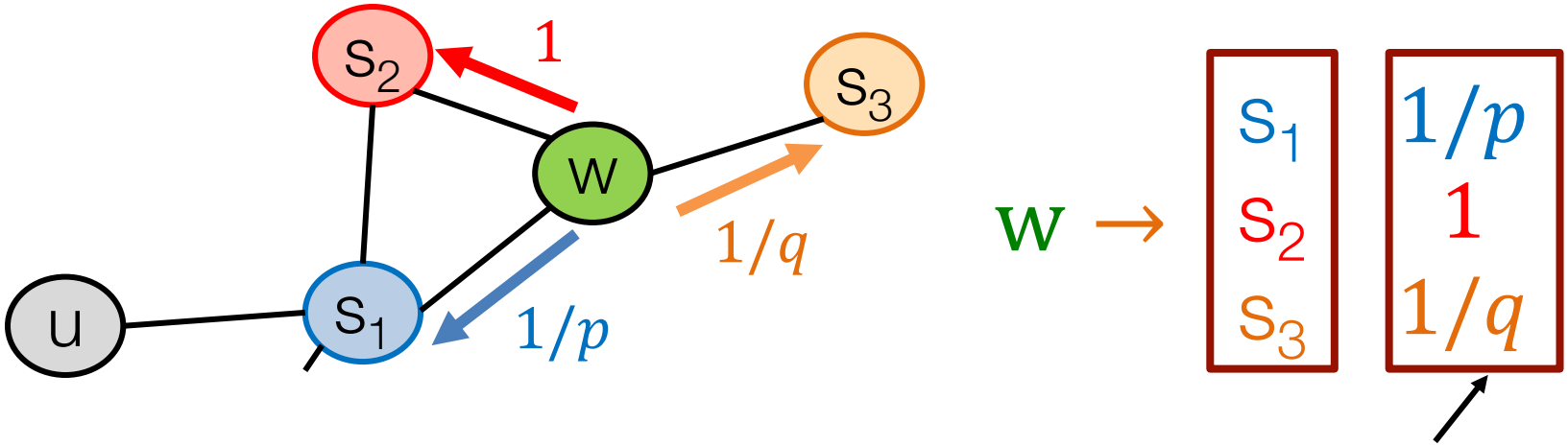


$1/p, 1/q, 1$ are unnormalized probabilities

- p, q model transition probabilities
  - p … return parameter
  - q … "walk away" parameter

Biased Random Walks

- Walker is at $\mathbf{w}$. Where to go next?



  - **BFS-like** walk: Low value of p
  - **DFS-like** walk: Low value of q

$N_S(u)$ are the nodes visited by the walker

node2vec algorithm

1) Simulate r random walks of length l starting from each node u

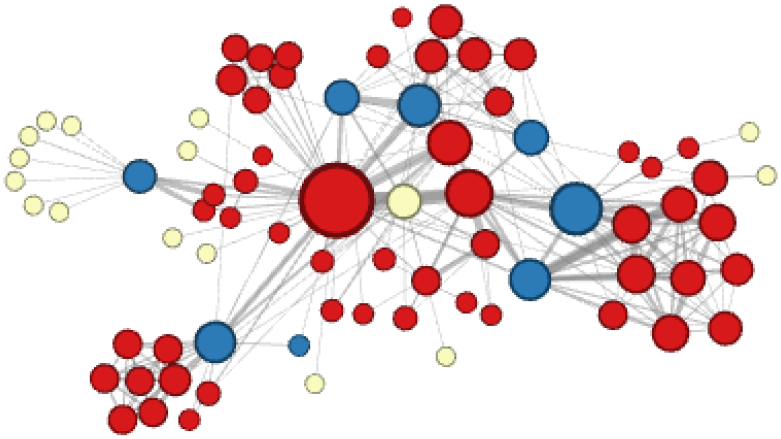
2) Optimize the node2vec objective using Stochastic Gradient Descent

Linear-time complexity

All 3 steps are individually parallelizable

Experiments: Micro vs. Macro

Network of character interactions in a novel



$p = 1, q = 2$

Microscopic view of the network neighbourhood

$p = 1, q = 0.5$

Macroscopic view of the network neighbourhood

Node Classification

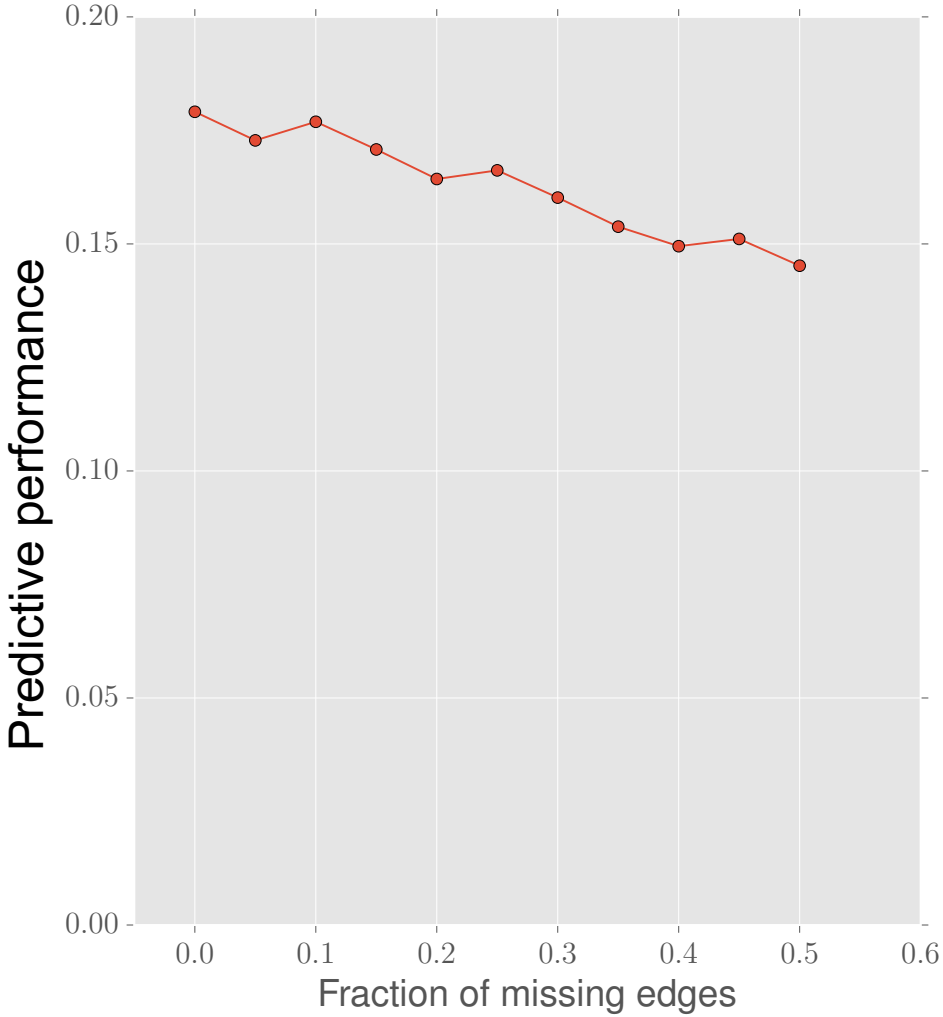
	BlogCatalog	Wiki-POS
Spectral Clustering	0.0405	0.0395
DeepWalk	0.2110	0.1274
LINE	0.0784	0.1164
node2vec	**0.2581**	**0.1552**

