Convolutional Neural Networks via Node-Varying Graph Filters

Fernando Gama, Geert Leus,
Antonio G. Marques & Alejandro Ribeiro

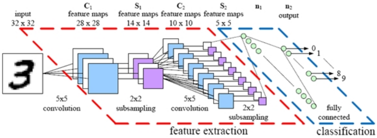
Dept. of Electrical and Systems Engineering
University of Pennsylvania
fgama@seas.upenn.edu

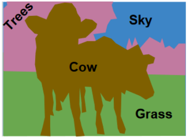
IEEE Data Science Workshop (DSW), June 6, 2018

Convolutional Neural Networks (CNNs)

- Information processing architecture $\Rightarrow$ Meaningful representation of data
- Remarkable performance in classification and regression tasks
  - $\Rightarrow$ Image classification (images)
  - $\Rightarrow$ Object recognition (images)
  - $\Rightarrow$ Speech recognition (time)
  - $\Rightarrow$ Game playing (time and images)
- Architecture parameters efficiently learned from training data
- Concatenation of simple layers $\Rightarrow$ Convolution, pooling, nonlinearity

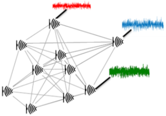
  $\Rightarrow$ Convolution and pooling only defined on regular domains

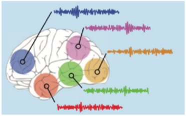




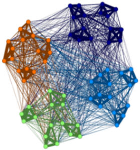
Data on irregular domains: Graph signals

- Time and images $\Rightarrow$ Information related by regular structure (grid)
- Many datasets present alternative structures (irregular)


Wireless Sensor Network


Brain Network

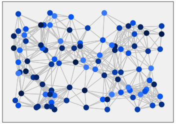

Transportation Network


Social Network

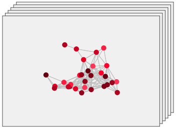
- Encode arbitrary pairwise relationships between data elements
  - $\Rightarrow$ Graph signals (data supported on the nodes of a graph)
  - $\Rightarrow$ Incorporate the underlying irregular structure into processing

Convolutional Neural Networks for Graphs

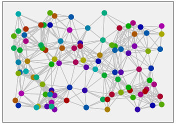
- Existing CNNs $\Rightarrow$ Remarkable performance in processing regular data
  - $\Rightarrow$ Convolution, pooling need a regular, multi-resolution domain
- Generalize CNNs to enable processing of signals supported on graphs
  - $\Rightarrow$ Convolution $\Rightarrow$ Linear shift-invariant graph filters [Bruna '14]
  - $\Rightarrow$ Pooling $\Rightarrow$ Create new graph $\Rightarrow$ Clustering [Defferrard '17]
- GNN architectures defined on the original graph [Gama '18]
- Resolution change via shifts $\Rightarrow$ Node-variant graph filters [Segarra '17]
  - $\Rightarrow$ Distribute features across the nodes of the original graph
  - $\Rightarrow$ No increase in dimension $\Rightarrow$ No need for pooling



Input



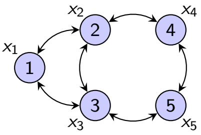
Clustering



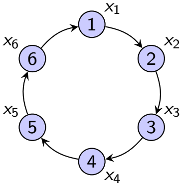
NVGF

Signals Supported on Graphs (Graph Signals)

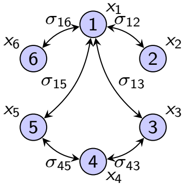
- Network structure $\Rightarrow$ Graph matrix $\mathbf{S}$ (Adjacency $\mathbf{A}$, Laplacian $\mathbf{L}$)
  $\Rightarrow [\mathbf{S}]_{ij}$ = Relationship between i and j (underlying graph support)
- Define a signal $\mathbf{x}$ on top of the graph $\Rightarrow [\mathbf{x}]_i$ = Signal value at node i
- Graph Signal Processing $\Rightarrow$ Exploit structure encoded in $\mathbf{S}$ to process $\mathbf{x}$
- Discrete time $\Rightarrow$ Cyclic graph $\Rightarrow$ Time n follows time $n-1$
- Random signals $\Rightarrow$ Covariance graph $\Rightarrow$ Correlation between elements



General graph



Cyclic graph



Covariance graph

Linear Shift Invariant Graph Filters

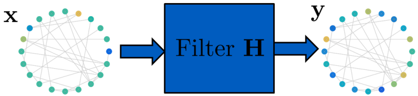
- Cycle $\Rightarrow$ Shift operator $\mathbf{S}$ acts as an actual shift $\Rightarrow [\mathbf{Sx}]_n = [\mathbf{x}]_{n+1}$

