Learning Local Receptive Fields and their Weight Sharing Scheme on Graphs

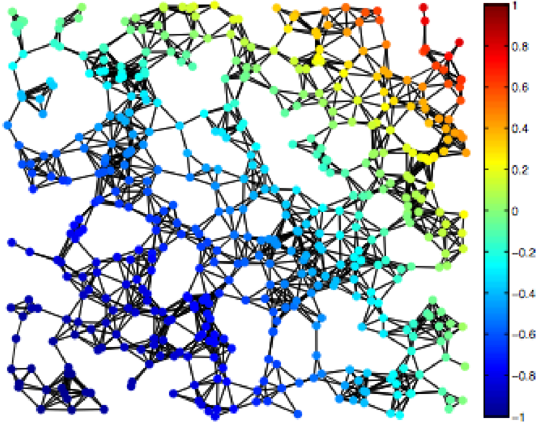
Jean-Charles Vialatte, Vincent Gripon, Gilles Coppin

Cityzen Data

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

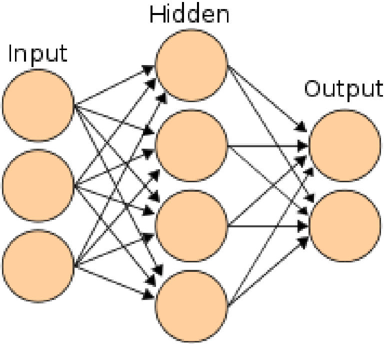
Nov. 14th 2017, IEEE Global SIP

Graph signal classification



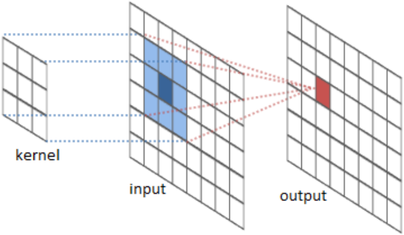
- a signal is taking values on vertices of a graph
- can we use a neural network to classify such signals ?

Off the shelf: fully connected layers



- With only the vertices as input, fully connected layers can be used
- Fits most industrial use cases
- Drawback: the edges carry important information but they are not used

Leveraging the underlying structure: convolutions



On images, convolutions make use of the underlying grid graph structure

- locality
- weight sharing
- neighbor matching

First approach: Spectral definition

Convolutions are defined as pointwise multiplications in the spectral domain

$$L = D - A = U \Lambda U^T$$

$$X \otimes W = U^T (UX.UW)$$

Examples:

- J. Bruna, et al, "Spectral networks and locally connected networks on graphs," arXiv preprint arXiv:1312.6203, 2013.
- M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," arXiv preprint arXiv:1506.05163, 2015.
- M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," NIPS, 2016.
- R. Levie, et al, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," arXiv preprint:1705.07664, 2017.

First approach: Spectral definition

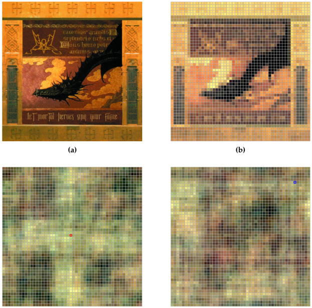
$$X \otimes W = U^T(UX.UW)$$

Pros

- Elegant and fast
- Work out of the shelf. Don't need to specify any weight sharing

Cons

- Spectral convolutions on grids do not match regular convolutions



(a) (b) (c) (d)

Second approach: Using the vertex domain

Convolutions with a kernel are defined as a function of neighboring vertices. Usually a dot product.

$$(X \otimes W)(v_i) = \sum_{j \in \mathcal{N}_{v_i}} w_{ij} X(v_j)$$

Examples:

- J-C. Vialatte, V. Gripon, and G. Mercier, "Generalizing the convolution operator to extend cnns to irregular domains," arXiv preprint arXiv:1606.01166, 2016.
- F. Monti, et al, "Geometric deep learning on graphs and manifolds using mixture model cnns," arXiv preprint:1611.08402, 2016.
- B. Pasdeloup, et al. "Convolutional neural networks on irregular domains through approximate translations on inferred graphs," arXiv preprint arXiv:1710.10035, 2017.

This is the approach used in the submitted paper

Second approach: Using the vertex domain

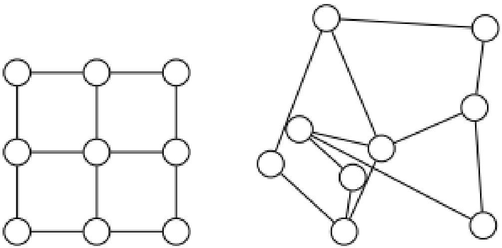
$$(X \otimes W)(v_i) = \sum_{j \in \mathcal{N}_{v_i}} w_{ij} X(v_j)$$

Pros

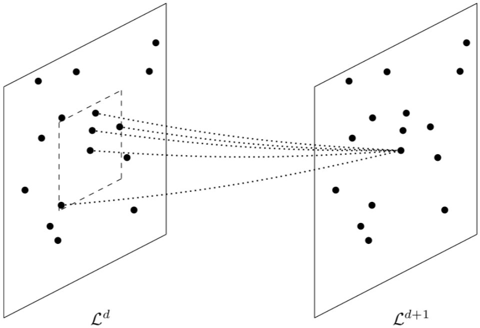
- Vertex-domain convolutions on grids match regular convolutions
- Same results on images if the full underlying graph structure is known

Cons

- w_{ij} ?
- How to define the weight sharing ?



