KERNEL-BASED GRAPH CONVOLUTIONAL NETWORKS

Hichem SAHBI

CNRS, Sorbonne University, Paris, France
ICPR 2020

Motivation and Contribution

Motivation

- Graph convolutional networks (GCNs) aim at generalizing deep learning to arbitrary irregular domains.
- Most of existing spatial GCNs follow a neighborhood aggregation scheme, where convolutions are recursively obtained by aggregating neighboring node representations using averaging or sorting operations.
- However, these operations are either ill-posed (mainly translation and receptive fields in convolutions) or weak to be discriminant.
- For highly nonlinear (and low dimensional) input graph signals (such as 3D skeletons in action recognition), relying on convolutions in the input space may limit the discrimination power of the learned graph representations.
- Indeed, aggregation based on averaging (when achieved in the input space) may dilute input node representations prior to convolution.
- An explicit expansion of the input node representations may enhance the discrimination power but comes at the expense of a substantial increase in the number of training parameters (thereby the risk of overfitting).
- It also increases the computational complexity both in space and time.

Contribution

- We consider, instead, an implicit mapping of the input graph signal in a RKHS.
- The method achieves an averaging aggregation and convolution in that space, in order to enhance the representational power of nodes and also the learned graph representations (without increasing the number of training parameters).
- Experiments conducted on the task of skeleton-based action recognition show the superiority of the proposed method against different baselines as well as the related work.

Graph convolutional networks at a glance

- Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph endowed with (i) a signal $\{s(u) \in \mathbb{R}^D\}_u$ and (ii) an adjacency matrix $\mathbf{A}$. Consider $g_\theta = (\mathcal{V}_\theta, \mathcal{G}_\theta)$ as a graphlet with $|\mathcal{V}_\theta| \ll |\mathcal{V}|$ and $|\mathcal{E}_\theta| \ll |\mathcal{E}|$. Following standard GCNs

$$(\mathcal{G} \star g_\theta)_u = \sigma(\mathbf{K}_\theta(u)), \quad \text{with} \quad \mathbf{K}_\theta(u) = \left\langle \sum_{u'} s(u').[\mathbf{A}^r]_{uu'}, w_\theta \right\rangle.$$

- In spite of being agnostic to any arbitrary permutation of nodes in $\mathcal{G}$ and g_θ, the above definition suffers from limited discrimination power, when s is low dimensional.

Our kernel-based graph convolutional networks

- Considering κ as a symmetric positive definite function (i.e., $\exists \psi : \mathcal{X} \rightarrow \mathcal{H}$, s.t., $\kappa(s(u'), s(v)) = \langle \psi(s(u')), \psi(s(v)) \rangle$).
- For a particular setting of w_θ as $\frac{1}{|\mathcal{V}_\theta|} \sum_{i=1}^N \alpha_i^\theta \psi(s(v_i^\theta))$ related to the representer theorem (Wahba 1971, Scholkopf 2001), with $\{v_i^\theta\}_i \subset \mathcal{V}_\theta$, $\{\alpha_i^\theta\}_i \subset \mathbb{R}$; the convolutional operator defined earlier can be rewritten as

$$\mathbf{K}_\theta(u) = \frac{1}{|\mathcal{N}_r(u)|.|\mathcal{V}_\theta|} \sum_{u' \in \mathcal{N}_r(u)} \left(\sum_{i=1}^N \alpha_i^\theta \, \kappa(u', v_i^\theta) \right).$$

- This evaluation does not require any explicit alignment between node pairs and it is thereby still invariant to any arbitrary permutation of nodes in $\mathcal{V}$ and $\mathcal{V}_\theta$.
- The strength of this kernel trick is its ability to handle nonlinear data as node representations are mapped into a high dimensional (and more discriminating) space $\mathcal{H} = \mathbb{R}^H$.
- E.g., the polynomial $\kappa(s(u), s(v)) = \langle s(u), s(v) \rangle^p$, its mapping is $\psi(s(u)) = s(u) \otimes \cdots \otimes s(u)$ (Maji 2012, Vedaldi 2012, Sahbi 2015).
- As H grows exponentially w.r.t p and polynomially w.r.t D, the kernel form is rather computationally more efficient.
- Considering a non-parametric setting, when only $\{\alpha_i^\theta\}_i$ are allowed to vary in $w_\theta = \frac{1}{|\mathcal{V}_\theta|} \sum_i \alpha_i^\theta \psi(v_i^\theta)$, and when $H \gg |\{\alpha_i^\theta\}_i|$, the kernel trick is computationally advantageous with few parameters.
- One question that arises is how to make this approach parametric; to maintain the kernel trick advantage without significantly increasing the computational cost when naively evaluating $\{\kappa(.,.)\}$.
- Solutions such as sampling and reduced set are both limited in this particular setting; sampling may generate a smaller (but biased) fixed set of support vectors $\{v_i^\theta\}_i$, while reduced set requires solving a difficult pre-image problem (Burges 1997).
- Our alternative, in this work, is to control the size of $\{v_i^\theta\}_i$ while allowing entries in $\{v_i^\theta\}_i$ to vary as a part of the end-to-end GCN (and also kernel) learning.
- This also allows modeling a larger class of filters $\{w_\theta\}$ that better fit the classification task at hand (see also experiments).