  $\Rightarrow$ Analogy to convolution operation to define a filter

$$\mathbf{y} \;=\; h_1\mathbf{S}^0\mathbf{x} + h_1\mathbf{S}^1\mathbf{x} + h_2\mathbf{S}^2\mathbf{x} + \ldots + h_L\mathbf{S}^L\mathbf{x} \;=\; \sum_{k=0}^{K-1} h_k\mathbf{S}^k\mathbf{x} \;:=\; \mathbf{Hx}$$

- $\mathbf{Sx}$ local operation $\Rightarrow$ $\mathbf{Hx}$ succession of $(K-1)$ local operations
- The filter $\mathbf{H}$ is linear shift invariant $\Rightarrow \mathbf{H}(\mathbf{Sx}) = \mathbf{S}(\mathbf{Hx}) = \mathbf{Sy}$
- Linear shift invariant filters generalize linear time invariant filters

  $\Rightarrow$ Generalize convolutional features in convolutional neural networks



Neural Networks (NNs)

- Consider a training set $\mathcal{T} = \{(\mathbf{x}, \mathbf{y})\}$ with input-output pairs $(\mathbf{x}, \mathbf{y})$
- Learning = Estimate output $\hat{\mathbf{y}}$ associated with input $\mathbf{x} \notin \mathcal{T}$

  $\Rightarrow$ Devise alternative representations of the dataset

- NNs stack layers composing pointwise nonlinearities with linear transforms (input $\mathbf{x}_0 = \mathbf{x}$ and output $\hat{\mathbf{y}} = \mathbf{x}_L$)

$$\mathbf{x}_1 = \sigma_1\Big(\mathbf{A}_1 \mathbf{x}\Big), \ \ldots, \ \mathbf{x}_\ell = \sigma_\ell\Big(\mathbf{A}_\ell \mathbf{x}_{\ell-1}\Big), \ \ldots, \ \mathbf{x}_L. = \sigma_L\Big(\mathbf{A}_L \mathbf{x}_{L-1}\Big)$$

  $\Rightarrow$ $\mathbf{x}_\ell$ of dimension M_ℓ $\Rightarrow$ $\mathbf{A}_\ell$ of dimension $M_{\ell-1} M_\ell$

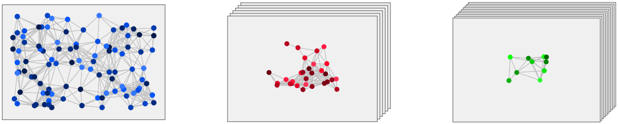
- Use $\mathcal{T}$ to find $\{\mathbf{A}_\ell\}$ that optimize loss function $\sum_{\mathcal{T}} f(\mathbf{y}, \mathbf{x}_L)$
- NNs are over-parametrized $\Rightarrow$ Difficult to train. Do not generalize

  $\Rightarrow$ Number of parameters to learn depends on data dimension

- CNNs regularize parameterization with convolution and pooling

Convolutional (C) Neural Networks (NNs)

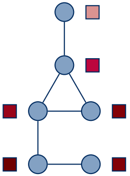
- Linear transform $\Rightarrow$ Convolution with bank of filters
  - $\Rightarrow$ K_ℓ: support of filters $\Rightarrow$ Independent of data dimension
  - $\Rightarrow$ Compute several features for a more expressive formulation
- $\mathbf{x}_\ell$ are F_ℓ features of dimension N_ℓ $\Rightarrow$ $M_\ell = F_\ell N_\ell$
- Number of parameters to learn: $K_\ell F_\ell F_{\ell-1}$ $\Rightarrow$ Independent of N_ℓ
- Dimension of data: $F_\ell N_\ell$ $\Rightarrow$ If F_ℓ increases, decrease N_ℓ
  - $\Rightarrow$ Pooling computes summaries $\Rightarrow$ Decrease dimension $\Rightarrow$ $N_\ell \leq N_{\ell-1}$
- Convolution $\Rightarrow$ Bank of linear shift-invariant graph filters
- Pooling $\Rightarrow$ Multiscale hierarchical clustering $\Rightarrow$ Smaller graphs

