Kernel-based Graph Convolutional Networks

Hichem SAHBI

CNRS Sorbonne University, Paris

ICPR 2020

Outline

- Introduction
- Kernel-based graph convolutional networks
- Experiments
- Conclusion

Outline

Motivation

- Graph convolutional networks (GCNs) aim at generalizing deep learning to arbitrary irregular domains.
- Existing spatial GCNs follow a neighborhood aggregation scheme.
- However, these convolutional operations are ill-posed (mainly translations and receptive fields) or weak to be discriminating.
- For highly nonlinear (and low dimensional) input graph signals (as 3D skeletons in action recognition), relying on input features is not enough.

Motivation

- Graph convolutional networks (GCNs) aim at generalizing deep learning to arbitrary irregular domains.
- Existing spatial GCNs follow a neighborhood aggregation scheme.
- However, these convolutional operations are ill-posed (mainly translations and receptive fields) or weak to be discriminating.
- For highly nonlinear (and low dimensional) input graph signals (as 3D skeletons in action recognition), relying on input features is not enough.

Motivation

- Graph convolutional networks (GCNs) aim at generalizing deep learning to arbitrary irregular domains.
- Existing spatial GCNs follow a neighborhood aggregation scheme.
- However, these convolutional operations are ill-posed (mainly translations and receptive fields) or weak to be discriminating.
- For highly nonlinear (and low dimensional) input graph signals (as 3D skeletons in action recognition), relying on input features is not enough.

Motivation

- Graph convolutional networks (GCNs) aim at generalizing deep learning to arbitrary irregular domains.
- Existing spatial GCNs follow a neighborhood aggregation scheme.
- However, these convolutional operations are ill-posed (mainly translations and receptive fields) or weak to be discriminating.
- For highly nonlinear (and low dimensional) input graph signals (as 3D skeletons in action recognition), relying on input features is not enough.

Contribution : dual (kernel-based) GCNs

- We consider, instead, an implicit mapping of the input graph signal in a RKHS as in kernel machines.
- The method achieves aggregation and convolution in that space, without increasing the number of training parameters.
- Our GCN model is able to achieve convolutions without explicitly realigning nodes in the receptive fields of the learned graph filters with those of the input graphs.
- 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.

Contribution : dual (kernel-based) GCNs

- We consider, instead, an implicit mapping of the input graph signal in a RKHS as in kernel machines.
- The method achieves aggregation and convolution in that space, without increasing the number of training parameters.
- Our GCN model is able to achieve convolutions without explicitly realigning nodes in the receptive fields of the learned graph filters with those of the input graphs.
- 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.

Contribution : dual (kernel-based) GCNs

- We consider, instead, an implicit mapping of the input graph signal in a RKHS as in kernel machines.
- The method achieves aggregation and convolution in that space, without increasing the number of training parameters.
- Our GCN model is able to achieve convolutions without explicitly realigning nodes in the receptive fields of the learned graph filters with those of the input graphs.
- 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.

Contribution : dual (kernel-based) GCNs

- We consider, instead, an implicit mapping of the input graph signal in a RKHS as in kernel machines.
- The method achieves aggregation and convolution in that space, without increasing the number of training parameters.
- Our GCN model is able to achieve convolutions without explicitly realigning nodes in the receptive fields of the learned graph filters with those of the input graphs.
- 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.

Outline

Standard vs kernel GCNs (I)

- Consider $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a graph endowed with (i) a signal $\{s(u) \in \mathbb{R}^D\}_u$ and (ii) an adjacency matrix $\mathbf{A}$, and $g_\theta = (\mathcal{V}_\theta, \mathcal{G}_\theta)$ as a graph filter :

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

- In spite of being agnostic to arbitrary node permutations, the above definition suffers from limited discrimination power.
- Kernel GCN : considering κ as a symmetric p.s.d function (i.e., $\exists \psi : \mathcal{X} \to \mathcal{H}$, 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 (Wahba71, Scholkopf01)

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

Standard vs kernel GCNs (I)

- Consider $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a graph endowed with (i) a signal $\{s(u) \in \mathbb{R}^D\}_u$ and (ii) an adjacency matrix $\mathbf{A}$, and $g_\theta = (\mathcal{V}_\theta, \mathcal{G}_\theta)$ as a graph filter :

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

- In spite of being agnostic to arbitrary node permutations, the above definition suffers from limited discrimination power.
- Kernel GCN : considering κ as a symmetric p.s.d function (i.e., $\exists \psi : \mathcal{X} \to \mathcal{H}$, 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 (Wahba71, Scholkopf01)

