Inductive Representation Learning on Large Graphs

William L. Hamilton*, Rex Ying*, Jure Leskovec
(with many thanks to Dan Jurafsky, Alex Ratner, and Bryan He)
Inductive Representation Learning on Large Graphs

William L. Hamilton*, Rex Ying*, Jure Leskovec
(with many thanks to Dan Jurafsky, Alex Ratner, and Bryan He)

https://github.com/williamleif/GraphSAGE
Representation learning on graphs

\[ \in \mathbb{R}^d \]

node classification, link prediction, question answering etc.
Embed nodes **inductively** (across graphs)

Inductive node embedding  

\[ \text{generalize to entirely unseen graphs} \]

e.g., train on protein interaction graph from model organism A and generate embeddings on newly collected data about organism B.
Embed nodes inductively (in evolving graphs)

- Train at $t=0$
- New node arrives at $t=1$
- Generate embedding for new node at $t=2$
GraphSAGE: The basic idea

INPUT GRAPH

TARGET NODE

AGGREGATE

BATCH OF NODES
GraphSAGE: Neighborhood aggregation

TARGET NODE

INPUT GRAPH

AGGREGATE

Aggregator options:

mean: simple, element-wise mean

LSTM: apply LSTM to random permutation of the points

max-pooling: \( \max(\{\sigma(Wh_u + b), \forall u \in \mathcal{N}(v)\}) \)
Performance: Improvement over baselines

~40% average improvement over raw features.
GraphSAGE: Neighborhood aggregation

INPUT GRAPH

TARGET NODE

TARGET NODE

INPUT GRAPH

AGGREGATE

Aggregator options:

**mean:** simple, element-wise mean

**LSTM:** apply LSTM to random permutation of the points

**max-pooling:** \( \max(\{\sigma(Wh_u + b), \forall u \in \mathcal{N}(v)\}) \)
On average, the trainable LSTM/pool aggregators give a gain of ~6.5% compared to the GCN-based aggregator.
Performance: Scalable to massive graphs

**Task:** Retrieve content that users are likely to “pin” (prior to ranking). 1+ billion possible candidates.

**Results:** GraphSAGE gives 350% improvement over PageRank-like baseline (36% vs. 11% prob@k).
Implementation challenge: Dynamic batch sizes

Every node has unique compute graph. Can’t batch on GPU!
Solution: Sample fixed-sized neighborhoods

Compute graphs have same structure = efficient GPU batching
Performance: Impact of sampling

Diminishing returns for sampling larger neighborhoods

(Results on citation data using GraphSAGE-mean)
Implementation sketch: Overview

Algorithm 2: GraphSAGE minibatch forward propagation algorithm

**Input:** Graph $G(V, E)$;
  - input features $\{x_v, \forall v \in V\}$;
  - depth $K$;
  - weight matrices $W^k, \forall k \in \{1, \ldots, K\}$;
  - non-linearity $\sigma$;
  - differentiable aggregator functions $AGGREGATE_k, \forall k \in \{1, \ldots, K\}$;
  - neighborhood sampling functions $\mathcal{N}_k : v \rightarrow 2^V, \forall k \in \{1, \ldots, K\}$

**Output:** Vector representations $z_v$ for all $v \in B$

1. $B^K \leftarrow B$;
2. for $k = K \ldots 1$ do
3. \hspace{1em} $B^{k-1} \leftarrow B^k$;
4. \hspace{1em} for $u \in B^k$ do
5. \hspace{2em} $B^{k-1} \leftarrow B^{k-1} \cup \mathcal{N}_k(u)$;
6. \hspace{1em} end
7. end
8. $h^0_u \leftarrow x_v, \forall v \in V^0$;
9. for $k = 1 \ldots K$ do
10. \hspace{1em} for $u \in B^k$ do
11. \hspace{2em} $h^k_{\mathcal{N}(u)} \leftarrow AGGREGATE_k(\{h^k_{u'}, \forall u' \in \mathcal{N}_k(u)\})$;
12. \hspace{2em} $h^k_u \leftarrow \sigma \left(W^k \cdot \text{CONCAT}(h^{k-1}_u, h^k_{\mathcal{N}(u)})\right)$;
13. \hspace{2em} $h^K_u \leftarrow h^K_u / \|h^K_u\|_2$;
14. end
15. end
16. $z_u \leftarrow h^K_u, \forall u \in B$
Implementation sketch: Overview

