Graph-based Methods in Pattern Recognition and Document Image Analysis (GMPRDIA)

Tutorial at the 15th IAPR International Conference on Document Analysis and Recognition (ICDAR2019)
Saturday 21st September 2019, University of Technology Sydney (UTS)

GMPRDIA 2019
Organizing Committee and Speakers

Muhammad Muzzamil LUQMAN
L3i, La Rochelle University, France

Pau RIBA
CVC, Barcelona, Spain

Anjan DUTTA
University of Exeter, UK

Muhammad Muzzamil LUQMAN  
L3i, La Rochelle University, France

- Research Scientist (Permanent)
- Ph.D. in Computer Science from François Rabelais University of Tours (France) and Autonoma University of Barcelona (Spain).
- Ph.D. thesis titles “Fuzzy Multilevel Graph Embedding for Recognition, Indexing and Retrieval of Graphic Document Images”.
- Research interests
  - Structural Pattern Recognition
  - Document Image Analysis
  - Camera-Based Document Analysis and Recognition
  - Graphics Recognition
  - Artificial Intelligence / Machine Learning

http://pageperso.univ-lr.fr/muhammad_muzzamil.luqman
Pau RIBA  
CVC, Barcelona, Spain

- Ph.D. student in Computer Science from the Computer Vision Center (CVC, Barcelona) under supervision of Josep Llados (since October 2016)
- Research interests:
  - Graph-based representation for visual objects
  - Graph-based algorithms for solving various tasks in Computer Vision Pattern Recognition and Machine Learning
  - Machine Learning
- [http://www.cvc.uab.es/people/priba/](http://www.cvc.uab.es/people/priba/)
  - [https://github.com/priba](https://github.com/priba)
GMPRDIA 2019
Organizing Committee and Speakers

Anjan DUTTA
University of Exeter, UK

- Lecturer (Assistant Professor) in Computer Vision & Machine Learning
- Until July 2019, he was a Marie-Curie postdoctoral fellow under the P-SPHERE project at the Computer Vision Centre, Barcelona, Spain.
- Ph.D. in Computer Science from the Universitat Autònoma de Barcelona (UAB) in the year of 2014.
- Ph.D. thesis titled “Inexact Subgraph Matching Applied to Symbol Spotting in Graphical Documents”
- Research interests
  - graph-based representation for visual objects
  - graph-based algorithms for solving various tasks in Computer Vision, Pattern Recognition and Machine Learning
- https://sites.google.com/site/2adutta/home
Question 1/3
How a graph is represented in computer memory?

- Type of graph representation in computer memory
- There are two ways:
  - Sequential representation
  - Linked representation
Question 1/3
How a graph is represented in computer memory?

Sequential representation

- Adjacency matrix

\[ A = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
\end{bmatrix} \]
Question 1/3
How a graph is represented in computer memory?

Sequential representation

- Adjacency matrix

![Graph Representation](image)

Adjacency matrix:

\[
A = \begin{bmatrix}
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 \\
\end{bmatrix}
\]
Question 1/3
How a graph is represented in computer memory?

Sequential representation

- Incidence matrix

\[ I = \begin{bmatrix}
  e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\
  0 & 0 & 0 & 0 & 1 & 1 \\
  1 & 0 & 0 & 1 & 1 & 1 \\
  1 & 1 & 0 & 0 & 0 & 0 \\
  0 & 1 & 1 & 0 & 0 & 0 \\
  0 & 0 & 1 & 0 & 0 & 1 
\]
Question 1/3
How a graph is represented in computer memory?

Linked representation

- Adjacency list

```
A =
1 2 5
2 1 3 5
3 2 4
4 3 5
5 1 2 4
```
How graphs are stored on disk?

SSGCI competition (http://ssgci.univ-lr.fr)

In which languages can I program/code a graph-based method?

**Matlab**

MatlabBGL, Graph and Network algorithms, GAIMC, ...

**Python**

Networkx, igraph, ...

**C/C++**

Boost Graph Library, ...

and many others ...
Saturday 21st September 2019
09h00 – 12h30

Part-1
- A historic perspective of graph-based methods in PR & DIA
- Neural Networks on graphs and modern trends in graph-based PR & DIA

Coffee break (10h30 - 11h00)

Part-2
- Applications of Graph Neural Networks
  - Learning Graph Distances
  - Table Detection
- Hands-on
  - Deep Graph Library
Structural and Statistical Pattern Recognition

<table>
<thead>
<tr>
<th></th>
<th>Structural</th>
<th>Statistical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data structure</td>
<td>symbolic data structure</td>
<td>numeric feature vector</td>
</tr>
<tr>
<td>Representational strength</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Fixed dimensionality</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sensitivity to noise</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Efficient computational tools</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
How images (and/or other types of content) are represented by graphs?
A graph $G = (V, E)$ is a mathematical structure for representing relationships.

