Inductive Representation Learning On Large Graphs

- William L. Hamilton, Rex Ying, Jure Leskovec

What is the problem?

How can we learn node embeddings for very large graphs (millions to billions of nodes), with time and memory constraints?

How can we generate node embeddings ‘online’?

How do we quickly generate embeddings for unseen nodes?

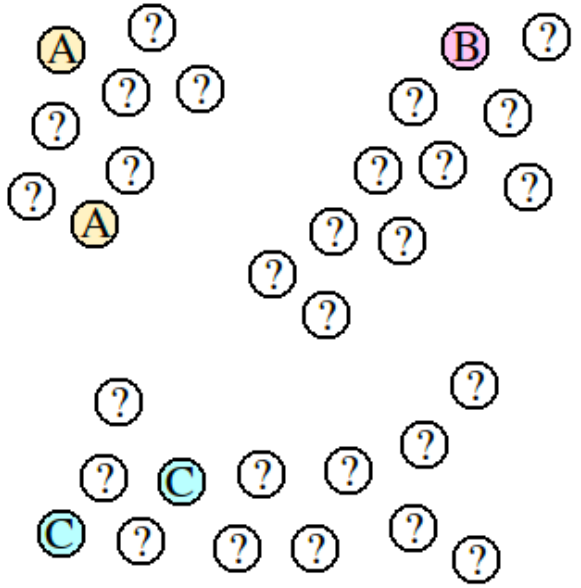
How can we leverage the existing embeddings to a new graph consisting of nodes from the same domain?

Transduction

The problem is that shallow-embedding based approaches are inherently *transductive*.

They reason from the labelled points in the training set to unlabelled points in the test set. *They see the full graph.*

We want our model to learn something more fundamental, that can be applied outside of the graphs that it was trained on.



Induction

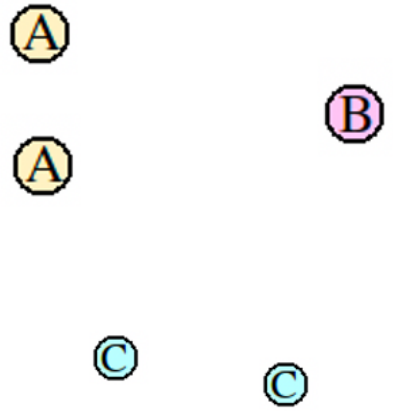
We want our model to learn something more fundamental, *just from the initial set of examples that it sees.*

This knowledge should be applicable to unseen nodes / graphs.

We want our model to be *inductive*.

We want it to learn *rules* from the training data.

We want it to learn a *function* that it can apply to unseen data.



Why is induction useful?

Most real-world networks are constantly increasing in size (e.g. most networks on social media). Training is expensive, therefore ML systems need to be able to generalize to new data.

Models can transfer knowledge to entirely new graphs (with nodes that have similar features).

The inductive knowledge can be parameterized. We need only to train and store these parameters.

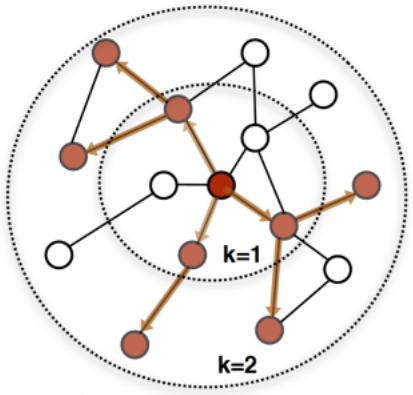
Learning can be efficiently done *online.*

GraphSAGE
(Sampling and Aggregation)

- Application of GCN to the inductive setting.
- Framework that generalizes and extends the GCN approach to different trainable aggregation functions (as opposed to only convolution)
- Faster, computationally less expensive training process via sampling a fixed-size neighborhood (mini-batching).
- Leverages node attribute information to learn functions that can generate node embeddings for unseen nodes.

Forward propagation

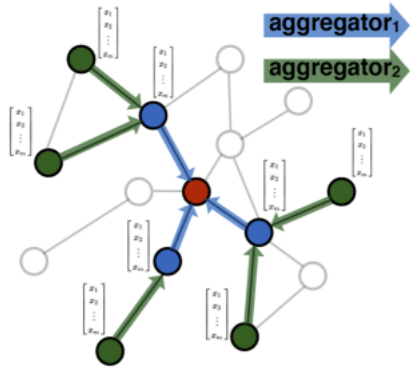
Sample neighborhoods of all nodes in the current minibatch, up to the depth K.



