A Deep Learning based Community Detection approach

GIANCARLO SPERLI’
CINI – ITEM NATIONAL LABORATORY
Social Network Analysis

Community Detection
Similar users over the networks, a subset of strictly connected nodes.

Why?
- Viral Marketing, brand awareness
- Advertising Targeting
- Recommendation

Contribute:
A novel algorithm to detect communities using deep learning approaches on large datasets
Deep Learning

Why?

- Train a network topology, predicting communities according to relationships that may change over the time.
- All existing approaches are limited

DeepWalk

Modularity Based

Graph $G = (V,E)$

$$b_{ij} = a_{ij} - \frac{k_i k_j}{2m}$$

1. Adjacency matrix
2. Expected number of edges

K-means
Proposed Approach with CNN

Traditional Application
- Object detection and image recognition

Goal
- Apply the same deep learning approach to social networking, for detecting communities

Input:
- Matrix with single pixel values
- 3 sub-matrices for RGB images
- 1 matrix for B/W image

Components:
- Convolution Layer
- Max-Pooling layer
- Fully Connected Layer

Output:
- Probability that input image belongs to a defined class
A Deep Learning based Community Detection approach

**CNN and Social Networks**

**Input Construction**

**Ground Truth**

Output

Probability that input node belongs to a community
The dimensional challenge

- Image size: 1024 x 1024 pixel
  - Matrix values: 1,048,576 x 32 bit = 4,2 MB

- Graph Size: 1,000,000 nodes
  - Matrix values: 1,000,000,000,000 x 32 bit = 4000 GB
Primitives for sparse Matrices:
- Storing
- Slicing
- Densifying

Conv2D doesn’t work with Sparse Matrices

Graph Size:
1,000,000 nodes

Non-Zero values with
$10^{-5}$ sparsity grade:

1,000,000 x 32 bit = 120 MB

Dense | Sparse
------|------
4000 GB | 120 MB

Proposed Algorithm

Re-implementing tf.Conv2D in a SparseConv2D
exploiting sparse structure of network adjacency matrices
A Deep Learning based Community Detection approach

Sparse Convolution

Kernel Initialization

<table>
<thead>
<tr>
<th>0.21</th>
<th>0.78</th>
<th>2.32</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.9</td>
<td>3.25</td>
<td>1.83</td>
</tr>
<tr>
<td>2.33</td>
<td>6.72</td>
<td>2.15</td>
</tr>
</tbody>
</table>

Sliced and Reshaped Node “n”
A Deep Learning based Community Detection approach

Sparse Convolution

Kernel Initialization

Sliced and Reshaped Node “n”
A Deep Learning based Community Detection approach

Sparse Convolution

Kernel Initialization

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.21</td>
<td>0.78</td>
<td>2.32</td>
</tr>
<tr>
<td>1.9</td>
<td>3.25</td>
<td>1.83</td>
</tr>
<tr>
<td>2.33</td>
<td>6.72</td>
<td>2.15</td>
</tr>
</tbody>
</table>

Sliced and Reshaped Node “n”

Densifying

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>NZV</td>
</tr>
</tbody>
</table>
Sparse Convolution

Kernel Initialization

Sliced and Reshaped Node “n”

Densifying

Kernel

Feature Value

A Deep Learning based Community Detection approach
A Deep Learning based Community Detection approach

Sparse Convolution

Kernel Initialization

Sliced and Reshaped Node “n”

Densifying

Kernel

Feature Value

Scrolling Window

Next Steps

Only around non-zero values
Sparse Convolution

Kernel Initialization

Sliced and Reshaped Node “n”

Densifying

Kernel

Final Sparse Feature Map
For each kernel

Scrolling Window

Next Steps

Only around non-zero values

A Deep Learning based Community Detection approach
Sparse Max Pooling

Sparse Feature Map
Sparse Max Pooling

A Deep Learning based Community Detection approach
Sparse Max Pooling

Sparse Feature Map

Densifying Max Pooling Window

Max (0,0,F1,F2)

