Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

Michael Defferrard, Xavier Bresson, Pierre Vandergheynst

Yuting Sun

Monday 27th November, 2017

Contents

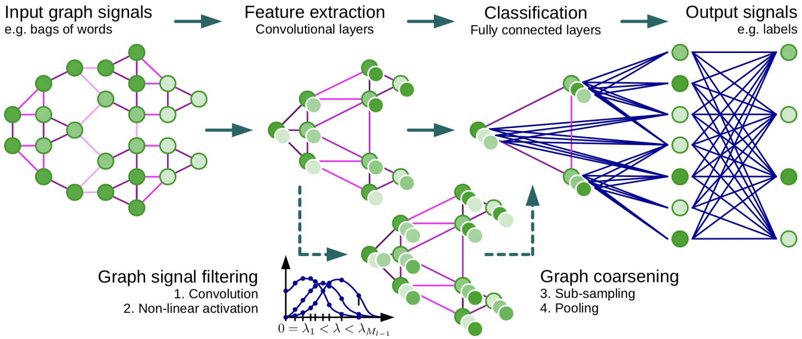
Introduction

- Convolutional neural networks (CNNs) offer an efficient architecture to extract highly meaningful statistical patterns in large-scale and high-dimensional datasets.
- In this work, we are interested in generalizing convolutional neural networks (CNNs) from low-dimensional regular grids to high-dimensional irregular domains
  - Low-dimensional: image, video, speech
  - High-dimensional: User data on social networks, gene data on biological regulatory networks, log data on telecommunication networks, text documents on word embeddings

Introduction

- Data lying on irregular or non-Euclidean domains that can be structured with graphs.
- Graphs can encode complex geometric structures and can be studied with strong mathematical tools such as spectral graph theory.
- A generalization of CNNs to graphs is not straightforward as the convolution and pooling operators are only defined for regular grids.
- The major bottleneck of generalizing CNNs to graphs, and one of the primary goals of this work, is the definition of localized graph filters which are efficient to evaluate and learn.

Architecture of a CNN on graphs



Proposed technique

Generalizing CNNs to graphs requires three fundamental steps:

- The design of localized convolutional filters on graphs
- A graph coarsening procedure that groups together similar vertices
- A graph pooling operation that trades spatial resolution for higher filter resolution.

Learning Fast Localized Spectral Filters

There are two strategies to define convolutional filters.

- spatial approach
  - A spatial approaches provides filter localization via the finite size of the kernel.
  - Although graph convolution in the spatial domain is conceivable, it faces the challenge of matching local neighborhoods.
  - There is no unique mathematical definition of translation on graphs from a spatial perspective.
- spectral approach
  - A spectral approach provides a well-defined localization operator on graphs via convolutions with a Kronecker delta implemented in the spectral domain.
  - A filter defined in the spectral domain is not naturally localized and translations are costly due to the $O(n^2)$ multiplication with graph Fourier basis.
  - Both limitations can be overcome with a special choice of filter parametrization.

Graph Fourier Transform

- Undirected and connected graphs $G = (V, E, W)$, where V is a finite set of $|V| = n$ vertices, E is a set of edges and $W \in R^{n\times n}$ is a weighted adjacency matrix encoding the connection weight between two vertices.
- A signal $x : V \to R$ defined on the nodes of the graph may be regarded as a vector $x \in R^n$ where x_i is the value of x at the i^{th} node.
- graph Laplacian:
  - combinatorial definition: $L = D - W \in R^{n\times n}$ where $D \in R^{n\times n}$ is the diagonal degree matrix with $D_{ii} = \sum_j W_{ij}$
  - normalized definition: $L = I_n - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ where I_n is the identity matrix.

Graph Fourier Transform

- As L is a real symmetric positive semidefinite matrix, it has a complete set of orthonormal eigenvectors $\{u_l\}_{l=0}^{n-1} \in R^n$, known as the graph Fourier modes, and their associated ordered real nonnegative eigenvalues $\{\lambda_l\}_{l=0}^{n-1}$, identified as the frequencies of the graph.
- The Laplacian is diagonalized by the Fourier basis $U = [u_0, \ldots, u_{n-1}] \in R^{n\times n}$ such that $L = U\Lambda U^T$ where $\Lambda = diag([\lambda_0, \ldots \lambda_{n-1}]) \in R^{n\times n}$.
- The graph Fourier transform of a signal $x \in R^n$ is then defined as $\widehat{x} = U^T x \in R^n$, and its inverse as $x = U\widehat{x}$.