A graph $G = (V, E)$ consists of a set of nodes $V$ connected by edges $E$. 
Directed and Undirected Graph

Directed Graph

Undirected Graph
Attributed Graph

An attributed Graph is a 4-tuple $G = (V, E, \alpha, \beta)$

- Set of nodes $V$
- Set of edges $E \subseteq V \times V$
- Node attribute function $\alpha : V \rightarrow L_V$
- Edge attribute function $\beta : E \rightarrow L_E$
Graph Representation: Issues to Consider

Graph representation of objects depends on:

1. Problem definition
2. Type of solution / methodology
3. Stability and noise tolerance
Discriminant units of information in an underlying image for representing it by a graph

- Critical Points
- Line Segments
- Homogeneous Regions
- Keypoints
- Convex Regions
- etc.
Critical Points

- Critical points from skeleton or edge analysis as nodes.

- Type of edges:
  - Adjacency
  - Proximity
  - k-NN
  - Delaunay triangulation

- Example
  - Symbol spotting by hashing serialized subgraphs.
  - Critical points as nodes and their connections as edges.

Line Segments

- Line segments from skeleton or edge analysis as nodes.
- Type of edges:
  - Adjacency
  - Proximity
  - k-NN
  - Delaunay triangulation
- Example
  - Subgraph matching applied to symbol spotting.
  - Each line segment as a node and up to 3 nearest neighbors are joined to form edges.

Homogeneous Regions

- Regions either existing or generated by a preprocessing stage as nodes.
- Type of edges:
  - Adjacency
  - Proximity
  - Delaunay triangulation
- Example
  - SSGCI competition, ICPR 2016.
  - RAG of cartoon characters
  - Subgraph spotting
Keypoints

- Detected keypoints using some off-the-shelf algorithm as nodes.
- Type of edges:
  - Proximity
  - k-NN
  - Delaunay triangulation
- Example
  - Symbol recognition.
  - Shape context of detected SIFT interest points.

Example: Skeleton Graph

- Skeleton graph
- Each junction or end point as a node of the graph
- Edges are created following the skeleton

Figure credit: Bai and Latecki PAMI 2008

Example: Region Adjacency Graph

- Region adjacency graph
- Each white region as a node in the graph
- Each pair of adjacent nodes is connected by an edge

Figure credit: Le Bodic et al 2012

Example: Graph of convexities

- Convex part segmentation
- Each convex part as node
- Nearest nodes are joined as edges


Figure credit: Riba et al, PRL 2017
Example: Graph of critical points

- Critical points, grid etc as nodes.
- Adjacent nodes on the writing are joined.
- Normalized coordinates as node attributes.

- Critical points as nodes.
- Adjacent nodes on the symbol are joined.
- Coordinate as node attributes.
- Line type as edge attributes.

Histograph dataset (http://www.histograph.ch/)
GREC dataset (http://www.fki.inf.unibe.ch/databases)
Example: Vecto-Quad graph representation

- Graph representation developed for line drawings
- Each node in the graph represents a line in underlying image
- Thin lines are termed as vectors
- Thick lines or filled shapes are termed as quadrilaterals
- Connections between the vectors/quadrilaterals are represented by edges
- Attributes on nodes as well as edges


Example: Vecto-Quad graph representation

- Vectors and Quadrilaterals representation well adapted to the underlying line-drawing images


Example: Vecto-Quad graph representation

- Graph-based representations have built-in rotation invariance


Example: MSER-regions based graph representation

- Graph representation developed for colored comic images
- Each node in graph represents an MSER region in underlying image
- Spatial relations between MSER regions are represented by edges in graph
- Attributes on nodes as well as edges

Thanh-Nam Le, Muhammad Muzzamil Luqman, Jean-Christophe Burie, Jean-Marc Ogier: Content-based comic retrieval using multilayer graph representation and frequent graph mining. ICDAR 2015: 761-765

Example: MSER-regions based graph representation

- Multilayer graph representation
  - Color layer
  - Hu-moments layer
  - Compactness layer

Thanh-Nam Le, Muhammad Muzzamil Luqman, Jean-Christophe Burie, Jean-Marc Ogier: Content-based comic retrieval using multilayer graph representation and frequent graph mining. ICDAR 2015: 761-765

Learning Graph Representation

- Learning graph that best represent an image for matching to another relevant image
- Fully connected graph of detected key points
- Learning node and edge parameters that prioritize a set of nodes for a particular structure

Figure credit: Cho et al 2013

How we can/used to solve Pattern Recognition problems using graphs?
A very general overview of historical evolution of graph-based solutions to Pattern Recognition

- Graph matching (isomorphism)
  
  [Messmer, 1995] [Sonbaty and Ismail, 1998]

- Graph Edit Distance (GED)
  
  [Bunke and Shearer, 1998] [Neuhaus and Bunke, 2006]

- Graph EMbedding (GEM)
  
  [Luqman et al., 2009] [Sidere et al., 2009] [Gibert et al., 2011]
Graph matching

Finding matches (isomorphism) between two graphs.

\[ X_{ia} = 1 \text{ if node } i \text{ in } G \text{ corresponds to node } a \text{ in } G' \]

\[ X_{ia} = 0 \text{ otherwise} \]
Graph matching

Maximizing the matching score $S$

\[
X^* = \arg \max_X S(g, g', X)
\]

\[
\text{s.t. } \begin{cases} 
X \in \{0, 1\}^{nn'} \\
\sum_{i=1}^n X_{ia} \leq 1, \sum_{a=1}^{n'} X_{ia} \leq 1 
\end{cases}
\]
Graph matching

How to measure the matching score $S$?