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

Standard vs kernel GCNs (I)

- Consider $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a graph endowed with (i) a signal $\{s(u) \in \mathbb{R}^D\}_u$ and (ii) an adjacency matrix $\mathbf{A}$, and $g_\theta = (\mathcal{V}_\theta, \mathcal{G}_\theta)$ as a graph filter :

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

- In spite of being agnostic to arbitrary node permutations, the above definition suffers from limited discrimination power.
- Kernel GCN : considering κ as a symmetric p.s.d function (i.e., $\exists \psi : \mathcal{X} \to \mathcal{H}$, 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 (Wahba71, Scholkopf01)

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

Standard vs kernel GCNs (I)

- Consider $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a graph endowed with (i) a signal $\{s(u) \in \mathbb{R}^D\}_u$ and (ii) an adjacency matrix $\mathbf{A}$, and $g_\theta = (\mathcal{V}_\theta, \mathcal{G}_\theta)$ as a graph filter :

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

- In spite of being agnostic to arbitrary node permutations, the above definition suffers from limited discrimination power.
- Kernel GCN : considering κ as a symmetric p.s.d function (i.e., $\exists \psi : \mathcal{X} \to \mathcal{H}$, 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 (Wahba71, Scholkopf01)

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

Standard vs. kernel GCNs (II)

- The strength of this kernel trick resides in its capacity 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 $\psi(s(u)) = s(u) \otimes \cdots \otimes s(u)$ (Maji12, Vedaldi12, Sahbi15).
- As H grows exponentially w.r.t p and polynomially w.r.t D, the kernel form is rather computationally more efficient.
- We control the size of $w_\theta = \frac{1}{|\mathcal{V}_\theta|} \sum_i \alpha_i^\theta \psi(v_i^\theta)$ while allowing entries in $\{v_i^\theta\}_i$ and $\{\alpha_i^\theta\}_i$ to vary as a part of the end-to-end GCN (and also kernel) learning.

Standard vs. kernel GCNs (II)

- The strength of this kernel trick resides in its capacity 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 $\psi(s(u)) = s(u) \otimes \cdots \otimes s(u)$ (Maji12, Vedaldi12, Sahbi15).
- As H grows exponentially w.r.t p and polynomially w.r.t D, the kernel form is rather computationally more efficient.
- We control the size of $w_\theta = \frac{1}{|\mathcal{V}_\theta|} \sum_i \alpha_i^\theta \psi(v_i^\theta)$ while allowing entries in $\{v_i^\theta\}_i$ and $\{\alpha_i^\theta\}_i$ to vary as a part of the end-to-end GCN (and also kernel) learning.

Standard vs. kernel GCNs (II)

- The strength of this kernel trick resides in its capacity 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 $\psi(s(u)) = s(u) \otimes \cdots \otimes s(u)$ (Maji12, Vedaldi12, Sahbi15).
- As H grows exponentially w.r.t p and polynomially w.r.t D, the kernel form is rather computationally more efficient.
- We control the size of $w_\theta = \frac{1}{|\mathcal{V}_\theta|} \sum_i \alpha_i^\theta \psi(v_i^\theta)$ while allowing entries in $\{v_i^\theta\}_i$ and $\{\alpha_i^\theta\}_i$ to vary as a part of the end-to-end GCN (and also kernel) learning.

Standard vs. kernel GCNs (II)