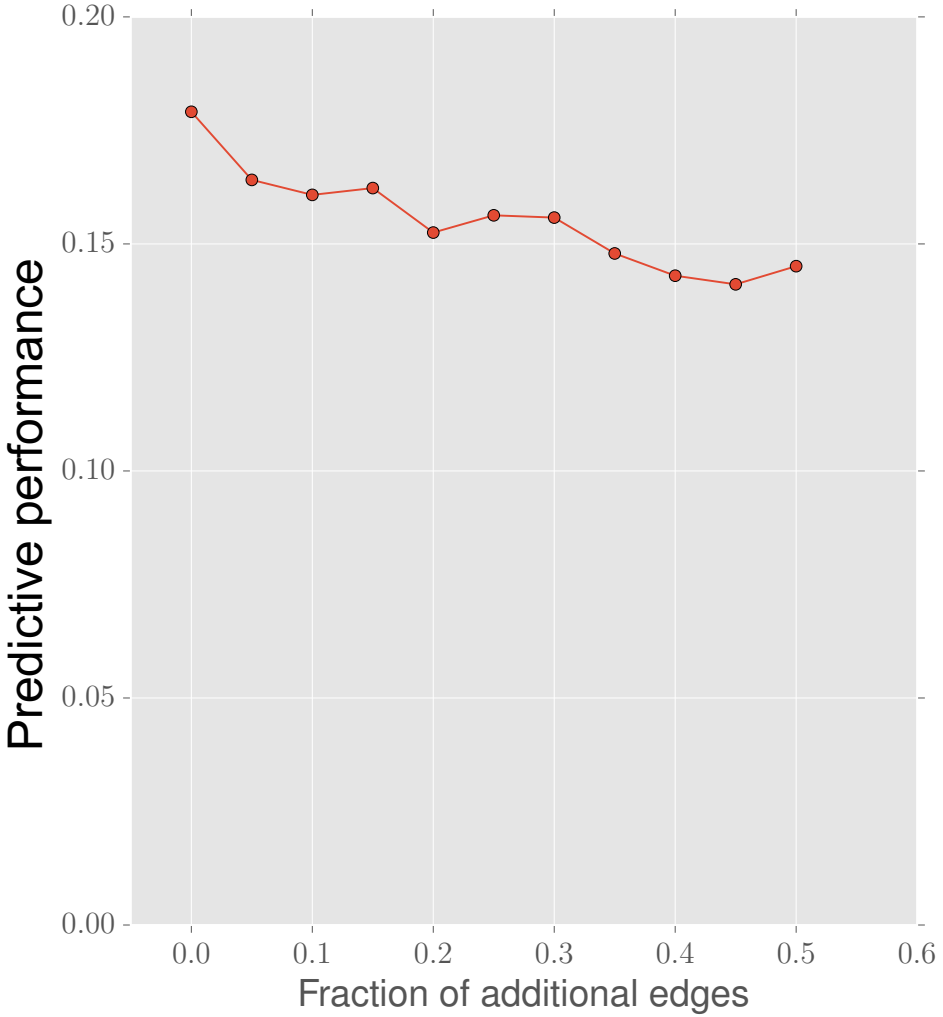
Macro-F_1 score

p, q	0.25, 0.25	4, 0.5
% gain	**22.3**	**21.8**

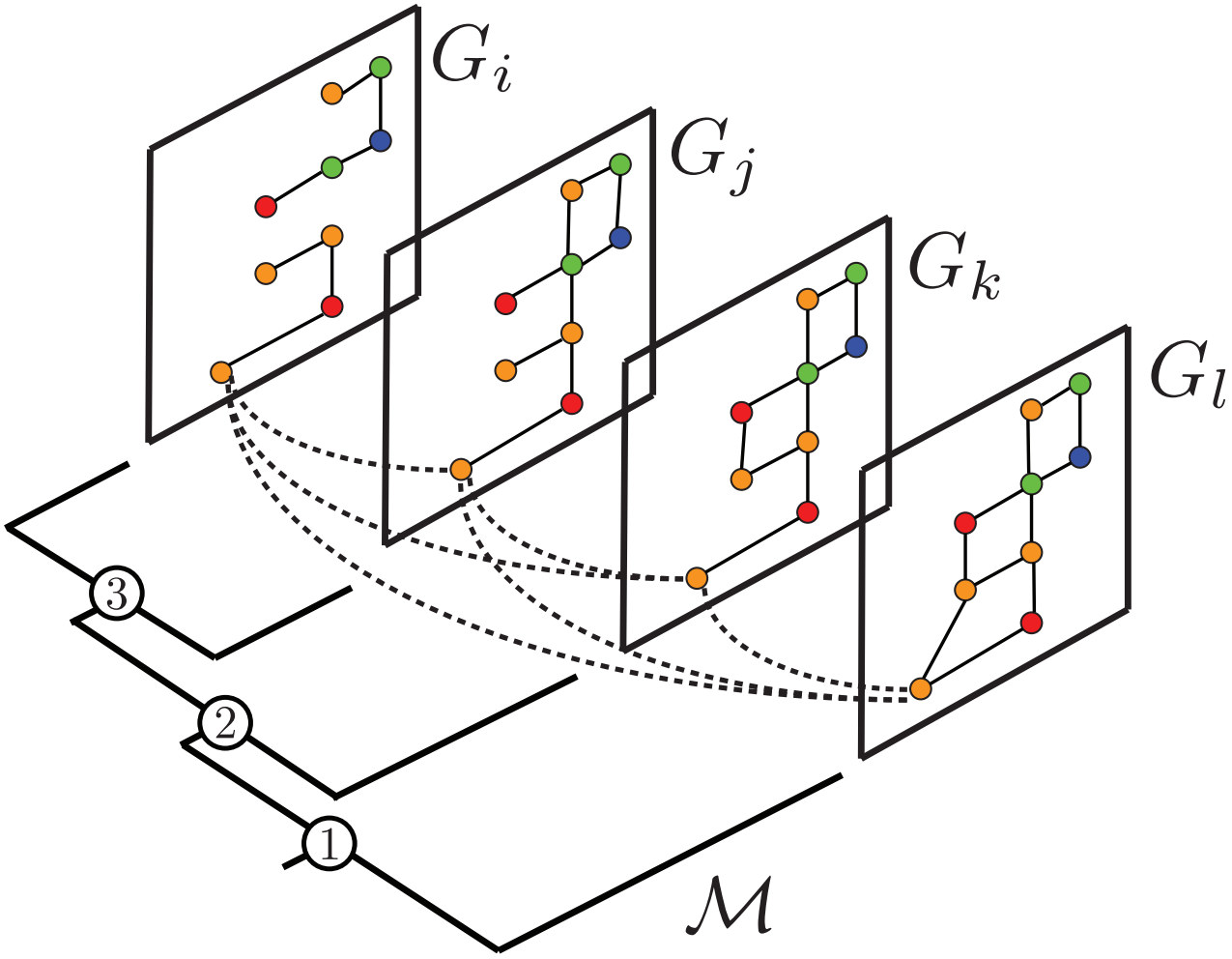
Outperforms in all cases, beating closest benchmark by up to 22%.

Incomplete Network Data



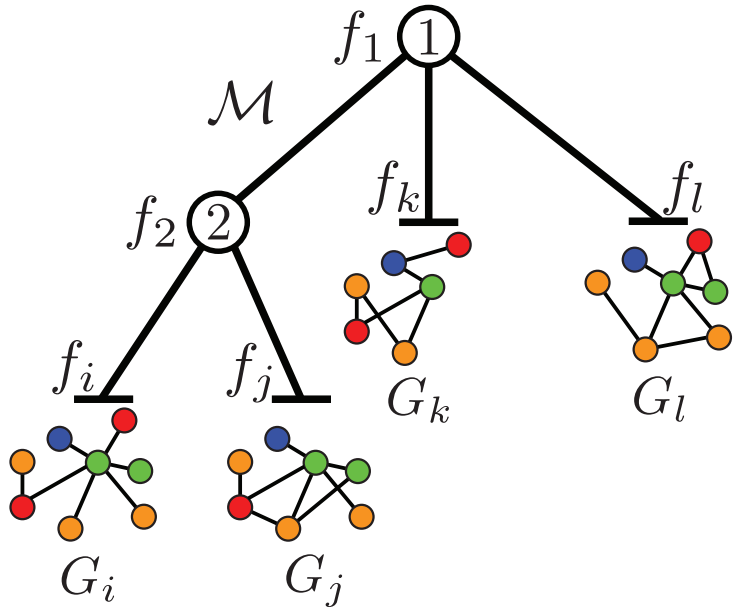


Multi-Layer Networks



Multi-Layer Networks

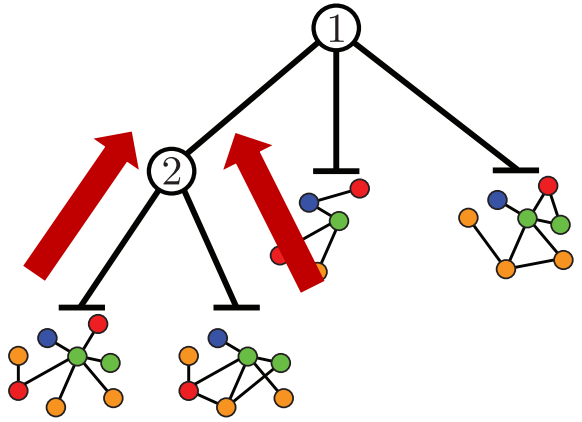
- Given layers G_i and hierarchy M
- Output: features of nodes in layers and in internal levels of the hierarchy



- Aim to capture multilevel hierarchical structure captured by M

Multi-Layer Networks

- For nodes in leaves G_i use node2vec objective
- For internal hierarchy:



$$c_i(u) = \frac{1}{2}\|f_i(u) - f_{\pi(i)}(u)\|_2^2.$$

  - $f_i(u)$ in layer i is close to $f_\pi(u)$ in parent $\pi(i)$

$$\max_{f_1, f_2, \ldots, f_{|M|}} \sum_{i \in \mathcal{T}} \Omega_i - \lambda \sum_{j \in \mathcal{M}} C_j.$$

Per-layer node2vec

Hierarchical dependency

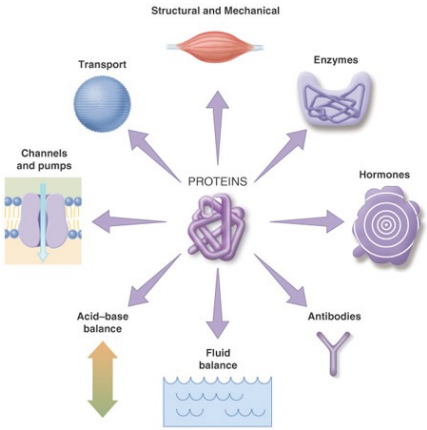
Implications

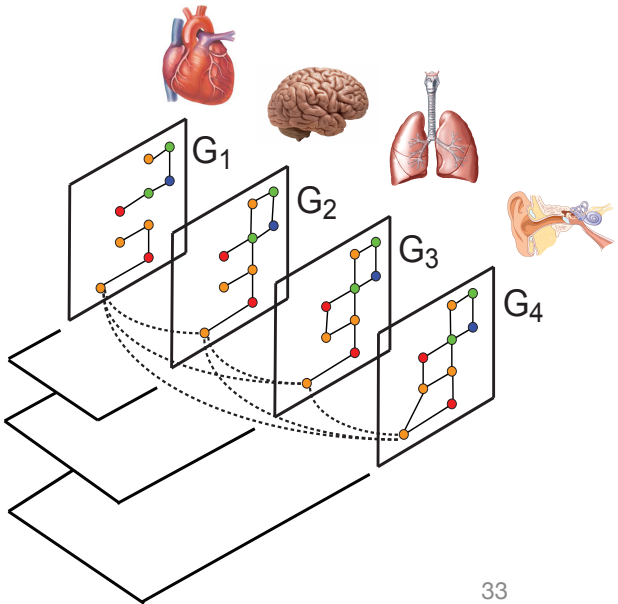
- Nodes in different layers representing the same entity/node have the same features in hierarchy ancestors
- We learn feature representations at multiple scales:
  - features of nodes in the layers
  - features of nodes in non-leaves in the hierarchy