- Each node and each edge has its own attribute
- Node similarity function

$S_V(a_i, a'_a)$
Graph matching

How to measure the matching score $S$?

$$S(g, g', X) = \sum_{X_{ia}=1} S_V(a_i, a'_a) + \sum_{X_{ia}=1, X_{jb}=1} S_E(a_{ij}, a'_{ab})$$

- Sum of $S_V$ and $S_E$ values for the assignment $X$. 
Graph matching

How to measure the matching score $S$?

- $X_{ia} = 1$ if node $i$ in $G$ corresponds to node $a$ in $G'$
- $X_{ia} = 0$ otherwise
Advances in graph matching

● Quadratic assignment problem
  ○ NP-hard, thus exact solution is infeasible

● Advances in approximate (inexact) algorithms
  ○ Error-tolerant (inexact) graph matching
  ○ Relaxation and Projection
Graph edit distance

- A measure of similarity between two graphs.
- Node and edge insertion, deletion, substitution.
- Summation of the edit costs

\[
\text{GED}(G, G') = \min_{(e_1, \ldots, e_k) \in \mathcal{P}(G, G')} \sum_{i=1}^{k} c(e_i)
\]

Graph embedding

By mapping a high dimensional graph into a point in suitable vector space, graph embedding permits to perform the basic mathematical computations which are required by various statistical pattern recognition techniques, and offers interesting solutions to the problems of graph clustering and classification.

Graph embedding

Graph embedding

Graph probing based methods

[Wiener, 1947] [Papadopoulos et al., 1999] [Gibert et al., 2011] [Sidere et al., 2012]
Graph embedding

Dissimilarity based methods

[Pekalska et al., 2005] [Ferrer et al., 2008] [Riesen, 2010] [Bunke et al., 2011]
Graph embedding

Graph feature extraction based methods

- Node information
- Edge information
- Structure
- Topology
- Geometry
- Node/Edge neighborhood information


Graph embedding

Graph feature extraction based methods - FMGE

Multilevel analysis of graph

<table>
<thead>
<tr>
<th>Graph Level Information</th>
<th>Structural Level Information</th>
<th>Elementary Level Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>[macro details]</td>
<td>[intermediate details]</td>
<td>[micro details]</td>
</tr>
</tbody>
</table>

- Graph order
- Graph size
- Node degree
- Homogeneity of subgraphs in graph
- Node attributes
- Edge attributes

<table>
<thead>
<tr>
<th>Graph order</th>
<th>Graph size</th>
<th>Embedding of node degree</th>
<th>Embedding(s) of subgraph(s) homogeneity</th>
<th>Embedding(s) of node attribute(s)</th>
<th>Embedding(s) of edge attribute(s)</th>
</tr>
</thead>
</table>

Graph embedding

- Graph kernels can be intuitively understood as functions measuring the similarity of pairs of graphs
- Graph kernels allow kernelized learning algorithms such as support vector machines to work directly on graphs, without having to do feature extraction to transform them to fixed-length, real-valued feature vectors
- Laplacian Graph Kernel, Treelet Kernel, Random Walk Kernel, Graphlet Kernel, etc.

Donatello Conte, Jean-Yves Ramel, Nicolas Sidere, Muhammad Muzzamil Luqman, Benoit Gaüzère, Jaume Gibert, Luc Brun, Mario Vento: A Comparison of Explicit and Implicit Graph Embedding Methods for Pattern Recognition. GbRPR 2013: 81-90
A very general overview of historical evolution of graph-based solutions to Pattern Recognition

- Graph matching (isomorphism)
  
  [Messmer, 1995] [Sonbaty and Ismail, 1998]

- Graph Edit Distance (GED)
  
  [Bunke and Shearer, 1998] [Neuhaus and Bunke, 2006]

- Graph EMbedding (GEM)
  
  [Luqman et al., 2009] [Sidere et al., 2009] [Gibert et al., 2011]
What kind of Pattern Recognition problems have been solved by using graphs?
Graph similarity
Graph classification
Graph clustering
Graphics detection / localization / recognition / classification / clustering / spotting
Chemical molecules recognition / classification / clustering
Fingerprint recognition
Handwriting recognition
Signature recognition / verification
Document image segmentation / classification / clustering / indexing
QBE and CBIR in document images
Focused retrieval in document images
etc.
Subgraph Spotting through Explicit Graph Embedding: An Application to Content Spotting in Graphic Document Images

Automatic indexing of comic page images for query by example based focused content retrieval

Content-based Comic Retrieval Using Multilayer Graph Representation and Frequent Graph Mining

How has the success story of deep learning influenced the graph-based methods of Pattern Recognition?
Success story of deep learning

Noun Phrase
- Article
- Noun

Verb
- Preposition

Prepositional Phrase
- Article
- Noun

Sentence
- Predicate / Verb Phrase

The dog sat beside the wall

Natural Language Processing (NLP)

Speech Data

Slide credit: Kipf et al. Deep Learning on Graphs with Graph Convolutional Networks
Evolution of deep learning