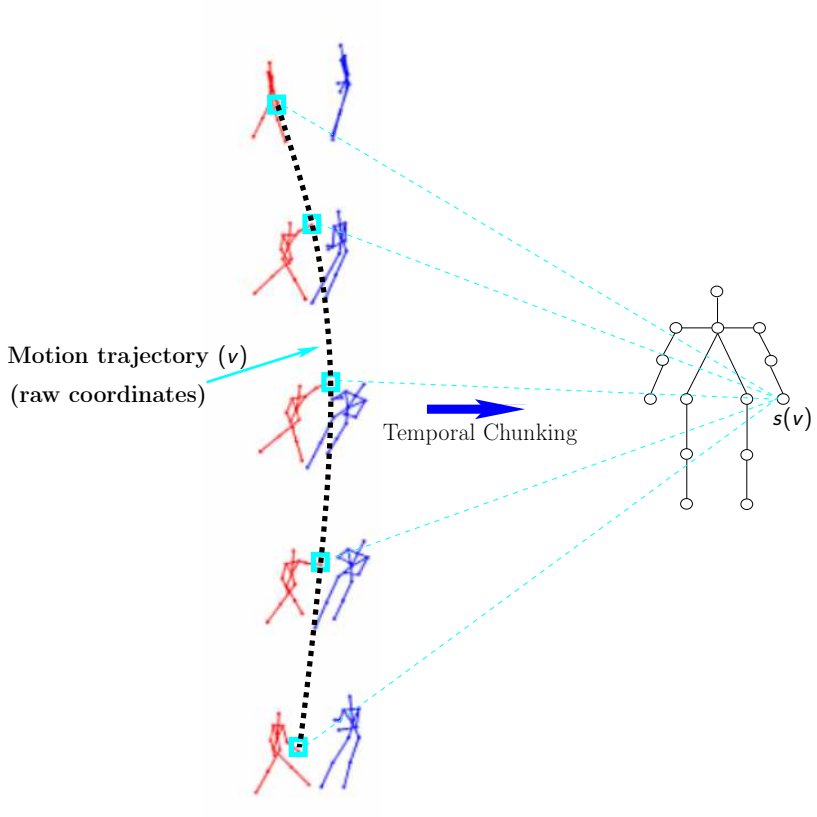
- The strength of this kernel trick resides in its capacity to handle nonlinear data as node representations are mapped into a high dimensional (and more discriminating) space $\mathcal{H} = \mathbb{R}^H$.
- E.g., the polynomial $\kappa(s(u), s(v)) = \langle s(u), s(v) \rangle^p$, its mapping is $\psi(s(u)) = s(u) \otimes \cdots \otimes s(u)$ (Maji12, Vedaldi12, Sahbi15).
- As H grows exponentially w.r.t p and polynomially w.r.t D, the kernel form is rather computationally more efficient.
- We control the size of $w_\theta = \frac{1}{|\mathcal{V}_\theta|} \sum_i \alpha_i^\theta \psi(v_i^\theta)$ while allowing entries in $\{v_i^\theta\}_i$ and $\{\alpha_i^\theta\}_i$ to vary as a part of the end-to-end GCN (and also kernel) learning.

Outline

Database and settings

- We evaluate our kernel-based GCN (KGCN) on the task of action recognition, using the SBU kinect dataset.
- This is an interaction dataset acquired using the Microsoft kinect sensor ; it includes in total 282 video sequences belonging to $C = 8$ categories with variable duration, viewpoint changes and interacting individuals.
- In all these experiments, we use the same evaluation protocol as the one suggested in (SBU12) (i.e., train-test split) and we report the average accuracy over all the classes of actions.
- We trained KGCN for 3000 epochs, with a batch size of 50, a momentum of 0.9 and a learning rate ν that decreases as $\nu \leftarrow \nu \times 0.99$ (resp. increases as $\nu \leftarrow \nu/0.99$).

Input skeleton graphs



Performances

GCNs / kernels	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

Ablation Study

KGCNs / kernels	Fixed-SV / Learned-α	Learned-SV / Fixed-α	Learned-SV / Learned-α
Linear	89.2308	90.7692	90.7692
Polynomial	84.6154	90.7692	93.8462
tanh	87.6923	90.7692	96.9231
Sigmoid	95.3846	95.3846	95.3846
Gaussian	84.6154	93.8462	98.4615
Laplacian	84.6154	93.8462	98.4615
Power	92.3077	95.3846	96.9231
I. Multi-quadric	81.5385	93.8462	95.3846
Log	84.6154	90.7692	96.9231
Cauchy	86.1538	92.3077	98.4615
HI	86.1538	95.3846	96.9231

Comparison

Methods	Perfs
GCNConv [57]	90.00
ArmaConv [61]	96.00
SGCConv [59]	94.00
ChebyNet [58]	96.00
Raw coordinates [53]	49.7
Joint features [53]	80.3
Interact Pose [62]	86.9
CHARM [63]	83.9
HBRNN-L [64]	80.35
Co-occurrence LSTM [66]	90.41
ST-LSTM [67]	93.3
Topological pose ordering[70]	90.5
STA-LSTM [56]	91.51
GCA-LSTM [55]	94.9
VA-LSTM [68]	97.2
DeepGRU [54]	95.7
Riemannian manifold trajectory[69]	93.7
Our best KGCN model	**98.46**

Outline

Conclusion

- We introduce in this paper a kernel-based GCN that defines convolutional graph filters in a high dimensional RKHS.
- The proposed kernel (dual) GCN formulation provides an effective way to enhance the discrimination power of the learned graph representations and it overtakes standard (primal) GCN approaches as well as the related work.
- As a future work, we are currently investigating the combination of explicit node expansion with implicit kernel mapping, in order to further enhance the generalization performances.