Application: Protein function

- **Proteins are worker molecules**
  - Understanding protein function has great biomedical and pharmaceutical implications
- Function of proteins depends on their tissue context

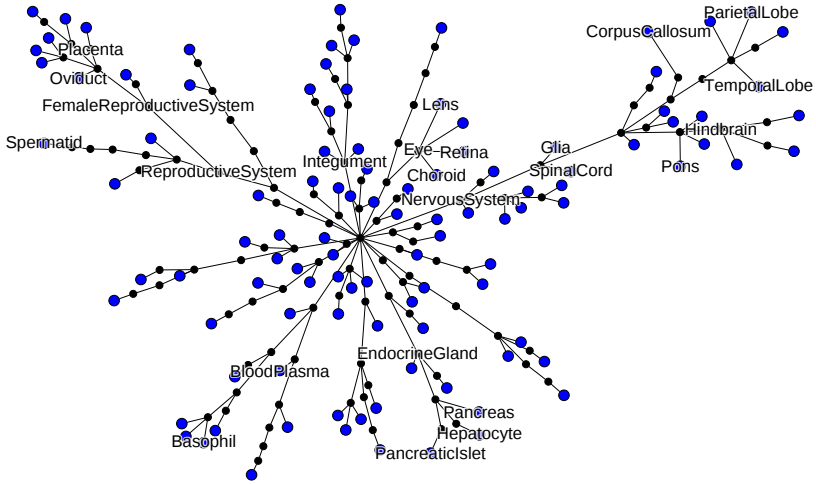
[Greene et al., Nat Genet '15]





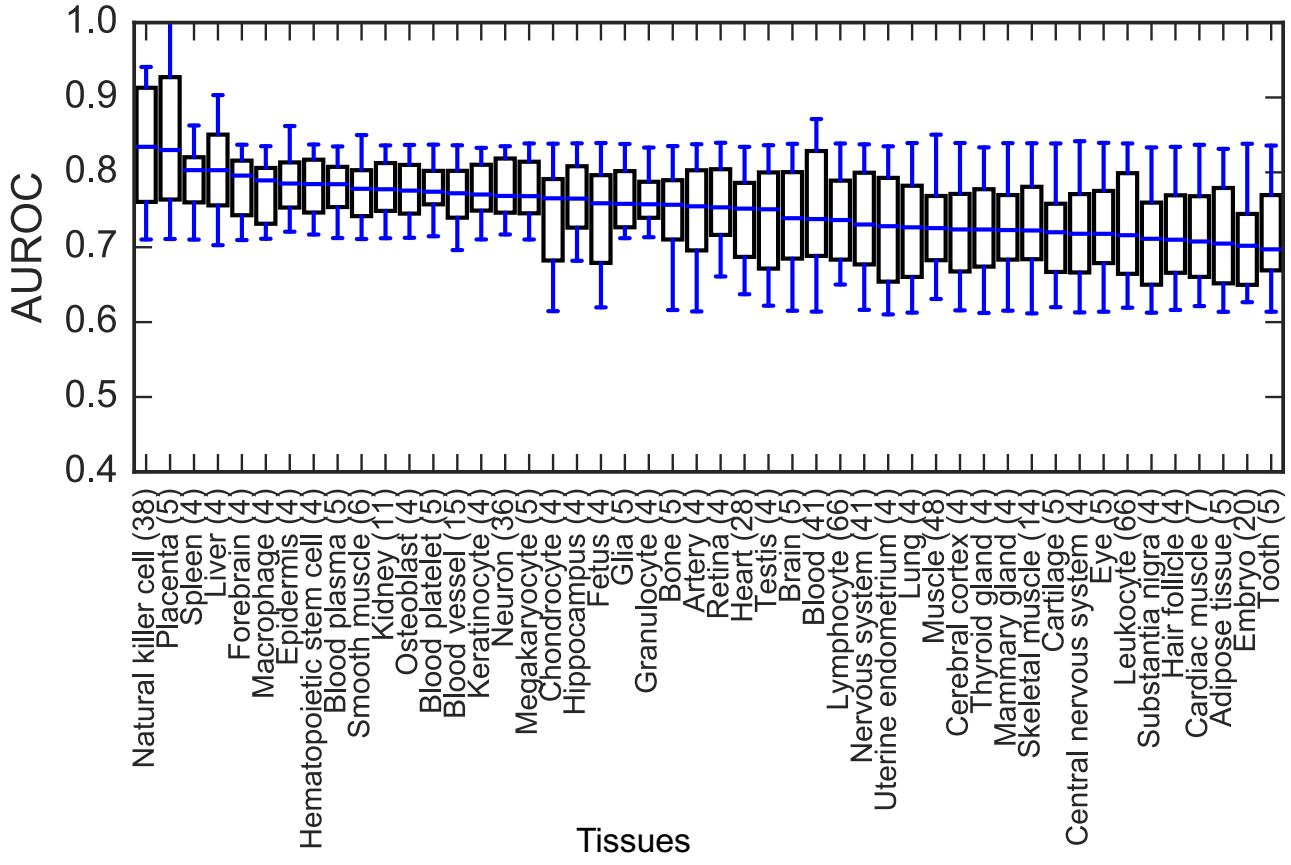
Experiments: Biological Nets

107 genome-wide tissue-specific protein interaction networks



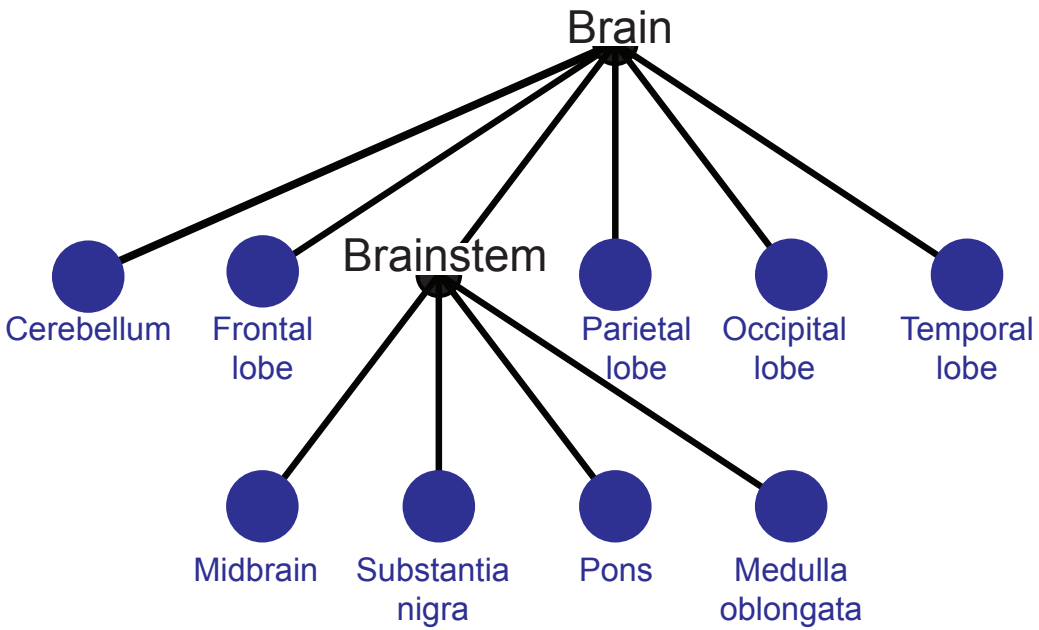
- 584 tissue-specific cellular functions
- Examples (tissue, cellular function):
  - (renal cortex, cortex development)
  - (artery, pulmonary artery morphogenesis)

