A non Euclidean approach for Atlas classification : *Graph Convolutions*

Joan Bruna, Gaspar Rochette

NYU

Introduction to Atlas classification

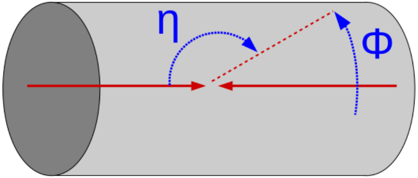
Two types of events :

- *signal* (interesting events), class $\mathbf{1}$, is a certain type of events that we want to distinguish from less interesting events.
- *background* (most events), class $\mathbf{0}$, is the rest.

Atlas data

Goal : automatically classify events coming from the **Atlas** collider, given for each energy peak :

- measures of the *energy*
- *phi* : azimuth angle
- *eta* : pseudorapidity



Graph convolutions and applications

Joan Bruna, Gaspar Rochette

References

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst.
Geometric deep learning : going beyond Euclidean data.
2016.

M. Defferrard, X. Bresson, and P. Vandergheynst.
Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering.
2016.

G. Louppe, K. Cho, C. Becot, and K. Cranmer.
QCD-Aware Recursive Neural Networks for Jet Physics.
2017.

Convolution definition

Let f be a function and ϕ a filter. f convolved with ϕ is defined as :

$$f * \phi(x) = \int f(x-u)\phi(u)du$$

for example, if f is an image, then the integral would be a sum.

If ϕ is non-zero only when close to 0, $f * \phi(x)$ only depends on values of $f(y)$ when y is close to x.

Convolutions

Convolutions are a **very powerful** tool for computer vision :

- extract complex **local patterns**
- **Fourier analysis** : convolution operator is diagonnal in Fourier basis, a decent image representation.
- **share filters** throughout the image
  $\rightarrow$ few coefficients
- robust to **small diffeomorphisms** & **translation covariant** (similar looking images will have similar outputs)
- **multiscale** influence in deep networks : influences from far away pixels are weak but in great numbers
- etc...

There is one problem :
It only works on Euclidean data (images, sounds).

Is it a problem ?

What if you data is **non-Euclidean** ? (Graphs, manifolds...)

An image can be a **poor representation of interactions**

- represent your data as nodes of a graph
- represent interactions/similarity as edge weights

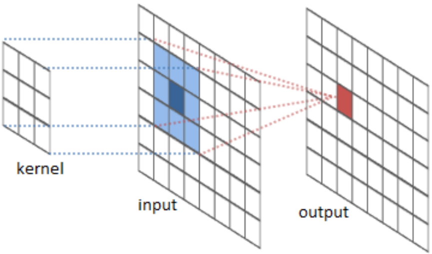
for example, two energy peaks occuring at p_i, p_j can have a similarity like :

$$w_{ij} = \exp\left(\frac{-\|p_i - p_j\|^2}{\sigma^2}\right)$$

Learn data structure :
In the example above, σ can be learnt from training. With regular convolution, the data structure is fixed.

Difficulty to generalize

Convolution on images :
"Sliding window"



Convolution on Graphs ?
Inconsistent neighborhood structure.
What would the window look like ?

Convincing operator on Graphs

Desirable properties :

- **local**
- **few parameters**
- respects **graph structure** : node index do not depend on the graph structure but its representation.
- **theoretical explanation** (why is it a convolution and not just some local operator ?)

Definition

Let (V, E) represent a graph, w_{ij} the weight of edge $i \to j$
Let x be a function $V \to \mathbb{R}$ (x_i is the state of node i)

A graph convolution is an operator GC changing the node state vector x such that :

$$GC(x) = \mathcal{P}(\tilde{W})x = \sum_{k=0}^{d} \alpha_k \tilde{W}^k x$$

where $\mathcal{P}$ is a polynomial of degree d, and W is a normalized version of the adjacency matrix (*e.g.* $D^{-1}W$)

Definition

$$GC(x) = \mathcal{P}(\tilde{W})x = \sum_{k=0}^{d} \alpha_k \tilde{W}^k x$$

- **local** : W^k represent an influence coming from nodes at distance k
- **few parameters** : degree d can be controlled
- respects **graph structure** : depends only on adjacency matrix
- **theoretical explanation** ? effect on Fourier Basis

Theoretical explanation

This formula comes from **spectal graph theory** : like regular convolution, graph convolutions are the operators that **share the Laplacian's eigen vectors** (Fourier Basis) : $GC(x)$ as defined here is a polynomial of the Graph Laplacian applied to x.

because of the Laplacian own stability, this operator is **provably stable by small deformation**.

A few properties

- input is a **sequence of events** : event i is $(p_1, \ldots, p_{s_i})$ where $p_k \in \mathbf{R}^d$ contains information about the k-th point of data. For example, p_k can contain the energy and locations of an energy peak, in an event counting s_i such peaks.
- Complexity $|V| \times diam(G)$ to transfer information from any two points in the graph
- This complexity can become $|V| \times \log |V|$ with **multi-grid**. (graph coarsening, multiple adjacency matrices)

Graph convolution on images

If we apply this convolution formed on a k-NN graph of an image (pixel nodes linked to adjacent pixels), we recover a very similar set of operators, but **isotropic**.