Spectral filtering of graph signals

- The convolution operator on graph *g is defined in the Fourier domain such that $x *_g y = U((U^T X) \odot (U^T y))$, where $\odot$ is the element-wise Hadamard product.
- A signal x is filtered by g_θ as

$$y = g_\theta(L)x = g_\theta(U\Lambda U^T)x = Ug_\theta(\Lambda)U^T x \tag{1}$$

- A non-parameter filter is defined as

$$g_\theta(\Lambda) = diag(\theta) \tag{2}$$

where the parameter $\theta \in R^n$ is vector of Fourier coefficients

Polynomial parametrization for localized filters

- Two limitations with non-parametric filters
- These issue can be overcome with a polynomial filter

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k \tag{3}$$

where the parameter $\theta \in R^K$ is vector of polynomial coefficients.

- The value at vertex j of the filter g_θ centered at vertex i is given by $(g_\theta(L)\delta_i)_j = (g_\theta(L))_{i,j} = \sum_k \theta_k (L^k)_{i,j}$ where the kernel is localized via a convolution with a Kronecker delta function $\delta_i \in R^n$.
- Spectral filters represented by Kth order polynomials of the Laplacian are exactly K-localized.
- Then learning complexity is $O(K)$, the support size of the filter (same as classical CNNs).

Recursive formulation for fast filtering

- Learning localized filters with K parameters, the cost to filter a signal x as $y = Ug_\theta(\Lambda)U^T x$ is $O(n^2)$ operations because of the multiplication with the Fourier basis U.
- Parametrize $g_\theta(L)$ as a polynomial function that can be computed recursively from L, as K multiplications by a sparse L costs $O(K|E|) \ll O(n^2)$.
- A filter can be parametrized as the truncated expansion

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\widehat{\Lambda}) \tag{4}$$

of order $K-1$, where the parameter $\theta \in R^K$ is a vector of Chebyshev coefficients and $T_k(\widehat{\Lambda}) \in R^{n\times n}$ is the Chebyshev polynomial of order k evaluated at $\widehat{\Lambda} = 2\Lambda/\lambda_{max} - I_n$, a diagonal matrix of scaled eigenvalues that lie in $[-1, 1]$.

Recursive formulation for fast filtering

- The filtering operation can be written as $y = g_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{L})x$ where $T_k(\widehat{L}) \in R^{n\times n}$ is the Chebyshev polynomial of order k evaluated at the scaled Laplacian $\widetilde{L} = 2L/\lambda_{max} - I_n$.
- Denoting $\overline{x}_k = T_k(\widetilde{L})x \in R^n$, use the recurrence relation to compute $\overline{x}_k = 2\widetilde{L}\overline{x}_{k-1} - \overline{x}_{k-2}$ with $\overline{x}_0 = x and \overline{x}_1 = \widetilde{L}x$.
- This filtering operation costs $O(K|E|)$ operations.

Learning Filters

- The jth output feature map of the sample s is given by

$$y_{s,j} = \sum_{i=1}^{F_{in}} g_{\theta_{i,j}}(L) x_{s,i} \in R^n \tag{5}$$

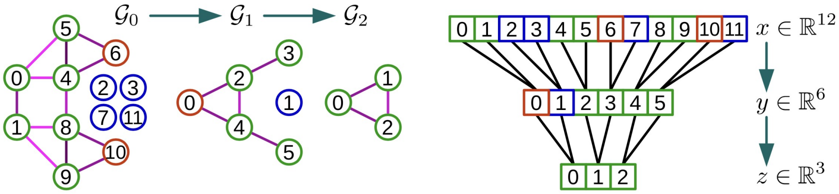
  where the $x_{s,i}$ are the input feature maps and the $F_{in} \times F_{out}$ vectors of Chebyshev coefficients $\theta_{i,j} \in R^K$ are the layer's trainable parameters.
- Train multiple convolutional layers with the backpropagation algorithm.
- Cost is $O(K|E|F_{in}F_{out}S)$ (S is the number of samples in mini-batch).