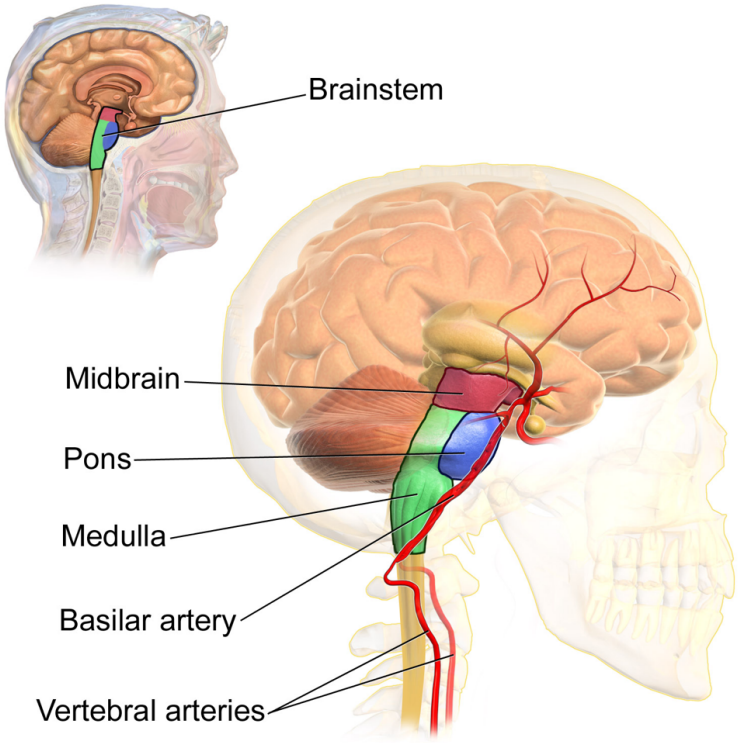
Tissue Specific Prediction



42% improvement over state-of-the-art baseline

Brain Tissues

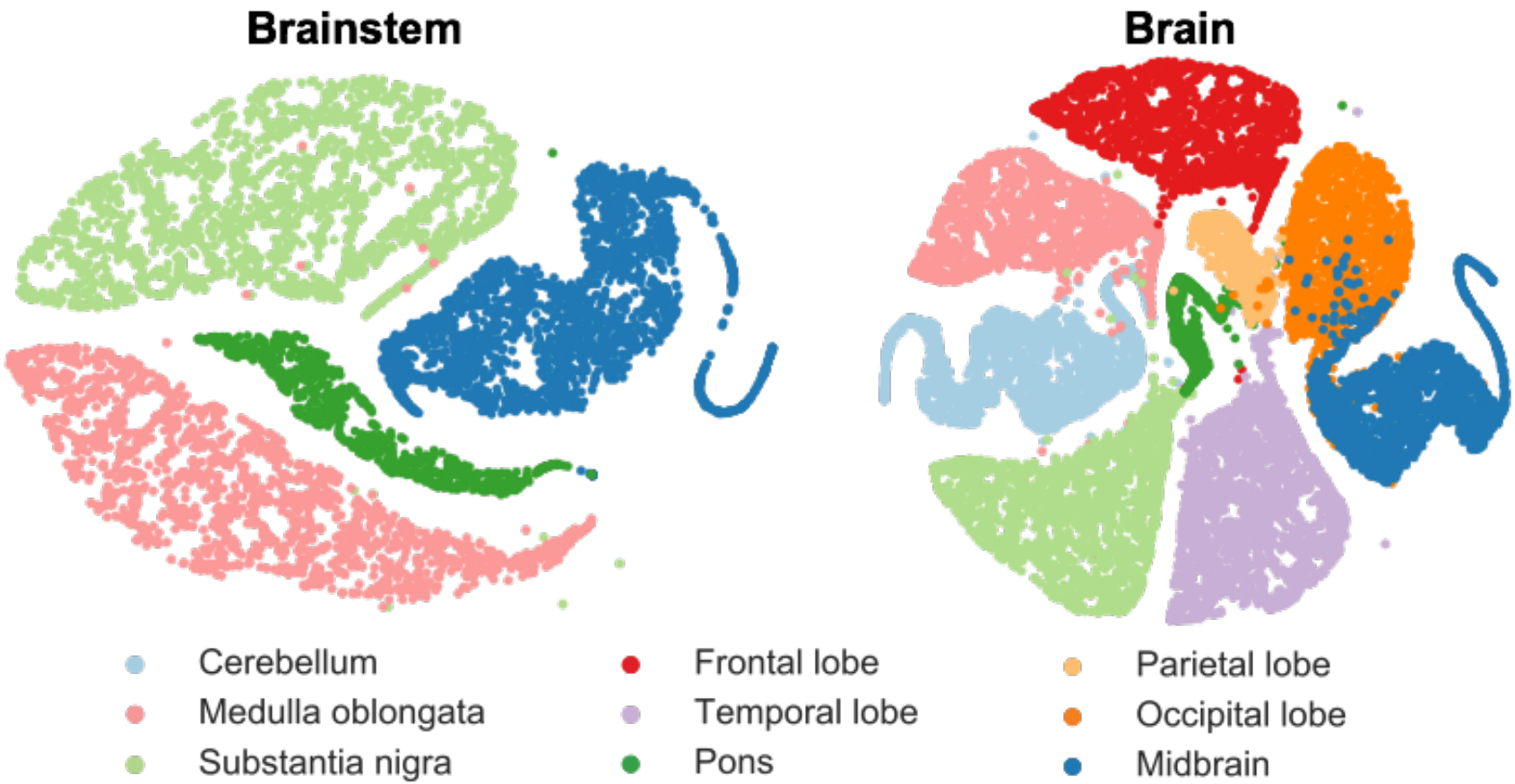




9 brain tissue PPI networks in two-level hierarchy

Embedding Brain Networks

- Do embeddings match anatomy?



node2vec: Summary

Task-independent feature learning in networks:

- An explicit locality preserving objective for feature learning
- Biased random walks capture diversity of network patterns
- Scalable and robust algorithm

A Different Setting

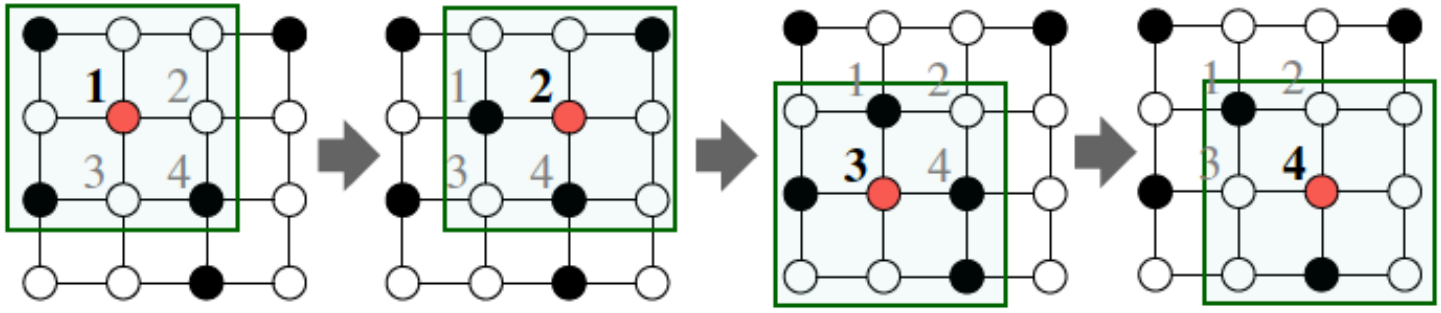
- So far: Node2vec
  - Unsupervised (task-agnostic)
  - Nodes have not attributes
- Next: GraphSage
  - Supervised (task-specific)
  - Nodes have attributes
    - Text, image, etc.

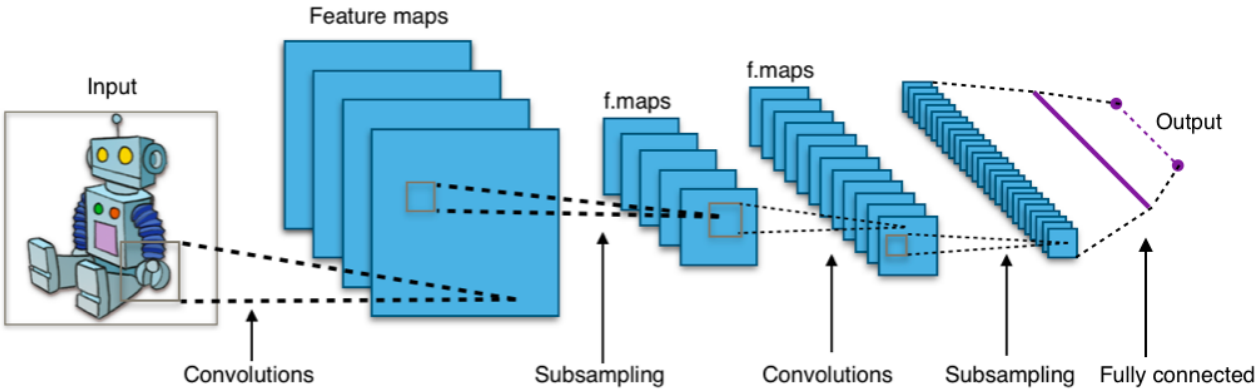
GraphSAGE: Supervised Feature Learning

Inductive Representation Learning on Large Graphs.
W. Hamilton, R. Ying, J. Leskovec. Neural Information Processing Systems (NIPS), 2017.
Representation Learning on Graphs: Methods and Applications.
W. Hamilton, R. Ying, J. Leskovec. IEEE Data Engineering Bulletin, 2017.