TARGET NODE

INPUT GRAPH
Implementation sketch: Overview

samples, support_sizes = sample(input_nodes, ...)
https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
Implementation sketch: Overview

```
samples, support_sizes = sample(input_nodes, ...)
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
samples, support_sizes = sample(input_nodes, ...)
https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
Implementation sketch: Overview

\[
samples, support_sizes = \text{sample}(input_nodes, \ldots)
\]
https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254

\[
\text{embeddings} = \text{aggregate}(samples, \ldots)
\]
https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
samples, support_sizes = sample(input_nodes, ...)
https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
embeddings = aggregate(samples, ...)
https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
    return samples, support_sizes

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
    neighs = adj_mat.embedding_lookup(samples)
    neighs = shuffle(neighs, axis=1)  # shuffle columns
    samples = neighs[:, :num_samples].flatten()
    return samples, support_sizes

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1)  # shuffle columns
        samples = neighs[:,:num_samples].flatten()
    return samples, support_sizes

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1)  # shuffle columns
        samples = neighs[:, :num_samples].flatten()
    return samples, support_sizes

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1)  # shuffle columns
        samples = neighs[:, :num_samples].flatten()
    return samples, support_sizes

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
Implementation sketch: Sample stage

def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1) # shuffle columns
        samples = neighs[:, :num_samples].flatten()
    return samples, support_sizes

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
Implementation sketch: Sample stage

```
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
    neighs = adj_mat.embedding_lookup(samples)
    neighs = shuffle(neighs, axis=1)  # shuffle columns
    samples = neighs[:, :num_samples].flatten()
    return samples, support_sizes
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
Implementation sketch: Sample stage

```python
@nb.njit(parallel=True)
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1)  # shuffle columns
        samples = neighs[:, :num_samples].flatten()
    return samples, support_sizes
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1)  # shuffle columns
        samples = neighs[:, :num_samples].flatten()
    return samples, support_sizes

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1)  # shuffle columns
        samples = neighs[:, :num_samples].flatten()
    return samples, support_sizes

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1)  # shuffle columns
        samples = neighs[:, :num_samples].flatten()
    return samples, support_sizes

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
def sample(input_nodes, num_layers, sample_sizes, adj_mat):
    samples = input_nodes
    support_size = 1
    support_sizes = []
    for i in range(num_layers):
        support_size *= sample_sizes[i]
        support_sizes.append(support_size)
        neighs = adj_mat.embedding_lookup(samples)
        neighs = shuffle(neighs, axis=1)  # shuffle columns
        samples = neighs[:, :num_samples].flatten()
    return samples, support_sizes

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L254
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
Implementation sketch: Aggregation stage

```python
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
Implementation sketch: Aggregation stage

```python
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)

    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)

    return messages

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
```
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples) - 1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
Implementation sketch: Aggregation stage

```python
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages
```

[GitHub link](https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278)
Implementation sketch: Aggregation stage

```python
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages
```

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
Implementation sketch: Aggregation stage

def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
Implementation sketch: Aggregation stage

def aggregate(samples, support_sizes, feature_matrix, embed_dim):
    messages = feature_matrix.embedding_lookup(samples)
    messages = messages.reshape(batch_size, support_sizes[-1], feature_dim)
    for i in range(len(samples)-1, 1, -1):
        messages = aggregate_func(messages, embed_dim)
    messages = messages.reshape(batch_size, support_sizes[-2], embed_dim)
    return messages

https://github.com/williamleif/GraphSAGE/blob/master/graphsage/models.py#L278
Additional complexities.....

TARGET NODE

INPUT GRAPH

AGGREGATE

Various unsupervised vs. supervised loss functions

“Skip-connections”

Efficient data I/O.