Deep Convolutional Networks on Graph-Structured Data

By Mikael Henaff, Yann Lecunn from NYU and Joan Bruna from Berkeley

June 16 2015

ASHER KABAKOVITCH
ITAY BONEH
Outline

- Introduction and motivation.
- Graph Laplacian
- Graph Fourier Transform
- Spectral Filtering
- Algorithms: Pooling, forward & backward pass
- Graph estimation-Similarity matrix
- Results
- Summary
Introduction - Motivation

Why structured data?

- To include more information (relations between different data points - e.g. euclidean distance in images).
- To decrease learning complexity by making geometric Assumptions like locality, shift-invariance.
Introduction

Euclidean and non Euclidean data

- **Euclidean:** sound, time-series, images, video.
- **Non Euclidean:** Social networks, Biological networks, Infrastructure networks. Graphs are used to model uneven connections. Some of data lies in the structure. Natural representation.
Introduction to graphs

Basic definitions

- Vertices - V
- Edges - E, Weights
- Adjacency matrix - W
- Degree matrix
- No orientation

\[ D_{ii} = \sum_j W_{ij} \]
Introduction

Examples to problems on graphs

- **Graph classification** - E.g. take a molecule graph-> is it toxic?
- **Graph partitioning** - Parallel to semantic segmentation on a grid.
- **Vertex classification** - E.g. a cause for a molecule being poisonous or not.
Introduction to graphs

Functions on graphs

\[ f: V \rightarrow \mathbb{R} \]

\[ F: E \rightarrow \mathbb{R} \quad \text{In an undirected graph } F_{ij} = -F_{ji} \]
Introduction to graphs

Basic operations

Gradient: $\nabla$

\[ L^2(V) \rightarrow L^2(E) \]
\[ (\nabla f)_{ij} \rightarrow f_i - f_j \]

Divergence: $\nabla$

\[ L^2(E) \rightarrow L^2(V) \]
\[ (\text{div} F)_i = \sum_{j:(i,j) \in E} w_{ij} F_{ij} \]
Introduction to graphs

Basic operations

Laplacian: $L^2(V) \rightarrow L^2(V)$

$$\Delta f_i = \frac{1}{a_{i,j}} \sum_{(i,j) \in E} w_{ij} (f_i - f_j)$$

-Defined as the difference between a vertex and its local average.
Defining **convolutions and pooling on graphs** (non-euclidean).

- We can’t use the euclidean shift-invariant definition since the structure isn’t shift-invariant, so we use the spectral definition (Convolution is element-wise multiplication in the Fourier domain).
- The problem becomes to defining the Fourier transform on graphs.
Graph Fourier Transform

Euclidean (1D)
- Laplacian
- Laplacian eigenfunctions

\[ \frac{d^2}{dx^2} f = \lambda f \Rightarrow f = e^{-i\omega x}, \quad \lambda = \omega^2 \]

Non-euclidean (graph)
- Laplacian
- Laplacian eigenvectors
- A simple example of a labeled, undirected graph and its $L=D-W$

<table>
<thead>
<tr>
<th>Labeled graph</th>
<th>Degree matrix</th>
<th>Adjacency matrix</th>
<th>Laplacian matrix</th>
</tr>
</thead>
</table>
| ![Graph Image] | \[
\begin{pmatrix}
2 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 3 & 0 \\
0 & 0 & 0 & 0 & 1
\end{pmatrix}
\] | \[
\begin{pmatrix}
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 0
\end{pmatrix}
\] | \[
\begin{pmatrix}
2 & -1 & 0 & 0 & -1 & 0 \\
-1 & 3 & -1 & 0 & -1 & 0 \\
0 & -1 & 2 & -1 & 0 & 0 \\
0 & 0 & -1 & 3 & -1 & -1 \\
-1 & -1 & 0 & -1 & 3 & 0 \\
0 & 0 & 0 & -1 & 0 & 1
\end{pmatrix}
\] |
The symmetric normalized Laplacian matrix is defined as:

\[ L^{\text{sym}} = D^{-1/2} L D^{-1/2} = D^{-1/2} (D - W) D^{-1/2} = I - D^{-1/2} W D^{-1/2} \]

\[
L^{\text{sym}}_{i,j} := \begin{cases} 
1 & \text{if } i = j \text{ and } \deg(v_i) \neq 0 \\
-\frac{1}{\sqrt{\deg(v_i) \deg(v_j)}} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\
0 & \text{otherwise.}
\end{cases}
\]
Graph Fourier Transform

Non-euclidean space (graph):

- Laplacian functions, $U$ take on different forms, which are the eigenvectors of $L$.

\[ U = (u_1, \ldots, u_N) \quad Lu = \lambda u \]

- Those eigenvectors are the Fourier transform, which are packed into $U$, the unique Fourier transform matrix.