Idea: Convolutional Networks

- **CNN on an image:**

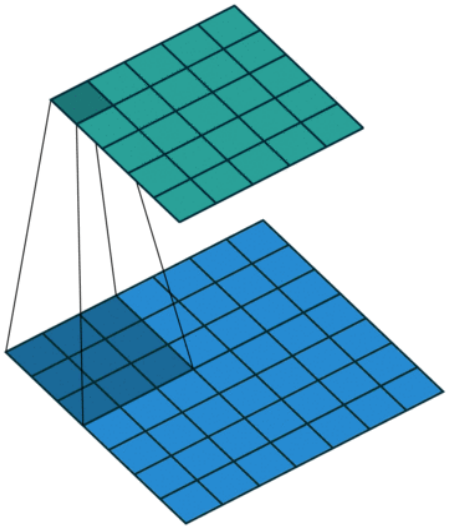




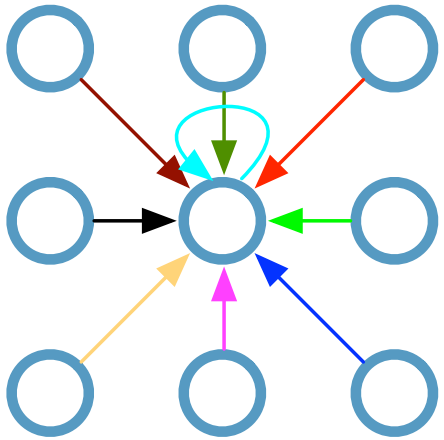
Goal is to generalize convolutions beyond simple lattices
Leverage node features/attributes (e.g., text, images)

From Images to Networks

Single CNN layer with 3x3 filter:



Image



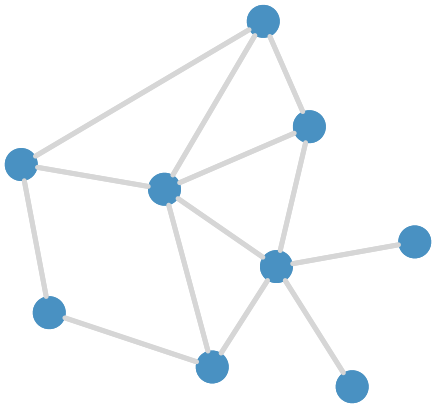
Graph

Transform information at the neighbors and combine it

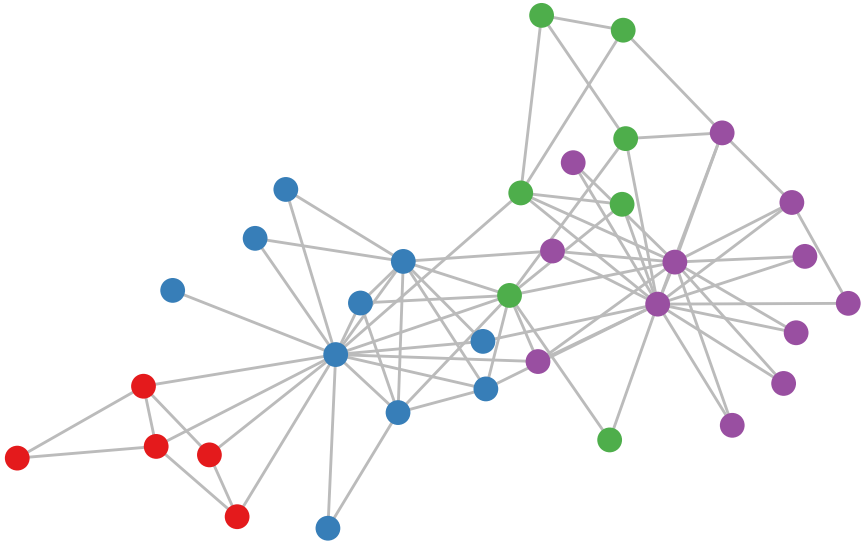
- Transform "messages" h_i from neighbors: $W_i\, h_i$
- Add them up: $\sum_i W_i\, h_i$

Real-World Graphs

But what if your graphs look like this?



or this:

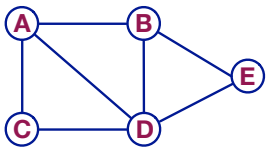


- Examples:

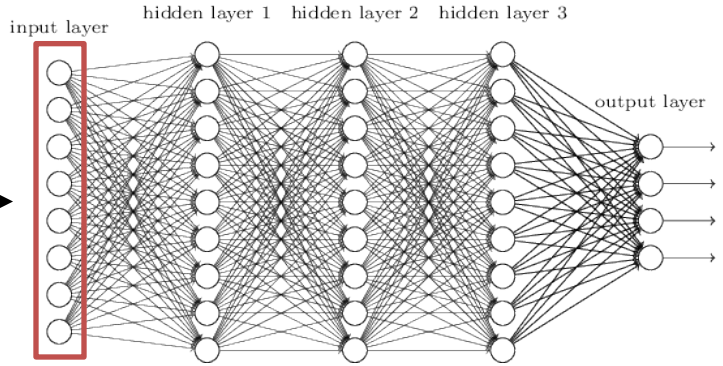
  Social networks, Information networks, Knowledge graphs, Communication networks, Web graph, …

A Naïve Approach

- Join adjacency matrix and features
- Feed them into a deep neural net:



	A	B	C	D	E	Feat	
A	0	1	1	1	0	1	0
B	1	0	0	1	1	0	0
C	1	0	0	1	0	0	1
D	1	1	1	0	1	1	1
E	0	1	0	1	0	1	0

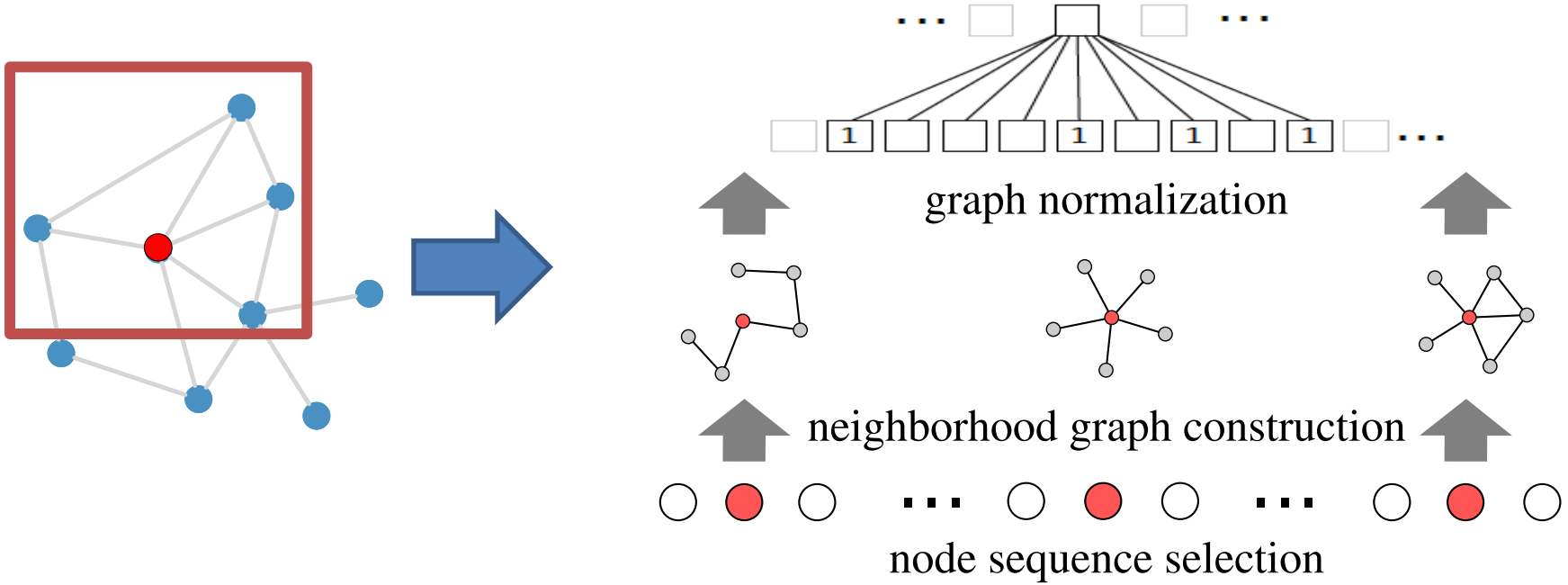


?

- Problems:
  - $O(N)$ parameters
  - No inductive learning possible
  - Not invariant to node ordering

Graph Convolutional Networks

Graph Convolutional Networks:

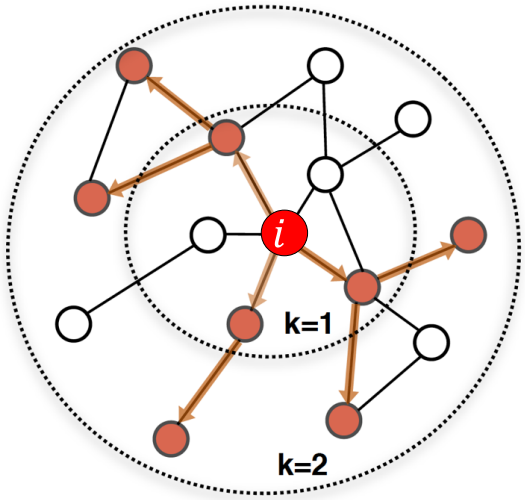


Problem: For a given subgraph how to come with canonical node ordering

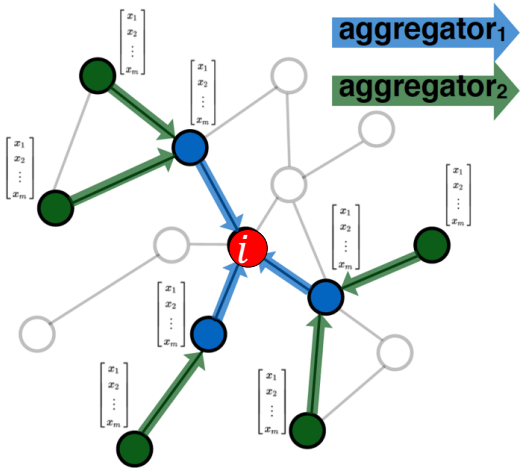
Niepert, Mathias, Mohamed Ahmed, and Konstantin Kutzkov. "Learning convolutional neural networks for graphs." ICML. 2016.