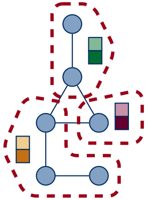


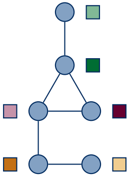
Input Layer 1 Layer 2

Node-Variant Graph Filters

- Clustering $\Rightarrow$ Open problem $\Rightarrow$ Difficult to determine a *good* cluster
- Avoid increasing dimensionality by using one filter per layer ($F_\ell = 1$)
  $\Rightarrow$ Expressiveness? Node-variant graph filter (NVGF)
- NVGF: $\mathbf{H}^{\text{nv}} = \sum_{k=0}^{K-1} \text{diag}(\mathbf{h}_k)\mathbf{S}^k$ $\Rightarrow$ $[\mathbf{h}_k]_n$: coefficient at node n
  $\Rightarrow$ $\mathbf{x}_\ell = \mathbf{H}_\ell^{\text{nv}}\mathbf{x}_{\ell-1}$ $\Rightarrow$ Dimension of data is constant at every layer
  $\Rightarrow$ Distribute features across the original graph


Input


Clustering


NVGF

Hybrid Node-Variant Graph Filters

- NVGF $\Rightarrow$ Dimension of data remains constant at each layer
- Number of learnable parameters: $\sum_{\ell=1}^{L} NK_\ell$ $\Rightarrow$ Depends on N

  $\Rightarrow$ Undesirable for large graphs $\Rightarrow$ Hybrid NVGF

- $\mathbf{C}_\mathcal{B} \in \{0,1\}^{N \times B}$: tall binary matrix ($N \gg B$) such that $\mathbf{C}_\mathcal{B}\mathbf{1}_B = \mathbf{1}_N$
- $\mathbf{h}_{\mathcal{B},k} \in \mathbb{R}^B$: reduced vector of B filter coefficients

$$\mathbf{H}^{\mathrm{h}} = \sum_{k=0}^{K-1} \mathrm{diag}(\mathbf{C}_\mathcal{B}\mathbf{h}_{\mathcal{B},k})\mathbf{S}^k$$

  $\Rightarrow$ $B = N$, $\mathbf{C}_\mathcal{B} = \mathbf{I}$ $\Rightarrow$ node-variant; $B = 1$ $\Rightarrow$ node-invariant

- $\mathbf{C}_\mathcal{B}$: cols. indicate membership; $\mathbf{h}_{\mathcal{B},k}$: common coeff. for each group
- Number of learnable parameters: $\sum_{\ell=1}^{L} BK_\ell$ $\Rightarrow$ Independent of N

Hybrid NVGF CNN

procedure NVGF_CNN($\{\mathbf{x}\}$, $\mathcal{T}$, $\mathbf{S}$, $\{K_1, \ldots, K_{L-1}\}$, B)
    Create set $\mathcal{B}$ by selecting B nodes with highest degree
    Compute $\mathbf{C}_{\mathcal{B}}$ (see below)
    Create the $L-1$ layers:
    for $\ell = 1 : L-1$ **do**
        Create B filter taps $\{\mathbf{h}_{\mathcal{B},0}, \ldots, \mathbf{h}_{\mathcal{B},K_\ell - 1}\}$
        Obtain $\mathbf{H}_\ell^{\mathrm{h}} = \sum_{k=0}^{K_\ell - 1} \mathrm{diag}(\mathbf{C}_{\mathcal{B}} \mathbf{h}_{\mathcal{B},k}) \mathbf{S}^k$
        Apply non-linearity $\sigma_\ell(\mathbf{H}_\ell^{\mathrm{h}} \cdot)$
    end for
    Create readout layer
    Learn $\{\mathbf{h}_{\mathcal{B},0}, \ldots, \mathbf{h}_{\mathcal{B},K_\ell - 1}\}_{\ell=1}^{L-1}$ from $\mathcal{T}$
    Estimate output $\hat{\mathbf{y}}$ for $\mathbf{x} \notin \mathcal{T}$
end procedure

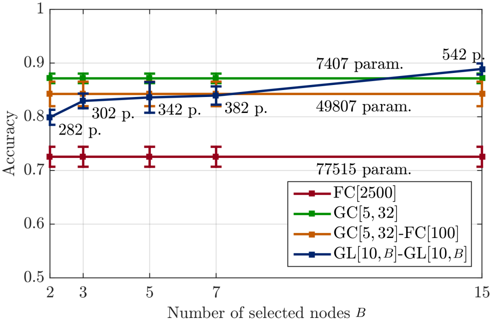
- $\mathcal{B} = \{v_1, \ldots, v_B\}$: B nodes with highest degree $\Rightarrow [\mathbf{C}_{\mathcal{B}}]_{v_b, b} = 1$
  $\Rightarrow i \notin \mathcal{B}$: $[\mathbf{C}_{\mathcal{B}}]_{ij} = 1$ if $j \in \mathrm{argmax}_{b: v_b \in \mathcal{B}} \{w_{i, v_b}\}$
- Other assignment schemes, other expansion models are possible