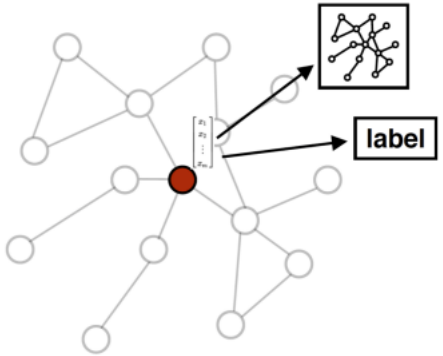
Aggregate and compute of sampled nodes that are in the k-hope neighborhood.

Concatenate this representation with the node's depth (k-1) representation, and apply a non-linearity.

Each step aggregates information from nodes that are in the k-hop neighborhood.



The embedding z is the output of the final layer.



Aggregation functions

Mean aggregator

Sample-wise mean of the hidden representations at each step

Convolutional aggregator

Modified mean aggregator (*without any concatenation)*

Very similar to the GCN framework, since the mean operation can be considered to be a rough approximation to a local spectral convolution

LSTM aggregator

Use an LSTM for its expressivity

Loses the permutation invariance of the aggregation function

Pooling aggregator

Pass the representation h for each neighbor into a feedforward NN and take mean or max (element wise)

Experiments

Generalization on evolving graphs

Predict paper subject categories on an undirected citation graph that was derived from the Thomson Reuters Web of Science Core Collection.

1. Six different node labels.
2. 2000-2004 data used for training and the 2005 data data is used for testing (30% for validation).

Predict the subreddit of posts from a post-to-post graph dataset of Reddit posts made in September, 2014.

1. Posts were connected if the same user comments on both.
2. First 20 days were used for training and the remaining for testing (with 30% for validation).

Generalization across graphs

Classification of cellular functions of proteins in various protein-protein interaction (PPI) graphs, across different human tissues.

Gene ontology sets (bins of pre-defined functions) were labels (121 in total).

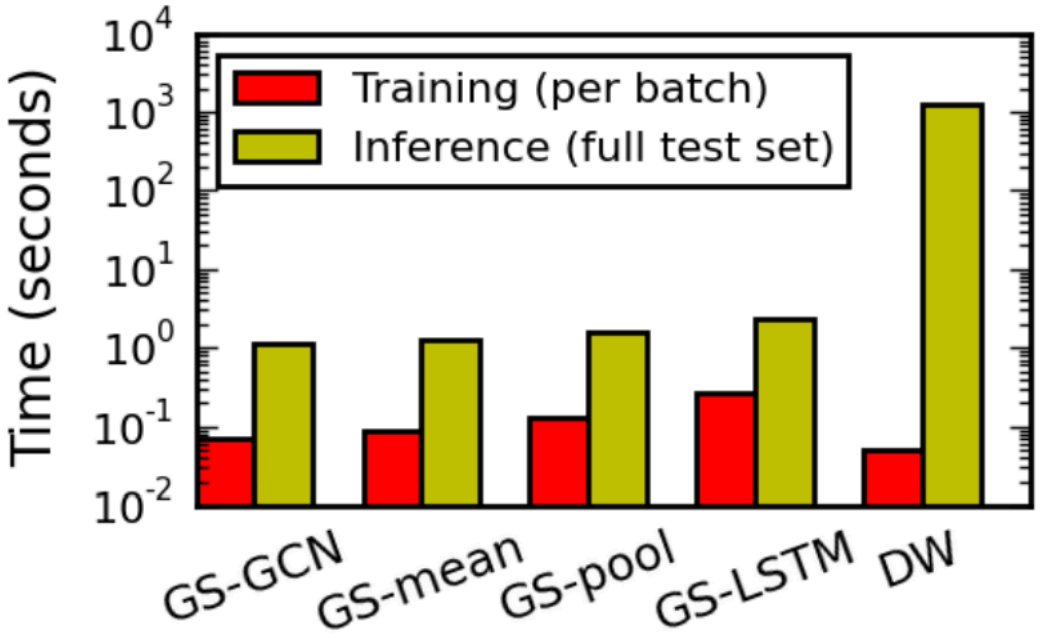
20 graphs are used for training, 2 for validation, and 2 graphs for testing (average F1 score reported).