Our Approach: GraphSAGE

Idea: Node's neighborhood defines a computation graph



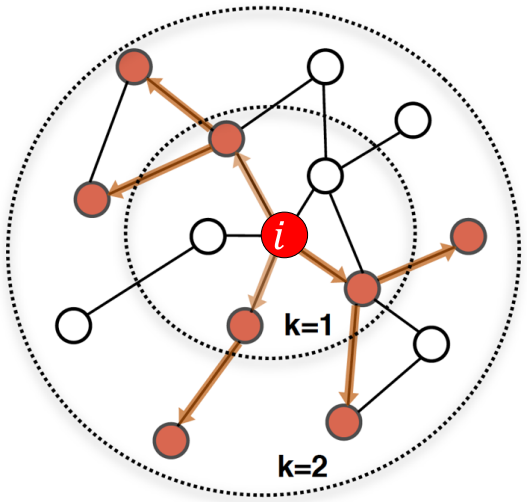
Determine node computation graph



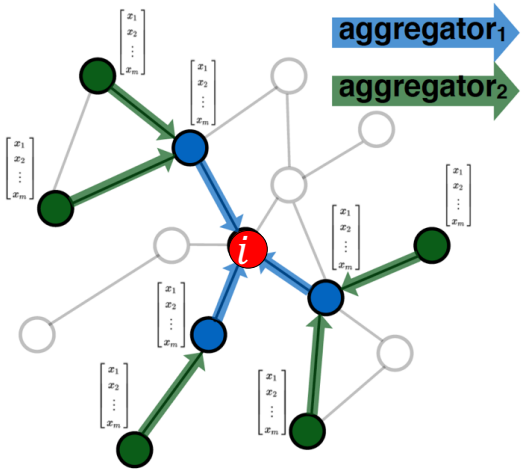
Propagate and transform information

Learn how to propagate information across the graph to compute node features

Our Approach: GraphSAGE



Determine node computation graph



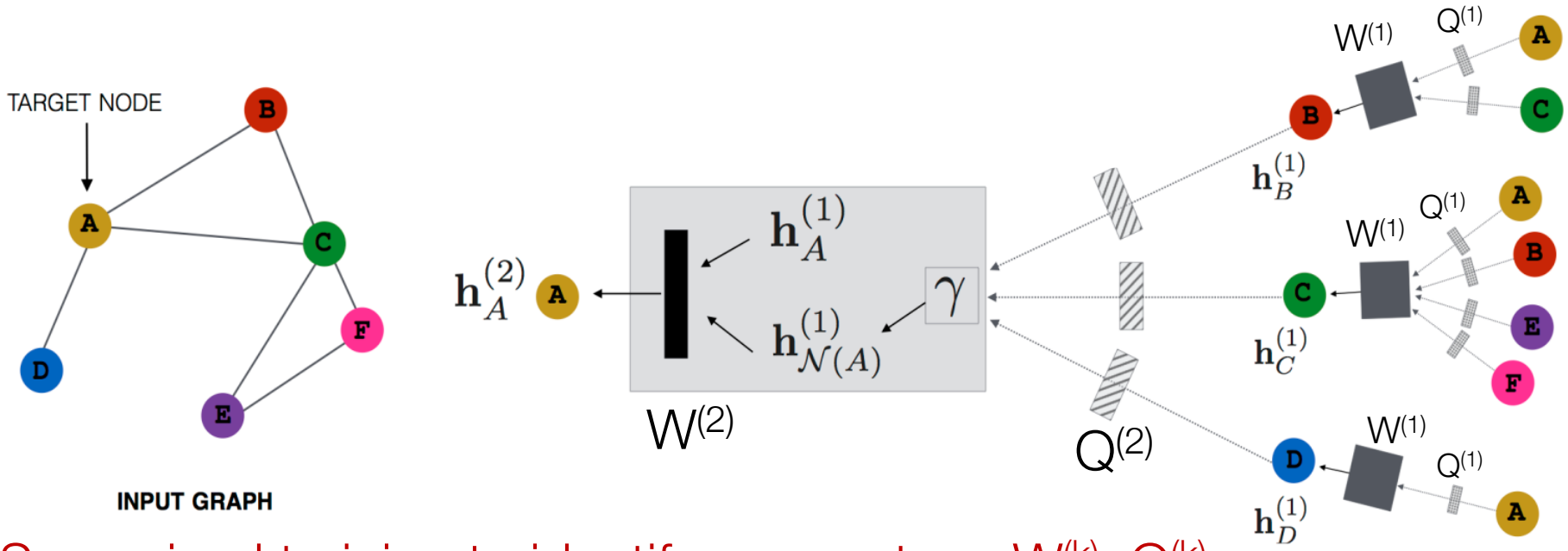
Propagate and transform information

Update for node i:

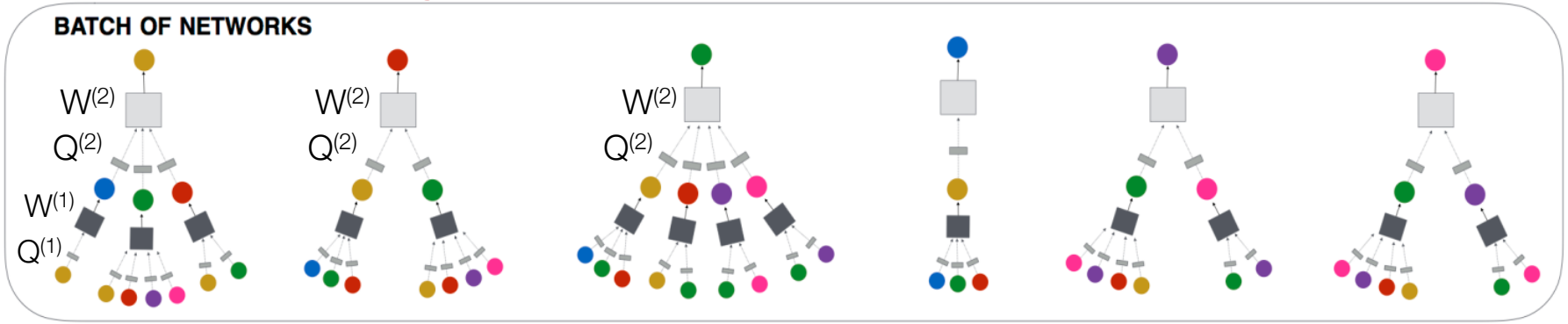
$$h_i^{(k+1)} = ReLU\left(W^{(k)} h_i^{(k)}, \sum_{n \in \mathcal{N}(i)} \left(ReLU(Q^{(k)} h_n^{(k)})\right)\right)$$

- $h_i^{(0)} = X_i$ (directly leverage node attributes)
- $\Sigma(\cdot)$: Aggregator function (avg., LSTM, max-pooling)