- **1958**: Perceptron - Rosenblatt
- **1959**: Backprop - Werbos
- **1982**: First NIPS
- **1987**: SVM - Vapnik
- **1995**: CNN - LeCun
- **1997**: RNN / LSTM - Schmidhuber
- **1998**: Autoencoder - LeCun, Hinton
- **1999**: ImageNet breakthrough - Krizhevsky
- **2006**: First GPU - NVIDIA
- **2010**: ImageNet - Vapnik
- **2012**: Speech Recognition - Microsoft
- **2014**: AI Research - Facebook
- **2015**: Autonomous cars - Tesla
- **2016**: Autonomous cars - Google

**Slide credit**: M. Bronstein et al. Geometrical Deep Learning, Tutorial, CVPR, 2017
CNN: LeNet 5

- 3 convolutional + 1 fully connected layer
- 1M parameters
- Training set: MNIST 70K images
- Trained on CPU
- \( \tanh \) non-linearity

CNN: AlexNet

- 5 convolutional + 3 fully connected layer
- 60M parameters
- Trained on ImageNet 1.5M images
- Trained on GPU
- ReLU non-linearity
- Dropout regularization

Convolutional neural network

- Hierarchical compositionality
- Weight sharing
- Big data
- Computational power
Traditional vs “deep” learning

- Hand crafted features → Classifier → Output
- Deep neural network → Output
a graph convolution can be generalized from a standard 2D convolution

(a) 2D Convolution. Analogous to a graph, each pixel in an image is taken as a node where neighbors are determined by the filter size. The 2D convolution takes the weighted average of pixel values of the red node along with its neighbors. The neighbors of a node are ordered and have a fixed size.

(b) Graph Convolution. To get a hidden representation of the red node, one simple solution of the graph convolutional operation is to take the average value of the node features of the red node along with its neighbors. Different from image data, the neighbors of a node are unordered and variable in size.

Fig. 1: 2D Convolution vs. Graph Convolution.
CNN: Message passing in a grid graph

- Individual message transforms
  \[ W_v h_v \]

- Sum everything up
  \[ \sum_v W_v h_v \]

- Full update
  \[ h_{v}^{t+1} = \sigma(W_0^t h_0^t + W_1^t h_1^t + \ldots + W_v^t h_v^t + \ldots + W_g^t h_g^t) \]

Animation by V. Dumoulin
Graph structured data

What if the data look like this?

or this:
Graph structured data

Real world examples:

- Social networks
- World wide web
- Protein interaction networks
- Telecommunication networks
- Knowledge graphs
- ...
Message passing on graphs

Consider this undirected graph:

Calculate update for node in green:

Update rule:

\[ h_v^{t+1} = \sigma(h_v^t W_0^t + \frac{1}{c_{vw}} \sum_{w \in N_v} h_w^t W_1^t) \]

More general or simpler function also can be chosen

2. T. Kipf, M. Welling, Semi-Supervised Classification with Graph Convolutional Networks. ICLR, 2017
Several iteration of message passing

Node and edge updation:

Initial stage:

Final stage:
Graph wise classification
Node wise classification

Figure credit: Shotton et al IJCV 2007
Neural Message Passing

Message function:

\[ m_v^{t+1} = \sum_{w \in N_v} M(h_v^t, h_w^t, e_{vw}) \]

Update function:

\[ h_v^{t+1} = U(h_v^t, m_v^{t+1}) \]

Readout function:

\[ \hat{y} = R(\{ h_v^T \mid v \in G \}) \]
Running Example
Message passing

Message function:

\[ M(h^t_v, h^t_w, e_{vw}) \]
Message passing

Message function:

\[ M(h^t_v, h^t_w, e_{vw}) \]
Message passing

Message function:

\[ M(h_v^t, h_w^t, e_{vw}) \]
Message passing

Message function:

\[ M(h_v^t, h_w^t, e_{vw}) \]
Message passing

Message function:

\[ m_{v}^{t+1} = \sum_{w \in \mathcal{N}_v} M(h_v^t, h_w^t, e_{vw}) = \]

\[ \quad = \text{message} \]
Message passing

Example message function:

\[ M(h_v^t, h_w^t, e_{vw}) = \frac{A_{vw}}{\sqrt{\text{deg}(v) \text{deg}(w)}} h_w^t \]

where \( h_v \) is the hidden state of the node \( v \), \( e_{vw} \) is edge feature of \( vw \), and \( A_{vw} \) is a learned matrix.

Message passing

Update function:

$$U(h^t_v, m^{t+1}_v)$$
Message passing

Update function:
Message passing

Example update function:

\[ U_t(h_v^t, m_v^{t+1}) = \sigma(W_t^t m_v^{t+1}) \]

where \( W_t \) are learned matrices one for each time step, \( \sigma \) is a non-linearity function such as ReLU (Rectified Linear Unit).