\[ f(x) = \sum_{k \geq 0} \int_X f(x') \phi_k(x') dx' \phi_k(x) \]

\[ \hat{f}_k = \langle f, \phi_k \rangle_{L^2(X)} \]
Laplacian Eigenfunctions

Laplacian eigenfunctions, $U$, take on different forms
Spectral Filtering

Euclidean

- From convolution in space to convolution in spectral domain

Convolution of two vectors \( f = (f_0, \ldots, f_{n-1})^T \) and \( g = (g_0, \ldots, g_{n-1})^T \)

\[
\begin{align*}
f \ast g &= \begin{bmatrix}
g_0 & g_1 & \cdots & g_{n-1} \\
g_{n-1} & g_1 & \cdots & g_{n-2} \\
\vdots & \vdots & \ddots & \vdots \\
g_2 & g_3 & \cdots & g_0 & g_1 \\
g_1 & g_2 & \cdots & g_0 \\
\end{bmatrix} \begin{bmatrix}
f_1 \\
f_2 \\
\vdots \\
f_n \\
\end{bmatrix} \\
&= \Phi \begin{bmatrix}
\hat{g}_0 \\
\vdots \\
\hat{g}_{n-1}
\end{bmatrix} \Phi^T f = \Phi \begin{bmatrix}
\hat{g}_0 \\
\vdots \\
\hat{g}_{n-1}
\end{bmatrix} \begin{bmatrix}
\hat{f}_0 \\
\vdots \\
\hat{f}_{n-1}
\end{bmatrix}
\end{align*}
\]
Spectral Filtering

Non-euclidean

- A graph convolution of $X$ -Input signal with $M$ input channels and $N$ locations with filters $g$ on $G$ is defined by

$$x *_{G} g = U^T (U x \odot U g)$$

- Learning filters on a graph amounts to learning spectral multipliers (Convolution Theorem)

$$w_g = (w_1, \ldots, w_N)$$

- Convolutions diagonalize in the Fourier domain

Problems:

- Expensive (no FFT)
- Need to recalculate $(U, g)$ for each graph
- No finite support in graph space
Localization of Spectral Filtering

Motivation:

● Regular CNNs kernel are restricted to have a small spatial support, independent of input size.

● Therefore we’ll look for a smaller set of spectral multipliers.
Localization of Spectral Filtering

Localization in the graph domain ⇔ smoothness in the Fourier domain.

Due to the convolution theorem: \[ \left| \frac{\partial^k \hat{x}(\xi)}{\partial \xi^k} \right| \leq C \int |u|^k |x(u)| du \]

They learned a subset of spectral multipliers and performed smooth interpolations (splines). By considering a smoothing kernel: \( \mathcal{K} \in \mathbb{R}^{N \times N_0} \)

\[ w_g = \mathcal{K} \tilde{w}_g \]
Algorithm for forward and backward passes

**Algorithm 1 Train Graph Convolution Layer**

1: Given GFT matrix $U$, interpolation kernel $K$, weights $w$.
2: **Forward Pass:**
3: Fetch input batch $x$ and gradients w.r.t outputs $\nabla y$.
4: Compute interpolated weights: $w_{f'}f = Kw_{f'}f$.
5: Compute output: $y_{sf'} = U^T \left( \sum_f U x_{sf} \odot w_{f'}f \right)$.
6: **Backward Pass:**
7: Compute gradient w.r.t input: $\nabla x_{sf} = U^T \left( \sum_{f'} \nabla y_{sf'} \odot w_{f'}f \right)$
8: Compute gradient w.r.t interpolated weights: $\nabla w_{f'}f = U^T \left( \sum_s \nabla y_{sf'} \odot x_{sf} \right)$
9: Compute gradient w.r.t weights $\nabla w_{f'}f = K^T \nabla w_{f'}f$.

$P$ - input feature maps
$Q$ - output feature maps
$f = 1, \ldots, P$, $f' = 1, \ldots, Q$
Graph Pooling with Hierarchical Graph Clustering

calculate statistics of neighbors into a single node in the output. Choose neighbors using Multi-resolution Spectral Clustering:

Normalized spectral clustering according to Ng, Jordan, and Weiss (2002)

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number $k$ of clusters to construct.