MV
Sparse Max Pooling

Sparse Feature Map

Densifying Max Pooling Window

Max (0,0,F1,F2)

Next Steps

MV

Analyze only 2x2 windows with at least 1 F value

A Deep Learning based Community Detection approach
A Deep Learning based Community Detection approach

Sparse Max Pooling

Sparse Feature Map

Densifying Max Pooling Window

Max (0,0,F1,F2)

Analyze only 2x2 windows with at least 1 F value

Next Steps

Next Kernels and Nodes

Reduced Feature Map For Node ”n”
Another Approach: Divide et impera

Kernel Initialization

Sliced and Reshaped Node “n”
Another Approach: Divide et impera

Kernel Initialization

| 0.21 | 0.78 | 2.32 |
| 1.9  | 3.25 | 1.83 |
| 2.33 | 6.72 | 2.15 |

Sliced and Reshaped Node “n”

Densifying

| 0  | 0  | 0  | 0  |
| 0  | 0  | 0  | 0  |
| 0  | 0  | 0  | 0  |
| 0  | 0  | 0  | 0  |
Another Approach: Divide et impera

Kernel Initialization

Sliced and Reshaped Node “n”

Densifying

Kernels

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0,21</td>
<td>0,78</td>
<td>2,32</td>
</tr>
<tr>
<td>1,9</td>
<td>3,25</td>
<td>1,83</td>
</tr>
<tr>
<td>2,33</td>
<td>6,72</td>
<td>2,15</td>
</tr>
</tbody>
</table>

A Deep Learning based Community Detection approach
Another Approach: Divide et impera

Kernel Initialization

Sliced and Reshaped Node “n”

Densifying

Kernels

0.21 0.78 2.32
1.9 3.25 1.83
2.33 6.72 2.15

A Deep Learning based Community Detection approach

Section
Dense Feature Maps

Section
Sparse Feature Maps
Another Approach: Divide et impera

Kernel Initialization

Sliced and Reshaped Node “n”

Densifying

Convolution with all Matrix sections

Kernels

Section Dense Feature Maps

Section Sparse Feature Maps

TensorFlow™ Tf.conv2D

A Deep Learning based Community Detection approach
A Deep Learning based Community Detection approach

Another Approach: Divide et impera

Kernel Initialization

Sliced and Reshaped Node “n”

Densifying

Kernels

Convolution with all Matrix sections

Final Sparse Feature Maps

Section Dense Feature Maps

Section Sparse Feature Maps

Next Steps
A Deep Learning based Community Detection approach

System Overview
Experimental Protocol

Convolution Time Tests
- SparseConv2D vs Divide et Impera
- Artificial Large Datasets with different dimensions and sparsity

Training Tests
- SparseConv2D loss trend (kernels, levels, optimizer and rate)
- Email Dataset to test whole network

Accuracy Tests
- Using 3 different training
- Evaluation datasets generated from Email Dataset

Artificial Datasets
- Sparse Random Matrices
- Variable Dimensions
- Variable Sparsity

Large Dataset
- Dimension: 138,000 nodes
- Sparsity: $7,1 \times 10^{-6}$

Email Dataset
- Dimension: 1000 nodes
- Edges: 25000
Convolution Time Tests

- 138,000 nodes, $10^{-6}$
- SparseConv: 12 minutes
- Divide et impera: 602 minutes
Training

A Deep Learning based Community Detection approach
Entries with Hop Count

\[ entry = e^{\sigma \cdot (1-s)} \]

- \( s \): hop count of node \( n' \) to node \( n \)
- \( \sigma \): attenuation factor
- \( s_0 \): hop count threshold
Evaluation Tests

Accuracy prediction according to Deleted Edges

- $S_0 = 1$
- $S_0 = 2$
- $S_0 = 3$