Readout

Readout function:

\[ \hat{y} = R(\{ h_v^T \mid v \in G \}) \]

Example:

\[ R = f(\sum_v h_v^t) \]

This readout function sums the current hidden states of all the nodes and computes an output through a learnable neural network \( f \).
Convolutional Networks on Graphs

- **Message Function**
  \[ m_{v}^{t+1} = M(h_{v}^{t}, h_{w}^{t}, e_{vw}) = (h_{w}^{t}, e_{vw}) \]

- **Update Function**
  \[ h_{v}^{t+1} = U_{t}(h_{v}^{t}, m_{v}^{t+1}) = \sigma(H_{t}^{deg(v)} m_{v}^{t+1}) \]

- **Readout Function**
  \[ \hat{y} = R = f(\sum_{v,t} \text{softmax}(W_{t}h_{v}^{t})) \]

where \((.,.)\) denotes concatenation, \(H_{t}^{N}\) are learned matrices one for each time step \(t\) and degree edge label, \(f\) is a neural network and \(\sigma\) is a non-linearity function such as ReLU.

Gated Graph Sequence Neural Networks

- **Message Function**
  \[ m_{v}^{t+1} = M(h_{v}^{t}, h_{w}^{t}, e_{vw}) = A_{e_{vw}}h_{w}^{t} \]

- **Update Function**
  \[ h_{v}^{t+1} = U_{t}(h_{v}^{t}, m_{v}^{t+1}) = \text{GRU}(h_{v}^{t}, m_{v}^{t+1}) \]

- **Readout Function**
  \[ \hat{y} = R = \sum_{v} \sigma(i(h_{v}^{t}, h_{v}^{0}) \odot (j(h_{v}^{t}))) \]

where \( A_{e_{vw}} \) is a learned matrix one for each discrete edge label, \text{GRU} is Gated Recurrent Unit, \( i, j \) are neural networks and \( \odot \) is element wise multiplication, \( \sigma \) is a non-linearity function such as ReLU.
\[ z_t = \sigma (W_z \cdot [h_{t-1}, x_t]) \]

\[ r_t = \sigma (W_r \cdot [h_{t-1}, x_t]) \]

\[ \tilde{h}_t = \tanh (W \cdot [r_t \ast h_{t-1}, x_t]) \]

\[ h_t = (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t \]
Interaction Networks

- **Message Function**
  \[ m_v^{t+1} = M(h_v^t, h_w^t, e_{vw}) = f(h_v^t, h_w^t, e_{vw}) \]

- **Update Function**
  \[ h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) = g(h_v^t, x_v, m_v^{t+1}) \]

- **Readout Function**
  \[ \hat{y} = R = f(\sum_v h_v^t) \]

where \( f, g \) represent neural networks, \((.,.,.)\) denotes concatenation, \( x_v \) is an external vector representing some outside influence to the node \( v \).

Molecular Graph Convolutions

- Message Function
  \[ m_{v}^{t+1} = M(h_{v}^{t}, h_{w}^{t}, e_{vw}) = e_{vw}^{t} \]

- Update Function
  \[ h_{v}^{t+1} = U_{t}(h_{v}^{t}, m_{v}^{t+1}) = \alpha(W_{1}(\alpha(W_{0}h_{v}^{t}), m_{v}^{t+1})) \]

- Readout Function
  \[ \hat{y} = R = \alpha(W_{4}(\alpha(W_{2}, e_{vw}^{t}), \alpha(W_{3}(h_{v}^{t}, h_{w}^{t})))) \]

where \((.,.)\) denotes concatenation, \(W_{i}\) are learned weight matrices, \(\alpha\) is the ReLU activation.

Convolutional and Locally Connected Neural Networks

- **Message Function**
  \[ m^{t+1}_v = M(h^t_v, h^t_w, e_{vw}) = C_{vw}^t h^t_w \]

- **Update Function**
  \[ h^{t+1}_v = U_t(h^t_v, m^{t+1}_v) = \sigma(m^{t+1}_v) \]

where \( C_{vw} \) are parameterized by the eigenvectors of the graph Laplacian \( L \) and the other parameters of the model, \( \sigma \) is a non-linearity function such as ReLU.

1. Defferrard et al., Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, NIPS 2016.
2. Bruna et al., Spectral Networks and Locally Connected Networks on Graphs, ICLR 2014.
Graph Convolutional Networks

- **Message Function**
  \[ m_{v}^{t+1} = M(h_{v}^{t}, h_{w}^{t}, e_{vw}) = \frac{A_{vw}}{\sqrt{\text{deg}(v) \text{deg}(w)}} h_{w}^{t} \]

- **Update Function**
  \[ h_{v}^{t+1} = U_{t}(h_{v}^{t}, m_{v}^{t+1}) = \sigma(W^{t}m_{v}^{t+1}) \]

where \( A_{vw} \) is a learnable parameter, \( W^{t} \) are learned matrices one for each time step, \( \sigma \) is a non-linearity function such as ReLU

Recommended Reading

Tutorials:


List of papers:

- Defferrard et al., Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, NIPS 2016. https://arxiv.org/abs/1606.09375
- Li et al., Gated Graph Sequence Neural Networks, ICLR 2016. https://arxiv.org/abs/1511.05493
Recommended Reading

Source Code / Repositories:

- Neural Message Passing for Computer Vision: https://github.com/priba/nmp_qc
- Graph Convolutional Networks in TensorFlow: https://github.com/tkipf/gcn
- Graph Convolutional Networks in PyTorch: https://github.com/tkipf/pygcn
- PyTorch implementation of graph ConvNets: https://github.com/xbresson/graph_convnets_pytorch
- Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering: https://github.com/mdeff/cnn_graph

Other material:

- Blog post on Graph Convolutional Networks: http://tkipf.github.io/graph-convolutional-networks
Saturday 21st September 2019
09h00 – 12h30

Part-1
- A historic perspective of graph-based methods in PR & DIA
- Neural Networks on graphs and modern trends in graph-based PR & DIA

Coffee break (10h30 - 11h00)

Part-2
- Applications of Graph Neural Networks
  - Learning Graph Distances
  - Table Detection
- Hands-on
  - Deep Graph Library

Saturday 21st September 2019
09h00 – 12h30

Part-1
● A historic perspective of graph-based methods in PR & DIA
● Neural Networks on graphs and modern trends in graph-based PR & DIA

Coffee break (10h30 - 11h00)

Part-2
● Applications of Graph Neural Networks
  ○ Learning Graph Distances
  ○ Table Detection
● Hands-on
  ○ Deep Graph Library

Application:

Learning Graph Distances

Riba et al. Learning Graph Distances with Message Passing Neural Networks. In ICPR, 2018
Graph edit distance (Reminder)

- A measure of similarity between two graphs.
- Node and edge insertion, deletion, substitution.
- Summation of the edit costs

\[
\text{GED}(G, G') = \min_{(e_1, \ldots, e_k) \in P(G, G')} \sum_{i=1}^{k} c(e_i)
\]

Architecture

Graph similarity $d(G_W(x_1), G_W(x_2))$

Update
Message

Update
Message

Update
Message

Update
Message

$G_W(g_1)$

$G_W(g_2)$

$G_W(X)$ Network branch 1

$G_W(X)$ Network branch 2

W shared (siamese)

$D_W$
Graph similarity

- Hausdorff Distance

\[ H(A, B) = \max \left( \max_{a \in A} \inf_{b \in B} d(a, b), \max_{b \in B} \inf_{a \in A} d(a, b) \right) \]

- Chamfer Distance

\[ \hat{H}(A, B) = \sum_{a \in A} \inf_{b \in B} d(a, b) + \sum_{b \in B} \inf_{a \in A} d(a, b) \]

- Proposed distance.

\[ d(g_1, g_2) = \frac{\hat{H}(V_1, V_2)}{|V_1| + |V_2|} \]
Contrastive loss

Given $D_W = d(G_W(g_1), G_W(g_2))$ where $g_1$ and $g_2$ are graphs and $W$ a set of specific weights, the Loss Function is

$$l(D_W) = \frac{1}{2} \left\{ \begin{array}{ll}
D_W^2, & \text{if } Y = 1 \text{ (positive pair)} \\
\max(0, m - D_W)^2, & \text{if } Y = 0 \text{ (negative pair)}
\end{array} \right.$$  

where $m=1$ is the adaptive margin.
Datasets

Letter

- Synthetic Graphs
- 15 classes
- 750 Graphs per class
- 3 different distortion levels

George Washington

- Handwritten words
- Several graph constructions
- 105 keywords
- 4894 instances
- HistoGraph (subset for classification)
## Classification Letters

<table>
<thead>
<tr>
<th></th>
<th>LOW</th>
<th>MED</th>
<th>HIGH</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP*</td>
<td>99.73</td>
<td>94.27</td>
<td>89.87</td>
</tr>
<tr>
<td>HED†</td>
<td>97.87</td>
<td>86.93</td>
<td>79.2</td>
</tr>
<tr>
<td>Embedding‡</td>
<td>99.80</td>
<td>94.90</td>
<td>92.90</td>
</tr>
<tr>
<td>MPNN</td>
<td>95.04</td>
<td>83.20</td>
<td>72.27</td>
</tr>
<tr>
<td></td>
<td>±0.7224</td>
<td>±1.2189</td>
<td>±2.0060</td>
</tr>
<tr>
<td>Siamese MPNN</td>
<td>98.08</td>
<td>89.0136</td>
<td>74.77</td>
</tr>
<tr>
<td></td>
<td>±0.1068</td>
<td>±0.1808</td>
<td>±6.4505</td>
</tr>
<tr>
<td>Test BP</td>
<td>98.19</td>
<td>88.37</td>
<td>79.65</td>
</tr>
<tr>
<td></td>
<td>±0.1361</td>
<td>±0.41</td>
<td>±6.4345</td>
</tr>
<tr>
<td>Test HED</td>
<td>98.00</td>
<td>89.79</td>
<td>77.07</td>
</tr>
<tr>
<td></td>
<td>±0.1461</td>
<td>±0.3110</td>
<td>±5.6106</td>
</tr>
</tbody>
</table>
## Classification Histograph