Results

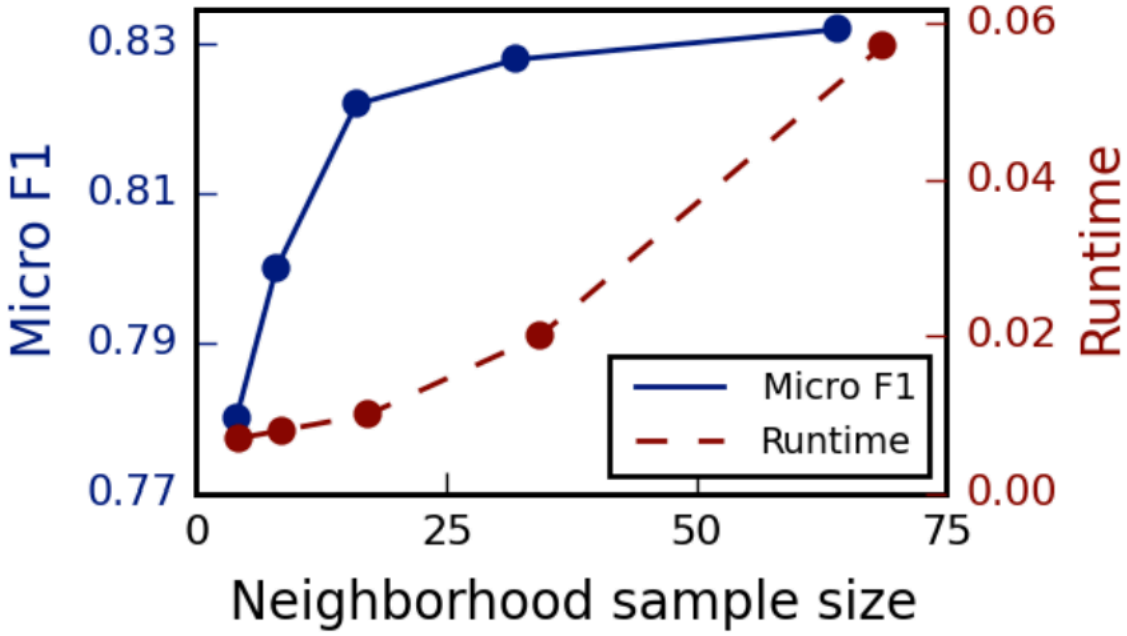
	Citation		Reddit		PPI	
Name	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1
Random	0.206	0.206	0.043	0.042	0.396	0.396
Raw features	0.575	0.575	0.585	0.585	0.422	0.422
DeepWalk	0.565	0.565	0.324	0.324	—	—
DeepWalk + features	0.701	0.701	0.691	0.691	—	—
GraphSAGE-GCN	0.742	0.772	**0.908**	0.930	0.465	0.500
GraphSAGE-mean	0.778	0.820	0.897	0.950	0.486	0.598
GraphSAGE-LSTM	0.788	0.832	**0.907**	**0.954**	0.482	**0.612**
GraphSAGE-pool	**0.798**	**0.839**	0.892	0.948	**0.502**	0.600
% gain over feat.	39%	46%	55%	63%	19%	45%

Micro-F1 (i.e. F1 scores computed using global TPs, FPs, and FNs) scores of all models (unsupervised and supervised), across each dataset.

Results



Training and inference time



Runtime and Micro F1 against a sample size (S1 = S2) of a 2-hop neighborhood (K = 2).

Results
(summary)

1. GraphSAGE Pool, LSTM, and Mean models consistently outperformed all the baselines. On average, they were all better than GraphSAGE-GCN.
    a. Concatenation of the embeddings improved performance.

2. Both GraphSAGE-LSTM and GraphSAGE-pool were better on average across all experimental settings. They were marginally better than the mean embedding variant.

3. All GraphSAGE variants were significantly faster at test time than their shallow embedding counterparts.
    1. Mini-batching and sampling the neighborhood nodes greatly increased speed of training and inference, without affecting performance.

4. Sampling with depths (K) > 2 or increasing the sampling size did not provide a significant gain in performance.

Comments

1. The relationship with the WL-Isomorphism test is interesting and proved to be significant (as in Xu, Hu 2019). Maybe it could have been useful in motivating and explaining generalization across graphs?

2. Why was an LSTM picked as an aggregator? Why wasn't the pooling aggregator (which is symmetric) better than the LSTM aggregator?

3. Comparison between skip and non-skip variants of each GraphSAGE model.

4. The F1-score equally weighs Precision and Recall. Maybe separate precision and recall values would have been helpful in comparing the top 3 GraphSAGE variants?

5. The random classifier already had an F1-score of 0.396 on the PPI dataset.

Future Directions

1. Using attention to aggregate node features

2. Non-uniform sampling schemes. Sampling schemes that can be learnt.

3. Exploring the relationship with the WL-isomorphism test.

4. How can we make inductive models explainable? What rules are they relying on? How biased are they towards sensitive attributes of minorities?
    1. Can we separate the positional vs. structural (role played by a node) information used to generate the embeddings? [Teru, Hamilton Neurips 19]

Thank You