Graph Coarsening

- Doing pooling operation for multiple layers is equivalent to a multi-scale clustering of the graph that preserves local geometric structures.
- Graph clustering is NP-hard.
- Use the coarsening phase of the Graclus multilevel clustering algorithm.
- Graclus uses a greedy algorithm to compute successive coarser versions of a given graph.
- Graclus greedy rule:
  - Pick an unmarked vertex i
  - Matching it with one of its unmarked neighbors j that maximizes the local normalized cut $W_{ij}(1/d_i + 1/d_j)$
  - Mark the two matched vertices
  - The coarsened weights are set as the sum of their weights.
  - The matching is repeated until all nodes have been explored.

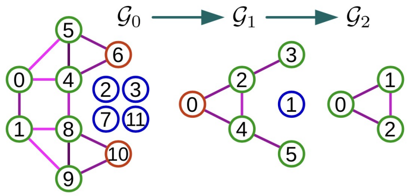
Fast Pooling of Graph Signals

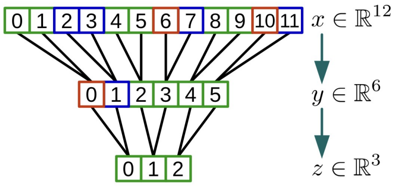
- We proceed in two steps: create a balanced binary tree and rearrange the vertices
- The structure of a balanced binary tree
    - Regular nodes (and singletons) either have two regular nodes (level 1 vertex 0 in Figure 2), OR
    - One singleton and a fake node as children (level 2 vertex 0), AND
    - Fake nodes always have two fake nodes as children (level 1 vertex 1)



Fast Pooling of Graph Signals

Pooling such a rearranged graph signal is analog to pooling a regular 1D signal. Figure 2 shows an example of the whole process.





Numerical Experiments

- Filters:
  - **Non-Param** (non-parametric and non-localized filters) : $g_\theta(\Lambda) = diag(\theta)$
  - **Spline**: $g_\theta(\Lambda) = B\theta$ where $B \in R^{n \times K}$ is the cubic B-spline basis and the parameter $\theta \in R^K$ is a vector of control points.
  - **Chebyshev**: $g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\widehat{\Lambda})$
- Graclus coarsening algorithm
- Some notations:
  - **FCk**: a fully connected layer with k hidden units
  - **Pk**: a (graph or classical) pooling layer of size and stride k
  - **GCk** and**Ck**: a (graph) convolutional layer with k feature maps
- All FCk, Ck and GCk layers are followed by a ReLU activation max(x, 0).
- The final layer is always a softmax regression
- The loss energy E is the cross-entropy with an l_2 regularization on the weights of all FCk layers.
- Mini-batches are of size S = 100.

Revisiting Classical CNNs on MNIST

- Benchmark MNIST classification problem: a dataset of 70,000 digits represented on a 2D grid of size 28×28.
- Our model: an 8-NN graph of the 2D grid which produces a graph of $n = |V| = 976$ nodes and $|E| = 3198$ edges. The weights of a k-NN similarity graph (between features) are computed as

$$W_{ij} = exp[-\frac{\|z_i - z_j\|_2^2}{\sigma^2}] \tag{6}$$

where z_i is the 2D coordinated of pixel i.

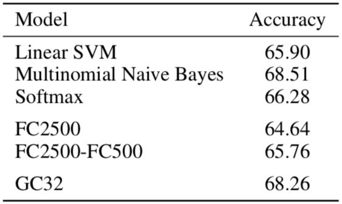
Model	Architecture	Accuracy
Classical CNN	C32-P4-C64-P4-FC512	99.33
Proposed graph CNN	GC32-P4-GC64-P4-FC512	99.14

Text Categorization on 20NEWS

- Text categorization problem on the 20NEWS: 18,846 (11,314 for training and 7,532 for testing) text documents associated with 20 classes
- We extracted the 10,000 most common words from the 93,953 unique words in this corpus.
- Each document x is represented using the bag-of-words model, normalized across words.
- Our model: 16-NN graph with $W_{ij} = exp[-\frac{\|z_i - z_j\|_2^2}{\sigma^2}]$, where z_i is the word2vec embedding of word i, which produced a graph of $n = |V| = 10,000$ nodes and $|E| = 132,834$ edges.
- All models were trained for 20 epochs by the Adam optimizer with an initial learning rate of 0.001.
- The architecture is GC32 with support K $=$ 5.