Parameter interpretation

$$GC(x) = \mathcal{P}_\theta(D^{-1}W)x = \sum_{k=0}^{d} \alpha_k (D^{-1}W)^k x$$

- degree d of polynomial $\mathcal{P}_\theta$ corresponds to kernel size for regular convolution
- each polynomial coefficient α_k corresponds to an influence from distance k neighborhood.
- $\mathcal{P}_\theta$ acts directly on $D^{-1}W$'s eigenvalues, so any symmetry in the graph structure leads to symmetric influence in GC

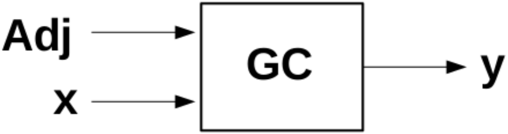
Possible architectures

In theory, **any convolution based layer can be used with graph convolution**, provided that it does not depend on data structures assumptions (pooling becomes more difficult, convolutions mustn't assume input size, etc...).

The main difference is that Graph Convolution also requires a **graph adjacency as input**. It can be one adjacency matrix reused for each layer, or it can be learnt.

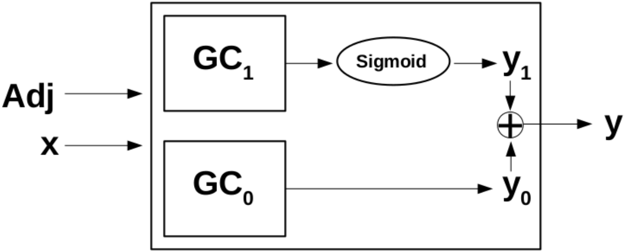
Here are a few graph convolution layers easily built from simple graph convolutions.

Graph Conv Layer



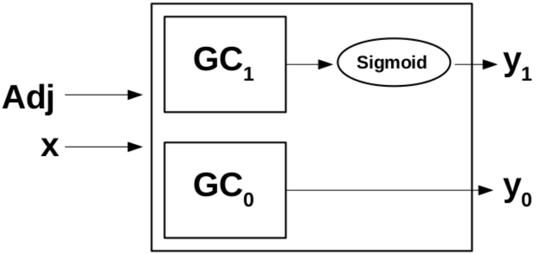
$$x \mapsto y = \rho\left(\mathcal{P}_\theta(W)x\right)$$

Modif Graph Conv Layer



$$x \mapsto y = \rho\left(\mathcal{P}_\theta^{(1)}(W)x + \sigma\left(\mathcal{P}_\theta^{(2)}(W)x\right)\right)$$

Residual Graph Conv Layer



$$x \mapsto y = \rho\left(\left[\mathcal{P}_\theta^{(1)}(W)x \mid \sigma\left(\mathcal{P}_\theta^{(2)}(W)x\right)\right]\right)$$

Architectures used

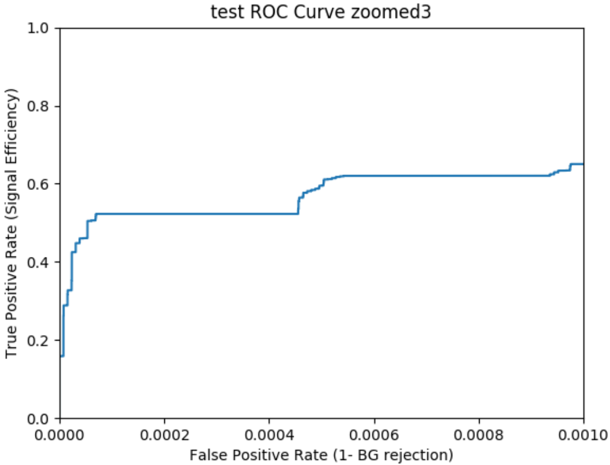
Networks based on the three previous layers, with various depths and numbers of feature maps. Inputs are energy, ϕ and η (and possibly edge renormalization factors).

Those were **single adjacency models**, meaning that only one adjacency is computed for one event, as a function of the geometrical data representation :

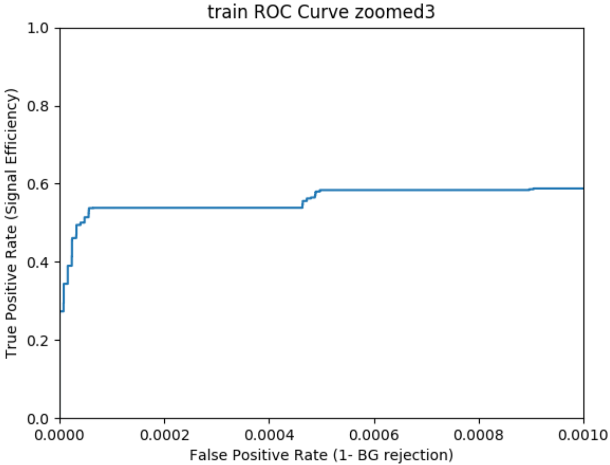
$$w_{ij} = \exp\left(-\|\alpha\phi + \beta\eta\|^2\right)$$

where α and β are learnt parameters. Weights can be renormalized, and the graph can be made sparse by zeroing out low weights (threshold, k-NN).

RGCs_FCL_50_200_200_200_200_BCE



RGCs_FCL_50_200_200_200_200_BCE



Regular Convolutions

Graph Convolutions

Definition

Comparison with regular convolutions

Atlas Classification

Architectures

Atlas results

Projects, Ideas

Projects, Ideas

Deep networks with sparse adjacency (only a few sets of hyper-parameters have been tested yet)

Graph Coarsening (pooling on graphs)

Node-Edge duality

- graph structure can be set with regards to the initial embedding in order to guarantee sparsity
- weights can be learnt for each layer as a function of the current node state.

This would take better advantage of the real strength of this model : **complex data representation**.