<table>
<thead>
<tr>
<th>Subset</th>
<th>BP*</th>
<th>PSGE†</th>
<th>Siamese</th>
<th>MPNN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>3-NN</td>
<td>5-NN</td>
</tr>
<tr>
<td>Keypoint</td>
<td>77.62</td>
<td>80.42</td>
<td>85.31</td>
<td>82.80</td>
</tr>
<tr>
<td></td>
<td>± 1.6552</td>
<td>± 0.5600</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Projection</td>
<td>81.82</td>
<td>80.42</td>
<td>73.15</td>
<td>69.65</td>
</tr>
<tr>
<td></td>
<td>± 2.6014</td>
<td>± 1.5064</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Retrieval George Washington

<table>
<thead>
<tr>
<th>Method</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHOC*</td>
<td>64.00</td>
</tr>
<tr>
<td>BOF HMM†</td>
<td>80.00</td>
</tr>
<tr>
<td>DTW</td>
<td></td>
</tr>
<tr>
<td>DTW’01</td>
<td>42.26</td>
</tr>
<tr>
<td>DTW’08</td>
<td>63.39</td>
</tr>
<tr>
<td>DTW’09</td>
<td>64.80</td>
</tr>
<tr>
<td>DTW’16</td>
<td>68.64</td>
</tr>
<tr>
<td>Mean Ensemble BP†</td>
<td>69.16</td>
</tr>
<tr>
<td>Siamese MPNN</td>
<td>75.85±3.64</td>
</tr>
</tbody>
</table>
Application:

Table Detection by GNN

Riba et al. Table Detection in Invoice Documents by Graph Neural Networks. In ICDAR, 2019
Motivation

- Invoice Documents
- Semi-structured Documents
- Tables share Repetitive Patterns
Motivation
## Graph Construction

<table>
<thead>
<tr>
<th>Style</th>
<th>Style Description</th>
<th>Amount</th>
<th>Net Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ISSUE OCTOBER 12 1993 2006-000000 2</td>
<td>$210.00</td>
<td>$210.00</td>
</tr>
<tr>
<td>2</td>
<td>ISSUE NOVEMBER 2 1993 2077-000000 2</td>
<td>$90.60</td>
<td>$90.60</td>
</tr>
<tr>
<td>3</td>
<td>ISSUE NOVEMBER 2 1993 3047-000000 3</td>
<td>$105.00</td>
<td>$105.00</td>
</tr>
</tbody>
</table>

**Pay This Amount**: $201.60

*Terms: Net 30 days from invoice date. No cash discount allowed.**
Graph Residual Block

- Follows the idea of ResNet
- GNN layer with skip connection
- Edge weights are learned at the beginning of the block
Architecture
Objective functions

- **Node classifier**: Linear classifier with Softmax operation

- **Edge classifier**: Binary Cross entropy

- Edge weights are learned at the beginning of the block
Table detection

- Discard 0’ed edges
- Subgraphs with nodes classified as Table are considered
- Confidence score of these subgraphs are thresholded for the final decision
Datasets

CON-ANONYM

- 960 documents
- 8 region annotation
- Common car invoices
- Not publicly available

RVL-CDIP

- Overall 25,000 images
- 5 region annotation
- Selected 518 invoice class
- Publicly available
## Node classification

<table>
<thead>
<tr>
<th>Task</th>
<th>CON-ANONYM</th>
<th>RVL-CDIP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All</td>
<td>Table</td>
</tr>
<tr>
<td>Pow 2</td>
<td>82.8</td>
<td>96.4</td>
</tr>
<tr>
<td>+ Edge</td>
<td>84.2</td>
<td>97.0</td>
</tr>
<tr>
<td>Pow 5</td>
<td>82.7</td>
<td>96.2</td>
</tr>
<tr>
<td>+ Edge</td>
<td>84.5</td>
<td>97.2</td>
</tr>
</tbody>
</table>
# Table Detection

<table>
<thead>
<tr>
<th>Task</th>
<th>CON-ANONYM</th>
<th></th>
<th>RVL-CDIP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1-Score</td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>Pow 2</td>
<td>69.4</td>
<td>65.8</td>
<td>73.4</td>
</tr>
<tr>
<td>+ Edge</td>
<td>70.8</td>
<td>65.2</td>
<td>77.6</td>
</tr>
<tr>
<td>Pow 5</td>
<td>68.4</td>
<td>65.3</td>
<td>71.8</td>
</tr>
<tr>
<td>+ Edge</td>
<td>73.7</td>
<td>78.4</td>
<td>69.5</td>
</tr>
</tbody>
</table>
Qualitative
Graph Neural Networks
Neural Message Passing (Reminder)

Message function:

\[ m_{v}^{t+1} = \sum_{w \in N_v} M(h_{v}^{t}, h_{w}^{t}, e_{vw}) \]

Update function:

\[ h_{v}^{t+1} = U(h_{v}^{t}, m_{v}^{t+1}) \]

Readout function:

\[ \hat{y} = R(\{ h_{v}^{T} \mid v \in G \}) \]

Simple Message Passing Layer

Let us consider a graph $G = (V, A)$ where $V$ is the set of nodes and $A$ the adjacency matrix

$$x^{(k+1)} = G_{(k)}(x^{(k)}) = \rho \left( \sum_{B \in \mathcal{A}(k)} B x^{(k)} \theta_{B}^{(k)} \right)$$