- Construct a similarity graph by one of the ways described in Section 2. Let $W$ be its weighted adjacency matrix.
- Compute the normalized Laplacian $L_{sym}$.
- Compute the first $k$ eigenvectors $u_1, \ldots, u_k$ of $L_{sym}$.
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- Form the matrix $T \in \mathbb{R}^{n \times k}$ from $U$ by normalizing the rows to norm 1, that is set $t_{ij} = u_{ij}/(\sum_k u_{ik}^2)^{1/2}$.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i$-th row of $T$.
- Cluster the points $(y_i)_{i=1,...,n}$ with the $k$-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.
Unsupervised Graph Estimation

- Given data \( X \in R^{LxN} \) where \( L \) is the number of samples and \( N \) the number of features: \( d(i, j) = \|X_i - X_j\|^2 \)
- Does not require labeled data
- This distance was use to build two Gaussian diffusion Kernels

1. \( \omega(i, j) = \exp\left(-\frac{d(i,j)}{\sigma^2}\right) \)  
2. \( \omega(i, j) = \exp\left(-\frac{d(i,j)}{\sigma_i \sigma_j}\right) \)
Supervised Graph Estimation

- The relevant statistics may not correspond to our imposed similarity criteria.
- Look for feature similarity that best suits a particular classification task.
- **Input:** normalized features $X \in \mathbb{R}^{L \times N}$ and labels $y \in \{1, \ldots, C\}^L$
- Using a fully connected network to determine feature similarity. With K layers of weights $W_i$. Using standard Relu and dropout.
- Extract the first layer features: $W_1 \in \mathbb{R}^{N \times M_1}$, $M_1$-hidden feature of layer 1.

$$d_{sup}(i, j) = \|W_{1,i} - W_{1,j}\|^2$$: Criterion that best serve the classification task.

- This construction can be seen as “distilling” the information Learnt by the first network into a kernel. $d_{sup}(i, j) = \|W_{1,i} - W_{1,j}\|^2$
Results Notations

- **GC\(_k\)**: graph convolutional layer with \(k\) feature maps.
- **P\(_k\)**: graph pooling layer with stride \(k\) and pool size \(2k\).
- **FC\(_k\)**: denotes a fully connected layer with \(k\) hidden units.
- **P\(_\text{net}\)**: number of free parameters in the network.
- **P\(_\text{graph}\)**: number of free parameters when estimating the graph.
Results-Reuters dataset

- Consists of training and test sets each containing 201,369 documents from 50 mutually exclusive classes.
- Each document is represented as a log-normalized bag of words for 2000 common non-stop words (the, is, at, which, and on).
- Used the fully connected network with two hidden layers consisting of 2000 and 1000 hidden units regularized with dropout.
Results - Reuters dataset, Hyperparameters

- Were chosen by performing initial experiments on a validation set consisting of one tenth of the training data.
- Subsampled weights: $k = 60$
- Used max pooling.
Results - **Reuters dataset**

Similarity graphs (W) for the Reuters dataset.

- **Global sigma**
- **Local sigma**
Results—Compares the test accuracy of the FC network and the GC4-P4-FC1000 network.

Table 1: Results for Reuters dataset. Accuracy is shown at epochs 200 and 1500.

<table>
<thead>
<tr>
<th>Graph</th>
<th>Architecture</th>
<th>$P_{net}$</th>
<th>$P_{graph}$</th>
<th>Acc. (200)</th>
<th>Acc. (1500)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>FC2000-FC1000</td>
<td>6 \cdot 10^6</td>
<td>0</td>
<td>70.18(^2)</td>
<td>70.18</td>
</tr>
<tr>
<td>Supervised</td>
<td>GC4-P4-FC1000</td>
<td>2 \cdot 10^6</td>
<td>2 \cdot 10^6</td>
<td>69.41</td>
<td>70.03</td>
</tr>
<tr>
<td>Supervised</td>
<td>GC8-P8-FC1000</td>
<td>2 \cdot 10^6</td>
<td>2 \cdot 10^6</td>
<td>69.15</td>
<td>-</td>
</tr>
<tr>
<td>Supervised low rank</td>
<td>GC4-P4-FC1000</td>
<td>2 \cdot 10^6</td>
<td>5 \cdot 10^5</td>
<td>69.25</td>
<td>-</td>
</tr>
<tr>
<td>Supervised low rank</td>
<td>GC8-P8-FC1000</td>
<td>2 \cdot 10^6</td>
<td>5 \cdot 10^5</td>
<td>68.35</td>
<td>-</td>
</tr>
<tr>
<td>Supervised</td>
<td>GC16-P4-GC16-P4-FC1000</td>
<td>2 \cdot 10^6</td>
<td>2 \cdot 10^6</td>
<td>69.04</td>
<td>-</td>
</tr>
<tr>
<td>Supervised</td>
<td>GC64-P8-GC64-P8-FC1000</td>
<td>2 \cdot 10^6</td>
<td>2 \cdot 10^6</td>
<td>69.09</td>
<td>-</td>
</tr>
<tr>
<td>RBF kernel</td>
<td>GC4-P4-FC1000</td>
<td>2 \cdot 10^6</td>
<td>2 \cdot 10^6</td>
<td>67.85</td>
<td>-</td>
</tr>
<tr>
<td>RBF kernel</td>
<td>GC8-P8-FC1000</td>
<td>2 \cdot 10^6</td>
<td>2 \cdot 10^6</td>
<td>66.95</td>
<td>-</td>
</tr>
<tr>
<td>RBF kernel</td>
<td>GC16-P4-GC16-P4-FC1000</td>
<td>2 \cdot 10^6</td>
<td>2 \cdot 10^6</td>
<td>67.16</td>
<td>-</td>
</tr>
<tr>
<td>RBF kernel</td>
<td>GC64-P8-GC64-P8-FC1000</td>
<td>2 \cdot 10^6</td>
<td>2 \cdot 10^6</td>
<td>67.42</td>
<td>-</td>
</tr>
<tr>
<td>RBF kernel (local)</td>
<td>GC4-P4-FC1000</td>
<td>2 \cdot 10^6</td>
<td>2 \cdot 10^6</td>
<td>68.56</td>
<td>-</td>
</tr>
<tr>
<td>RBF kernel (local)</td>
<td>GC8-P8-FC1000</td>
<td>2 \cdot 10^6</td>
<td>2 \cdot 10^6</td>
<td>67.66</td>
<td>-</td>
</tr>
</tbody>
</table>
Results - Merck Molecular Activity Challenge