Neural Consistency and Architecture Design

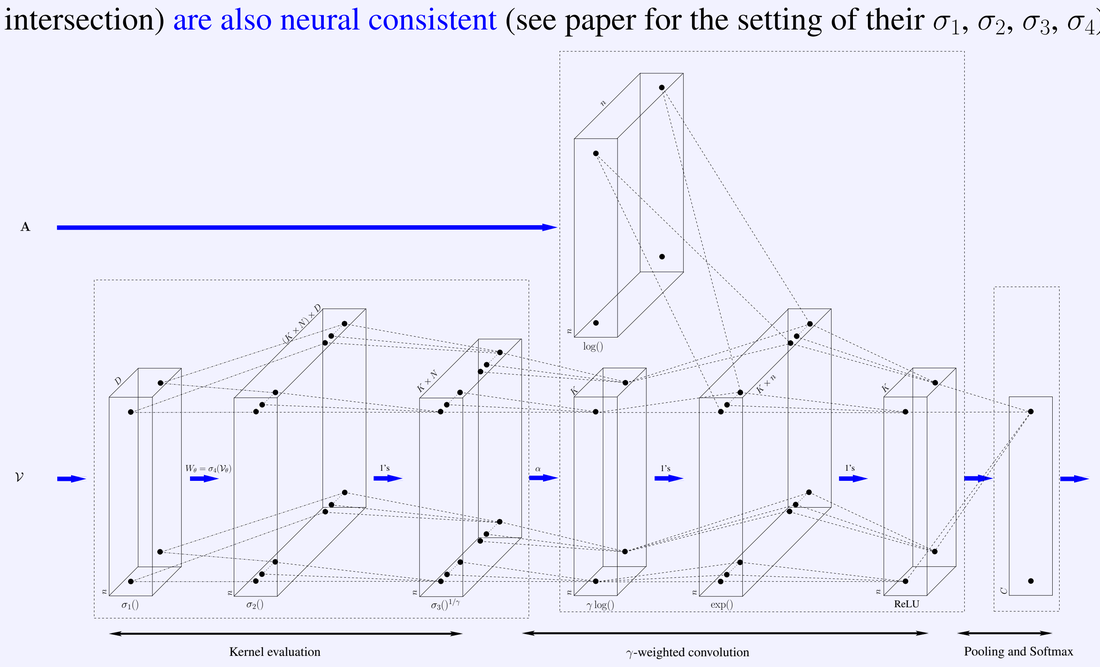
- In contrast to usual convolutional operators on graphs, the kernel one cannot be straightforwardly evaluated using standard neural units as kernels may have general forms.
- Hence, modeling requires adapting kernel-GCNs to the usual definition of neural units.

Definition 1 (Neural consistency) *Let $u_{.,d}$ (resp. $v_{.,d}$) denote the d^{th} dimension of the signal in a given node u (resp. v). For a given (fixed or learned) v, a kernel κ is referred to as "neural-consistent" if*

$$\kappa(u, v) = \sigma_3\left(\sum_d \sigma_2(\sigma_1(u_{.,d}).\omega_d) \right),$$

with $\omega_d = \sigma_4(v_{.,d})$ and being σ_1, σ_2, σ_3, σ_4 any arbitrary real-valued activation functions.

- For inner product-based kernels: linear $\langle u, v \rangle$, polynomial $\langle u, v \rangle^p$, and $\tanh(a\langle u, v \rangle + b)$ neural consistency is straightforward.
- For shift-invariant ones such as the gaussian, one may obtain neural consistency by rewriting $\exp(-\beta \|u - v\|_2^2) = \sigma_3\big(\sum_d \sigma_2(\sigma_1(u_{.,d}).\omega_d)\big)$ with $\sigma_1(.) = \exp(.)$, $\sigma_2(.) = \log(.)^2$, $\sigma_3(.) = \exp(-\beta(.))$ and $\omega_d = \exp(-v_{.,d})$.
- Other kernels (including Laplacian, inverse multiquadric, power, Cauchy and histogram intersection) are also neural consistent (see paper for the setting of their σ_1, σ_2, σ_3, σ_4).