V. Garcia and J. Bruna. Few-Shot Learning with Graph Neural Networks. ICLR, 2018
Formalization

\[ x^{(k+1)} = G_C(x^{(k)}) = \rho \left( \sum_{B \in A^{(k)}} B x^{(k)} \theta_B^{(k)} \right) \]

\[ G = (V, A) \]

\[
V = \begin{pmatrix} 1 & 2 \\ 2 & 0 \\ 3 & 2 \\ 1 & 1 \\ 0 & 1 \\ 3 & 1 \end{pmatrix} \quad A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \]
Formalization

\[ G = (V, A) \]

\[ x^{(k+1)} = G_C(x^{(k)}) = \rho \left( \sum_{B \in A^{(k)}} B x^{(k)} \theta_B^{(k)} \right) \]

\[ \mathcal{A} = \{ I_{|V|}, A \} \]

\[ x^{(0)} = V \in \mathbb{R}^{|V| \times d_0} \]

\[ x^{(1)} = \rho \left( I_{|V|} x^{(0)} \theta^{(0)}_{I_{|V|}} + A x^{(0)} \theta^{(0)}_A \right) \]
Formalization

\[ G = (V, A) \]

\[ \mathcal{A} = \{ I_{|V|}, A \} \]

\[ x^{(0)} = V \in \mathbb{R}^{|V| \times d_0} \]

\[ x^{(1)} = \rho \left( x^{(0)} \theta_{I_{|V|}}^{(0)} + Ax^{(0)} \theta_{A}^{(0)} \right) \]

\[ x^{(k+1)} = G_C(x^{(k)}) = \rho \left( \sum_{B \in \mathcal{A}_k} B x^{(k)} \theta_B^{(k)} \right) \]
Formalization

\[ G = (V, A) \]

\[
\begin{align*}
\mathbf{x}^{(k+1)} &= G_C(\mathbf{x}^{(k)}) = \rho \left( \sum_{B \in A^{(k)}} B \mathbf{x}^{(k)} \theta_B^{(k)} \right)
\end{align*}
\]

\[
A \mathbf{x}^{(0)} =
\begin{pmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
\end{pmatrix}
\begin{pmatrix}
1 \\
2 \\
0 \\
3 \\
2 \\
1 \\
\end{pmatrix}
= 
\begin{pmatrix}
1 & 1 \\
1 & 2 \\
0 & 1 \\
9 & 4 \\
0 & 1 \\
\end{pmatrix}
\]
Formalization

But what other operators can we use in $\mathcal{A}$?

$$x^{(k+1)} = G_C(x^{(k)}) = \rho \left( \sum_{B \in \mathcal{A}^{(k)}} B x^{(k)} \theta_B^{(k)} \right)$$

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 2 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 3 \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$
Formalization

But can we learn the operator we use in $\mathcal{A}$?

$$
\phi_k(B) = \begin{pmatrix}
\phi_k(B)_{0,0} & \cdots & \phi_k(B)_{0,5} \\
\vdots & \ddots & \vdots \\
\phi_k(B)_{5,0} & \cdots & \phi_k(B)_{5,5}
\end{pmatrix}
$$

$$
\phi_k(B)_{i,j} = \begin{cases}
0 & \text{if } B_{i,j} = 0 \\
\sigma \left( \text{MLP}_{\tilde{\theta}} \left( \| x^{(k)}_i - x^{(k)}_j \| \right) \right) & \text{otherwise}
\end{cases}
$$

$$
\chi^{(k+1)} = \mathcal{G}_C(\chi^{(k)}) = \rho \left( \sum_{B \in \mathcal{A}(k)} B \chi^{(k)} \theta^{(k)}_B \right)
$$
Simple Message Passing Layer

Let us consider a graph \( G = (V, A) \) where \( V \) is the set of nodes and \( A \) the adjacency matrix

\[
x^{(k+1)} = G_C(x^{(k)}) = \rho \left( \sum_{B \in A^{(k)}} B x^{(k)} \theta_B^{(k)} \right)
\]

V. Garcia and J. Bruna. *Few-Shot Learning with Graph Neural Networks*. ICLR, 2018
Frameworks
Deep Learning Frameworks

- PyTorch
  - Rapid prototyping in Research
  - Dynamic computational graphs
  - Debugging

- TensorFlow
  - Large-scale deployments
  - Cross-platform and embedded deployment
  - Static computational graphs
Graph Neural Networks Libraries

- Pytorch-geometric
- Deep Graph Library

- Graph Nets (DeepMind)
Graph Neural Networks Libraries

- Fast re-implementation of existing models
- Faster


M. Wang et al. Deep Graph Library: Towards Efficient And Scalable Deep Learning on Graphs. ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019
Scratch Implementation
Implementation
Recommended Reading


Recommended Reading

  - Workshops: ICCV, ECCV, BMVC, …
  - Tutorials: CVPR, NIPS, ECCV, SIGGRAPH, …
- Steeve Huang, “Hands-on Graph Neural Networks with PyTorch & PyTorch Geometric”
- “DeepGraphLibrary Tutorial”
Discussion and Closing

- Are graphs still relevant?
- Are graph-based methods still useful for Pattern Recognition and Document Image Analysis?
- What are the current trends and next steps?