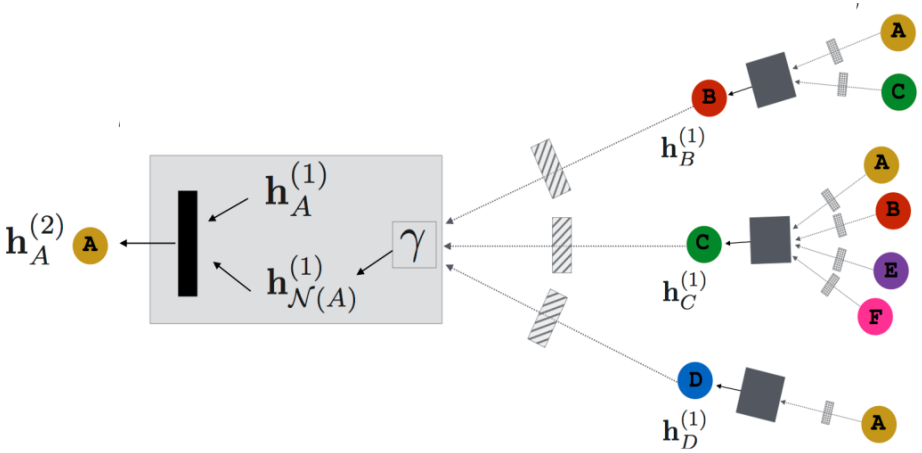
GraphSAGE: Example



Supervised training to identify parameters: $W^{(k)}$, $Q^{(k)}$



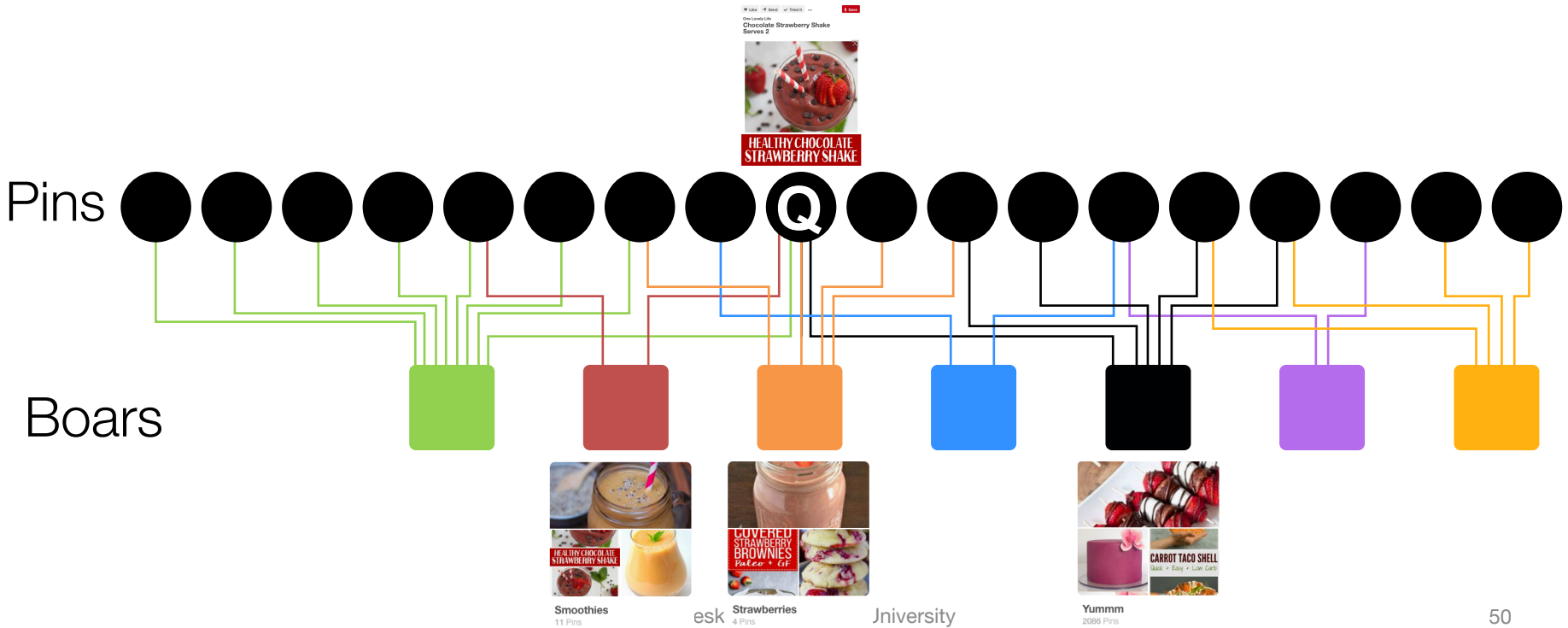
GraphSAGE: Benefits



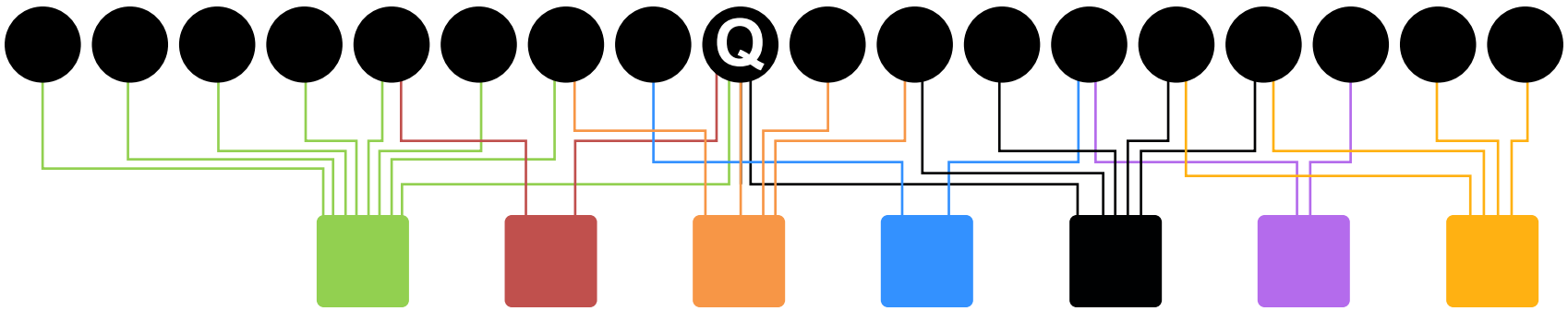
- Can use different aggregators γ
  - Mean (simple element-wise mean), LSTM (to a random order of nodes), Max-pooling (element-wise max)
- Can use different loss functions:
  - Cross entropy, Hinge loss, ranking loss
- Model has a constant number of parameters
- Fast scalable inference
- Can be applied to any node in any network

Large-Scale Application

- Supervised node embedding for graph-based recommendations
- Application: **Pinterest**



Pinterest Graph



Graph: 2B pins, 1B boards, 17B edges

- Graph is dynamic: need to apply to new nodes without model retraining
- Rich node features: content, image

Task: Item-Item Recs

Related Pin recommendations

- Given user is looking at pin **Q**, what pin **X** are they going to save next:

Query

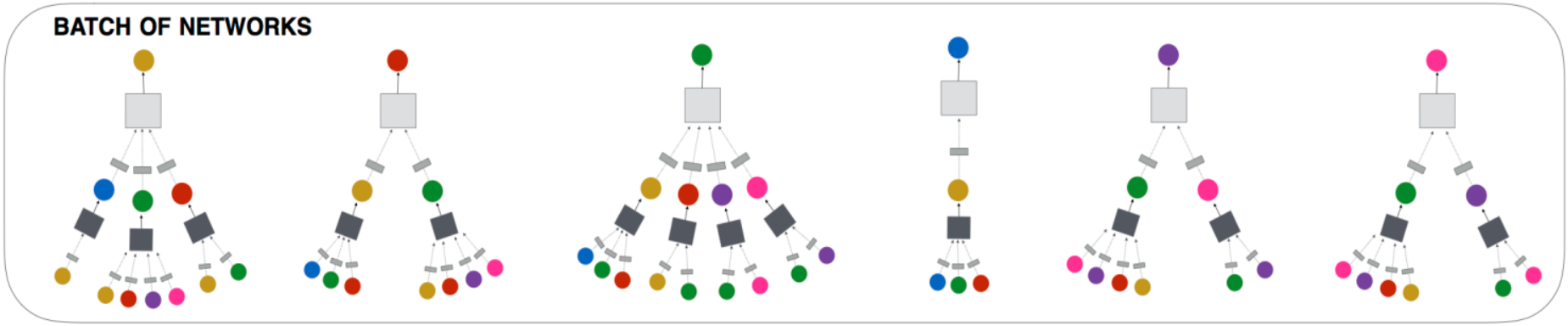
Positive

Rnd. negative

Hard negative

GraphSAGE Training

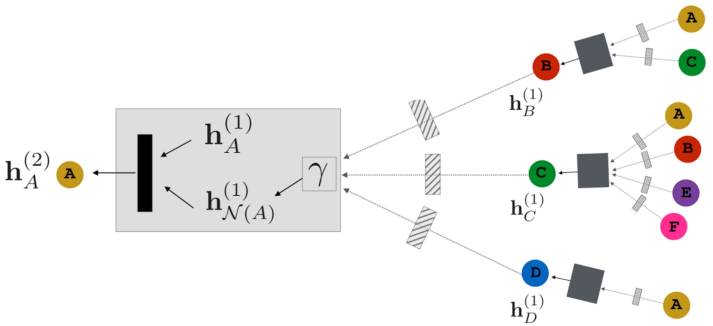
- Leverage inductive capability, and train on individual subgraphs
  - 300 million nodes, 1 billion edges, 1.2 billion pin pairs **(Q, X)**

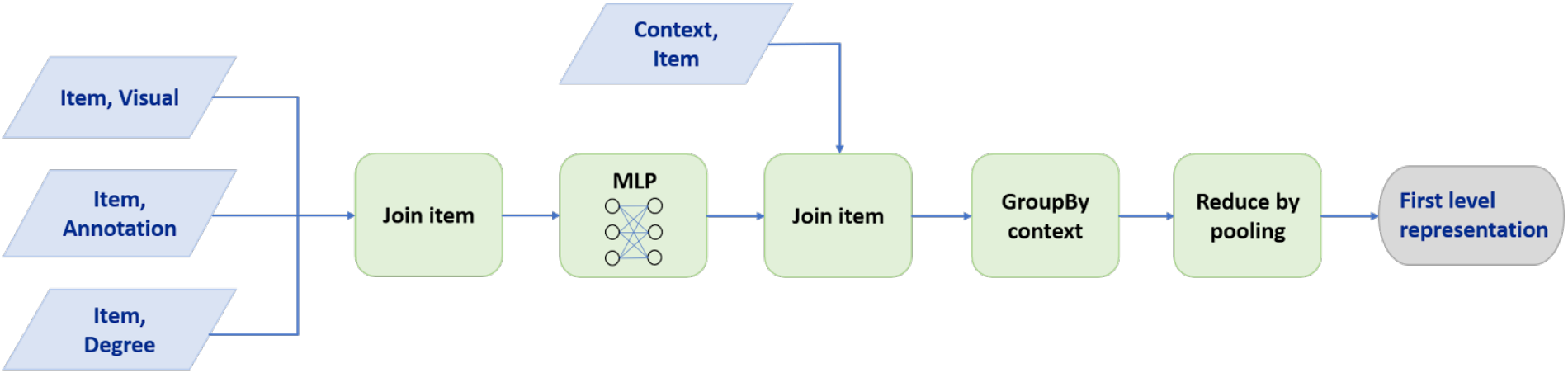


- Large batch size: 2048 per minibatch

GraphSAGE: Inference

- Use MapReduce for model inference





- Avoids repeated computation

Experiments

Related Pin recommendations

- Given user is looking at pin **Q**, predict what pin **X** are they going to save next
- **Baselines for comparison**
  - **Visual:** VGG-16 visual features
  - **Annotation:** Word2Vec model
  - **Combined:** combine visual and annotation
  - **RW:** Random-walk based algorithm
  - **GraphSAGE**
- **Setup:** Embed 2B pins, perform nearest neighbor to generate recommendations