- Computational biology benchmark
- The task: predict activity level for different molecules based on the distances in bonds between different atoms.
- Used the DPP4 dataset: has 8193 samples and 2796 features.
- Used a network with 4 hidden layers and is regularized using dropout and weight decay.
- K=40 subsample weights worked best and average pooling.
Results-Merck Molecular Activity Challenge

Similarity graphs (W) for Merck Molecular Activity Challenge

Global sigma

Local sigma
Results - Merck Molecular Activity Challenge

Figure 2: Evolution of Test accuracy. Left: Reuters dataset, Right: Merck dataset.
### Results - Merck Molecular Activity Challenge

#### Table 2: Results for Merck DPP4 dataset.

<table>
<thead>
<tr>
<th>Graph</th>
<th>Architecture</th>
<th>$P_{\text{net}}$</th>
<th>$P_{\text{graph}}$</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>FC4000-FC2000-FC1000-FC1000</td>
<td>22.1 · 10^6</td>
<td>0</td>
<td>0.2729</td>
</tr>
<tr>
<td>Supervised</td>
<td>GC16-P4-GC16-P4-FC1000-FC1000</td>
<td>3.8 · 10^6</td>
<td>3.9 · 10^6</td>
<td>0.2773</td>
</tr>
<tr>
<td>Supervised</td>
<td>GC64-P8-GC64-P8-FC1000-FC1000</td>
<td>3.8 · 10^6</td>
<td>3.9 · 10^6</td>
<td>0.2580</td>
</tr>
<tr>
<td>RBF Kernel</td>
<td>GC64-P8-GC64-P8-FC1000-FC1000</td>
<td>3.8 · 10^6</td>
<td>3.9 · 10^6</td>
<td>0.2037</td>
</tr>
<tr>
<td>RBF Kernel (local)</td>
<td>GC64-P8-GC64-P8-FC1000-FC1000</td>
<td>3.8 · 10^6</td>
<td>3.9 · 10^6</td>
<td>0.1479</td>
</tr>
</tbody>
</table>
Results- ImageNet

- Performed the same experiments on the ImageNet dataset for which the graph is known.
- Used FFT.
- Network consisted of 4 convolution/ReLU/max pooling layers followed by a 3 fully-connected layers with 4096 hidden units.
Results- 

**ImageNet**

<table>
<thead>
<tr>
<th>Graph</th>
<th>Architecture</th>
<th>Test Accuracy (Top 5)</th>
<th>Test Accuracy (Top 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-D Grid</td>
<td>Convolutional Network</td>
<td>71.854</td>
<td>46.24</td>
</tr>
<tr>
<td>2-D Grid</td>
<td>Spectral Network</td>
<td>71.998</td>
<td>46.71</td>
</tr>
</tbody>
</table>

- Input images were scaled down to 128x128 to accelerate training.
Results- ImageNet

- Both models converged after 40 epochs to the same performance.
- The spectral network learns faster than the ConvNet during the first part of the training.

Figure 3: ConvNet vs. SpectralNet on ImageNet.
Summary and discussion-Challenges

- Both forward and backward evaluations require a multiplication by the GFT, which is $O(N^2)$.
- When the input is not known, graph estimation is the bottleneck of the model, requiring $O(N^2)$ operations for general graphs.
- The architecture is sensitive to graph estimation errors (more localized and weight sharing).
Summary and discussion - Open issues

- Is it possible to find an approximation for the eigenbasis of the general graph Laplacians?
- Inserting $U$ into the network: Replacing the eigen-basis $U$ by an arbitrary unitary matrix and optimize by back propagation.
- Num-layers $O(N^2)$
Questions?