Accuracies of the proposed graph CNN and other methods on 20NEWS

The proposed model does not outperform the multinomial naive Bayes classifier on this small dataset. It does defeat fully connected networks, which require much more parameters.



Model	Accuracy
Linear SVM	65.90
Multinomial Naive Bayes	68.51
Softmax	66.28
FC2500	64.64
FC2500-FC500	65.76
GC32	68.26

Table 2: Accuracies of the proposed graph CNN and other methods on 20NEWS.

Comparison between Spectral Filters and Computational Efficiency

Model	Accuracy
Linear SVM	65.90
Multinomial Naive Bayes	68.51
Softmax	66.28
FC2500	64.64
FC2500-FC500	65.76
GC32	68.26

Table 2: Accuracies of the proposed graph CNN and other methods on 20NEWS.

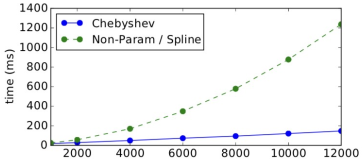


Figure 3: Time to process a mini-batch of $S = 100$ 20NEWS documents w.r.t. the number of words n.

		Accuracy		
Dataset	Architecture	Non-Param (2)	Spline (7) [4]	Chebyshev (4)
MNIST	GC10	95.75	97.26	97.48
MNIST	GC32-P4-GC64-P4-FC512	96.28	97.15	99.14

Table 3: Classification accuracies for different types of spectral filters ($K = 25$).

		Time (ms)		
Model	Architecture	CPU	GPU	Speedup
Classical CNN	C32-P4-C64-P4-FC512	210	31	6.77x
Proposed graph CNN	GC32-P4-GC64-P4-FC512	1600	200	8.00x

Table 4: Time to process a mini-batch of $S = 100$ MNIST images.

Influence of Graph Quality

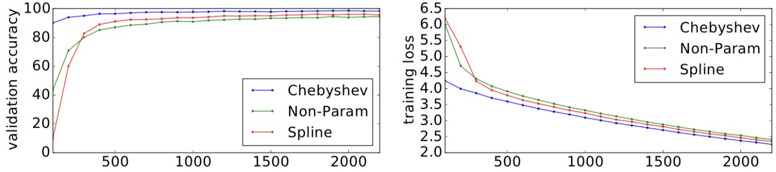


Figure 4: Plots of validation accuracy and training loss for the first 2000 iterations on MNIST.

Architecture	8-NN on 2D Euclidean grid	random
GC32	97.40	96.88
GC32-P4-GC64-P4-FC512	99.14	95.39

Table 5: Classification accuracies with different graph constructions on MNIST.

	word2vec			
bag-of-words	pre-learned	learned	approximate	random
67.50	66.98	68.26	67.86	67.75

Table 6: Classification accuracies of GC32 with different graph constructions on 20NEWS.

Conclusion

- **Spectral formulation**: Theoretical formulation of CNNs on graphs built on established tools in GSP.
- **Strictly localized filters**: Strictly localized in a ball of radius K.
- **Low computational complexity**: The evaluation complexity of our filters is linear.
- **Efficient pooling**: After a rearrangement of the vertices as a binary tree structure, is analog to pooling of 1D signals.
- **Experimental results**:
  - a useful model
  - computationally efficient
  - superior both in accuracy and complexity to the pioneer spectral graph CNN

Future Work

Future works will investigate two directions.

- Enhance the proposed framework with newly developed tools in GSP.
- Explore applications of this generic model to important fields where the data naturally lies on graphs, which may then incorporate external information about the structure of the data rather than artificially created graphs which quality may vary as seen in the experiments.

And, alternate the learning of the CNN parameters and the graph.

Question

Any Questions?