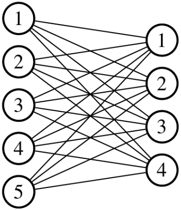
Local receptive fields / Local receptive graph



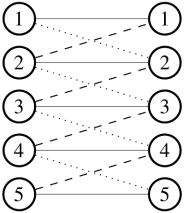
- graph of local receptive fields: local receptive graph
- the edges directly support the convolution

Usual case: learning one kernel

$$\mathbf{y} = f(W \cdot \mathbf{x} + \mathbf{b})$$



$$\begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \\ w_{51} & w_{52} & w_{53} & w_{14} \end{pmatrix}$$



$$\begin{pmatrix} w_2 & w_3 & 0 & 0 & 0 \\ w_1 & w_2 & w_3 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 \\ 0 & 0 & 0 & w_1 & w_2 \end{pmatrix}$$

Proposition: learning two kernels

$$\mathbf{y} = f(W \cdot S \cdot \mathbf{x} + \mathbf{b})$$

Learning W

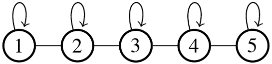
- weight kernel
- W tensor of shape *kernel_size* x *nb_input_channels* x *nb_feature_maps*

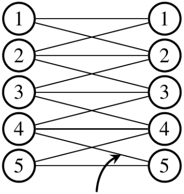
Learning S

- weight sharing scheme kernel
- controls how the parameters of W will be shared across the graph
- S tensor of shape *nb_input_vertices* x *nb_output_vertices* x *kernel_size*
- S is masked by the adjacency matrix of the graph

Graphical explanation

$$\mathbf{y} = f(W \cdot S \cdot \mathbf{x} + \mathbf{b})$$





$$\begin{pmatrix} \mathbf{s}_{11} & \mathbf{s}_{12} & 0 & 0 & 0 \\ \mathbf{s}_{21} & \mathbf{s}_{22} & \mathbf{s}_{23} & 0 & 0 \\ 0 & \mathbf{s}_{32} & \mathbf{s}_{33} & \mathbf{s}_{34} & 0 \\ 0 & 0 & \mathbf{s}_{43} & \mathbf{s}_{44} & \mathbf{s}_{45} \\ 0 & 0 & 0 & \mathbf{s}_{54} & \mathbf{s}_{55} \end{pmatrix} \quad (W \cdot S)_{45} = \sum_{k=1}^{\omega} S_{45k} W_k$$

Genericity

Fully connected layer

- kernel_size = nb_input_vertices x nb_output_vertices
- S_{ij} are all possible one-hot bit encoded vectors

Convolutional

- S one-hot bit encoded along third dimension
- S circulant along two first dimensions

Validation experiments on Image datasets

Restraining priors

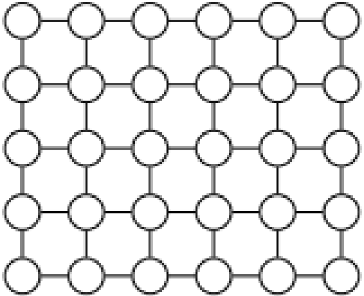
1. about edge matching for weight sharing
2. about any underlying graph structure

Full priors

3. widen a convolutional layer by also learning S

Experiments on Mnist

- No prior about edge matching or any ordering of vertices
- Knowledge of the underlying grid structure



A : adjacency matrix
A^k : connections with up to k-hop neighbors
S is masked with powers of A

Results

Conv5x5	A^1	A^2	A^3
(0.87%)	1.24% (1.21%)	1.02% (0.91%)	0.93% (0.91%)
A^4	A^5	A^6	A^{10}
0.90% (0.87%)	0.93% (0.80%)	1.00% (0.74%)	0.93% (0.84%)

Experiments on scramble Mnist

- No prior on underlying grid graph structure
- Usage of covariances between pixels

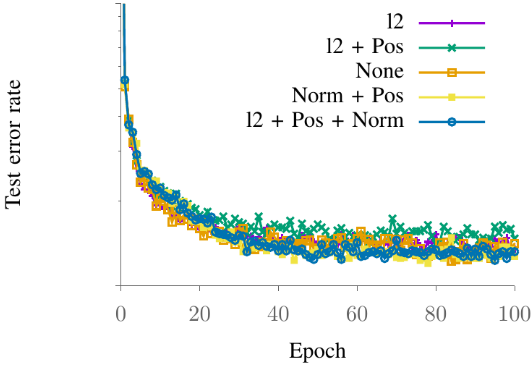
Results

- Tresholded: we keep 3% of edges with biggest covariance
- k-NN: for each pixel we keep $k = 25$ neighbors with biggest covariance

MLP	Conv5x5	Thresholded ($p = 3\%$)	k-NN ($k = 25$)
1.44%	1.39%	1.06%	0.96%

Forcing constraints ?

- Norm: normalizes S along third dimension
- Pos: only positive weights for S



Experiment on Cifar10

- Shallow networks widen by learning S

Support	Learn S	None	Pos	Norm	Both
Conv5x5	No	/	/	/	86.8 ± 0.2
Conv5x5	Yes	87.4 ± 0.1	87.1 ± 0.2	87.1 ± 0.2	87.2 ± 0.3
Grid^2	Yes	87.3 ± 0.2	87.3 ± 0.1	87.5 ± 0.1	87.4 ± 0.1

Conclusion

- We propose to learn weights and how they are shared
- The layer formulation is simple and generic
- It uses a graph representation of local receptive fields
- It attains performances comparable with convolutional ones

Future work

- Graph inference for initializing S
- Reducing number of parameters (ex: sharing S between layers in deep networks)
- Adding pooling
- Improving optimization
- Using S to define other operator-layers
- Semi-supervised and unsupervised