Numerical Experiments: Source Localization

- Consider Erdős-Rényi (ER) graph with $N = 15$ nodes, $p_{\mathrm{ER}} = 0.4$
- Assume node c started a diffusion at time $t = 0$
  - $\Rightarrow$ Graph signal $\boldsymbol{\delta}_c$ has 1 in node c and zeros elsewhere
- Consider observations $\mathbf{x} = \mathbf{A}^t \boldsymbol{\delta}_c$ for some unknown $t > 0$
- Localize the node c that originated the diffusion
- Test samples have white noise with σ_w^2 (training set is noiseless)
- Architectures tested are comprised of the following layers
  - $\Rightarrow$ FC[m]: fully connected layer with m hidden units
  - $\Rightarrow$ GC[K, F]: Chebyshev layer, K: length of filters, F: features
  - $\Rightarrow$ GL[K, B]: H-NVGF layer, K: length of filters, B: selected nodes
- Architectures include a (fully-connected) readout layer (not shown)

Source Localization: Number of Selected Nodes

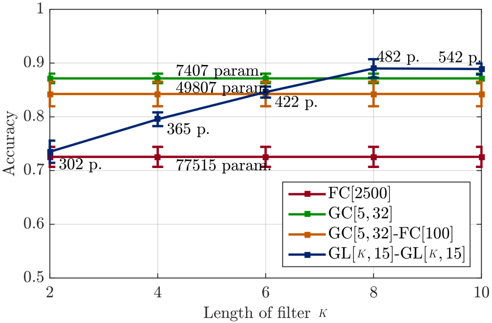
- Varying number of selected nodes B



- Performance of H-NVGF CNN improves when B increases
  - $\Rightarrow$ Still for $B = N$ few parameters and best performance

Source Localization: Length of Filter

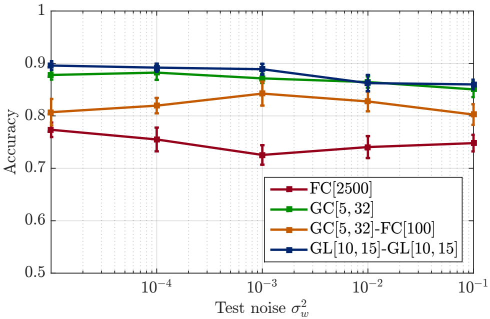
- Varying length of filter K



- Performance of H-NVGF CNN improves when K increases
  - $\Rightarrow$ After $K \geq 8$, two layers are enough to gather all information

Source Localization: Noise Power

- Varying noise power σ_w^2 in the test samples



- Drop of 5% across 5 orders of magnitude

Source Localization: Comparative Analysis

- Comparative analysis of the four architectures
- H-NVGF best performance with 10 times less parameters

Architecture	Parameters	Accuracy
FC[2500]	$77,515$	72.6%
GC[5, 32]	$7,407$	87.2%
GC[5, 32]-FC[100]	$49,807$	84.3%
GL[10, 15]-GL[10, 15]	542	88.9%

Numerical Experiments: 20NEWS Dataset

- Consider news articles that can be classified in 20 categories
- Model articles through a bag-of-words approach
- Graph support based on word2vec embedding, $N = 3,000$ nodes

- Comparable performance with almost 100 times less parameters

Architecture	Parameters	Accuracy
GC[5, 32]	1, 920, 212	60.75%
GL[5, 1500]	67, 521	60.34%

Conclusions

- Proposed CNN architecture that operates on graph signals
- Convolution stage replaced by a node-varying graph filter
  - $\Rightarrow$ No need for pooling (no need for clustering)
- Expressiveness $\Rightarrow$ Nodes with independent filter coefficients
- Multi-resolution analysis is adjusted by length of graph filters
- Hybrid NVGF $\Rightarrow$ Number of parameters independent of data dimension
- Tested on synthetic source localization and also `20NEWS` dataset
  - $\Rightarrow$ Similar performance with 10 to almost 100 less parameters