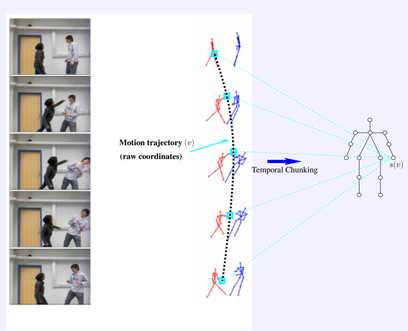
$$\mathbf{K}_\theta(u) = \frac{1}{|\mathcal{V}_\theta|} \sum_{u'} \exp\left(\log \mathbf{A}_{uu'} + \log \sum_{i=1}^N \alpha_i^\theta \, \kappa(u', v_i^\theta) \right).$$

Experiments

- **Evaluation Set (SBU):** 282 skeleton sequences acquired using the Microsoft Kinect sensor belonging to 8 categories. Skeleton representation is based on temporal chunking.

kernels \ GCNs	Standard GCN with different # of KPCA dimensions (H)										Our KGCN
	10	50	100	200	300	400	500	1000	2000	3000	
Linear	92.3077	overdim	overdim	overdim	overdim	overdim	overdim	overdim	overdim	overdim	90.7692
Poly	89.2308	95.3846	92.3077	93.8462	93.8462	93.8462	93.8462	overdim	overdim	overdim	93.8462
tanh	89.2308	93.8462	90.7692	93.8462	90.7692	92.3077	93.8462	92.3077	93.8462	92.3077	96.9231
sigmoid	93.8462	90.7692	93.8462	92.3077	92.3077	92.3077	92.3077	96.9231	93.8462	92.3077	95.3846
Gaussian	92.3077	92.3077	92.3077	92.3077	96.9231	93.8462	93.8462	93.8462	93.8462	93.8462	98.4615
Laplacian	92.3077	93.8462	95.3846	92.3077	90.7692	90.7692	95.3846	93.8462	90.7692	90.7692	98.4615
Power	90.7692	92.3077	95.3846	92.3077	92.3077	95.3846	95.3846	93.8462	93.8462	92.3077	96.9231
IMQ	87.6923	92.3077	95.3846	95.3846	93.8462	93.8462	90.7692	95.3846	93.8462	93.8462	95.3846
Log	93.8462	92.3077	92.3077	95.3846	93.8462	93.8462	95.3846	90.7692	95.3846	90.7692	96.9231
Cauchy	93.8462	95.3846	95.3846	92.3077	96.9231	93.8462	92.3077	95.3846	92.3077	93.8462	98.4615
HI	93.8462	92.3077	89.2308	90.7692	92.3077	92.3077	87.6923	87.6923	90.7692	87.6923	96.9231
time/epoch (s)	0.032	0.057	0.072	0.113	0.150	0.190	0.229	0.440	0.840	1.252	0.210

# of Filters (K) \ # of SVs (N)	1	4	8
1	84.6154	84.7552	85.1748
5	93.1469	**95.3846**	92.8671
10	92.1678	95.1049	95.1049



Method	Accuracy
GCNConv (Kipf et al. ICLR 2017)	90.00
ArmaConv (Bianchi et al, Arxiv 2019)	96.00
SGCConv (Wu et al, Arxiv 2019)	94.00
ChebyNet (Defferrard et al., NIPS 2016)	96.00
Raw coordinates (Yun et al., CVPR 2012)	49.7
Joint features (Yun et al., CVPR 2012)	80.3
Interact Pose (Ji et al., ICMEW 2014)	86.9
CHARM (Li et ak, ICCV 2015)	83.9
HBRNN-L (Du et al, CVPR 2015)	80.35
Co-occurence LSTM (Zhu et al. AAAI 2016)	90.41
ST-LSTM (Liu et al. ECCV 2016)	93.3
Topological pose ordering (Baradel et al. Arxiv 2017)	90.5
STA-LSTM (Song et al., AAAI 2017)	91.51
GCA-LSTM (Liu et al., IEEE TIP2018)	94.9
VA-LSTM (Zhang et al., ICCV 2017)	97.2
DeepGRU (Maghoumi et al., Arxiv 2018)	95.7
Riemannian manifold trajectory (Kacem et al., IEEE PAMI 2018)	93.7
Our best KGCN	98.43