Results: Ranking

Task: Given **Q**, rank **X** as high as possible among **2B** pins

- Hit-rate: Pct. P was among top-*k*
- MRR: Mean reciprocal rank

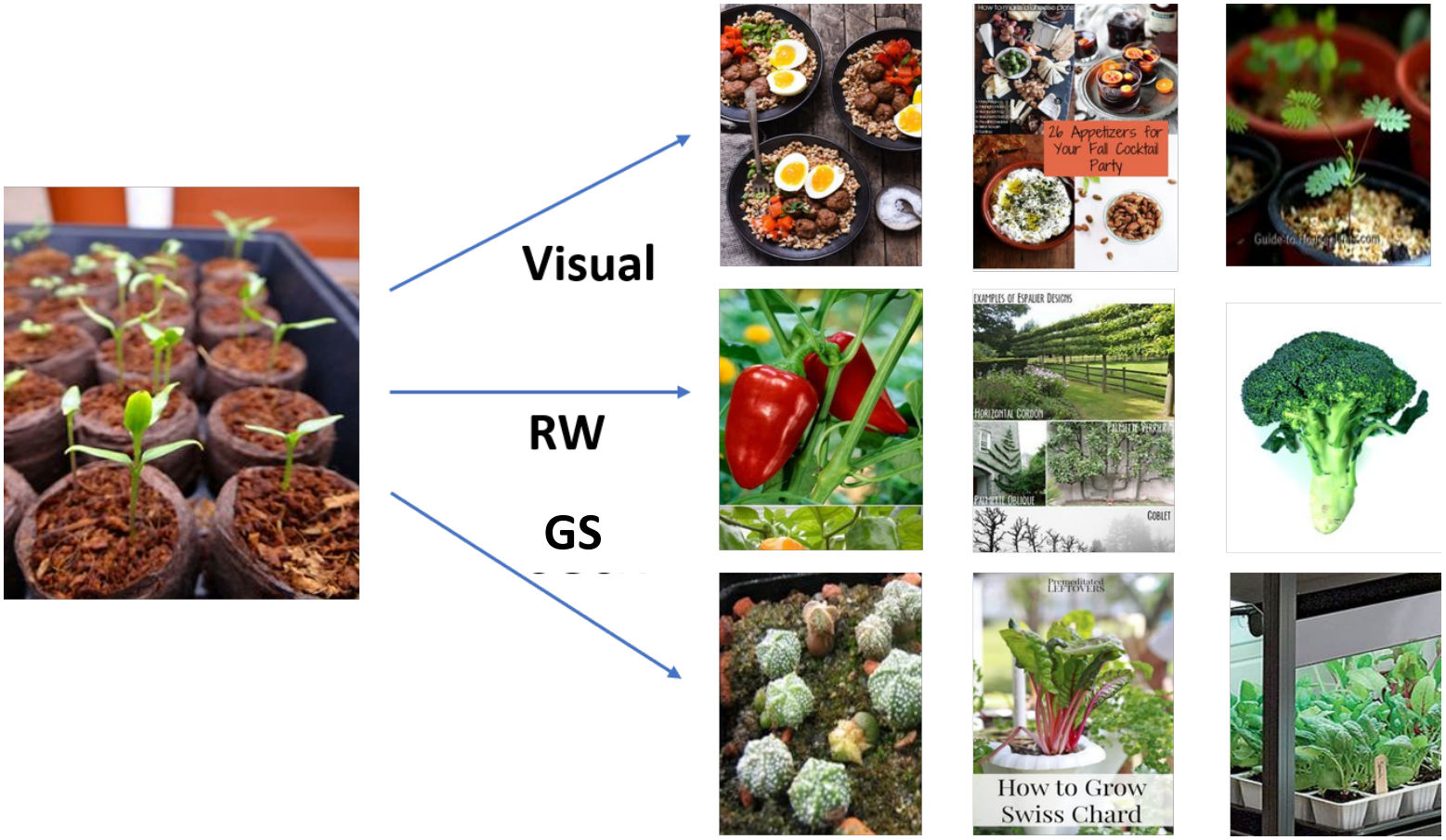
Method	Hit-rate	MRR
Visual	17%	0.23
Annotation	14%	0.19
Combined	27%	0.37
GraphSAGE	46%	0.56

Results: User Study

- **User study:** Which recommendation do you prefer?

Method	Win	Lose	Draw	Fraction of Wins
GraphSAGE vs. Visual	26.7%	18.6%	54.7%	58.9%
GraphSAGE vs. Annotation	28.4%	16.1%	55.5%	63.8%
GraphSAGE vs. RW	32.2%	21.4%	46.4%	60.1%

Example Recommendations

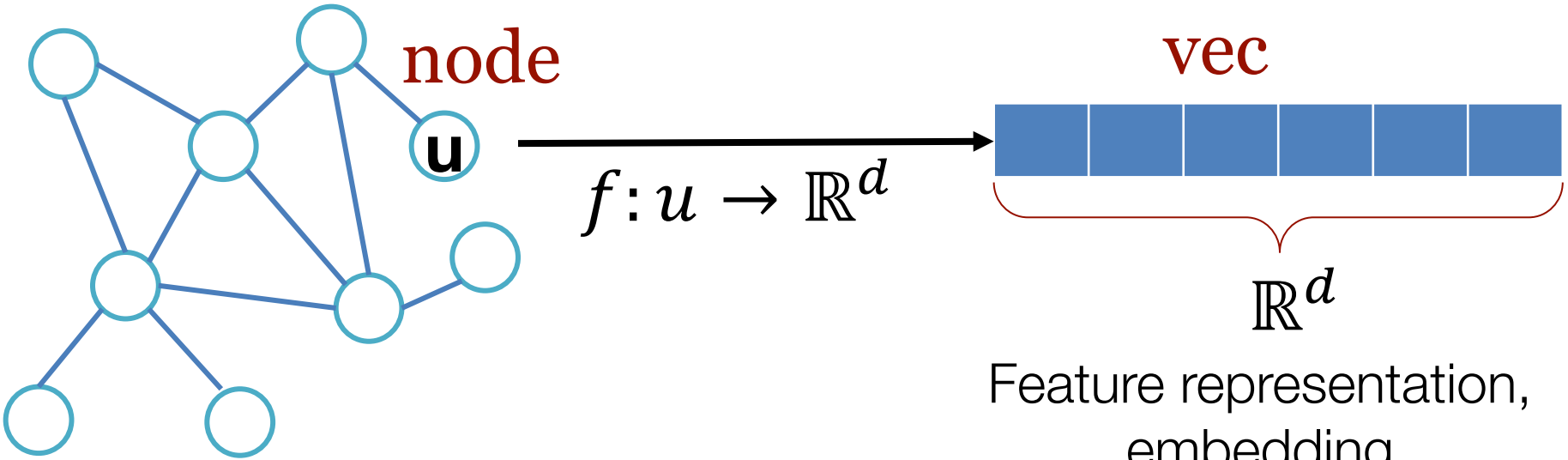


GraphSAGE: Summary

- Graph Convolution Networks
  - Generalize beyond simple convolutions
- Fuses node features & graph info
  - State-of-the-art accuracy for node classification and link prediction.
- Model size independent of graph size; can scale to billions of nodes
  - Largest embedding to date (3B nodes, 17B edges)
- Leads to significant performance gains

Conclusion

Feature learning for networks



Conclusion

Results from the past 1-2 years have shown:

- Representation learning paradigm can be extended to graphs
- No feature engineering necessary
- State-of-the-art results in a number of domains
- Use end-to-end training instead of multi-stage approaches

Conclusion

Next steps:

- Multimodal & dynamic/evolving settings
- Domain-specific adaptations (e.g. for recommender systems)
- Graph generation
- Prediction beyond simple parwise edges
- Theory
- Scalability

Post-doc positions open!

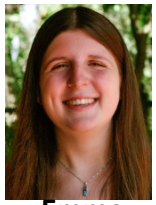
Email us at jure@cs.stanford.edu

PhD Students


Claire Donnat

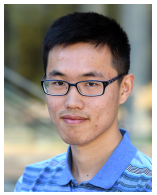

Mitchell Gordon


David Hallac


Emma Pierson


Geet Sethi


Himabindu Lakkaraju


Rex Ying


Tim Althoff


Will Hamilton

Post-Doctoral Fellows


David Jurgens

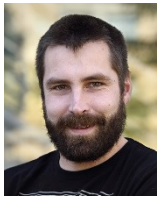

Marinka Zitnik


Michele Catasta


Srijan Kumar

Research Staff


Stephen Bach


Peter Kacin


Rok Sosic

Industry Partnerships

Funding

Collaborators

Dan Jurafsky, Linguistics, Stanford University
Christian Danescu-Miculescu-Mizil, Information Science, Cornell University
Stephen Boyd, Electrical Engineering, Stanford University
David Gleich, Computer Science, Purdue University
VS Subrahmanian, Computer Science, University of Maryland
Sarah Kunz, Medicine, Harvard University
Russ Altman, Medicine, Stanford University
Jochen Profit, Medicine, Stanford University
Eric Horvitz, Microsoft Research
Jon Kleinberg, Computer Science, Cornell University
Sendhill Mullainathan, Economics, Harvard University
Scott Delp, Bioengineering, Stanford University
Jens Ludwig, Harris Public Policy, University of Chicago

References

- node2vec: Scalable Feature Learning for Networks A. Grover, J. Leskovec. KDD 2016.
- Predicting multicellular function through multi-layer tissue networks. M. Zitnik, J. Leskovec. Bioinformatics, 2017.
- Inductive Representation Learning on Large Graphs. W. Hamilton, R. Ying, J. Leskovec. NIPS 2017
- Representation Learning on Graphs: Methods and Applications. W. Hamilton, R. Ying, J. Leskovec. IEEE Data Engineering Bulletin, 2017.

- Code:
  - http://snap.stanford.edu/node2vec
  - http://snap.stanford.edu/graphsage