Kernel-based Graph Convolutional Networks

Hichem SAHBI

CNRS Sorbonne University, Paris

ICPR 2020
Outline

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

1. Introduction
2. Kernel-based graph convolutional networks
3. Experiments
4. Conclusion
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.
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

1. Introduction
2. Kernel-based graph convolutional networks
3. Experiments
4. Conclusion
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 $A$, and $g_\theta = (\mathcal{V}_\theta, \mathcal{G}_\theta)$ as a graph filter:

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

In spite of being agnostic to arbitrary node permutations, the above definition suffers from limited discrimination power.

Kernel GCN: considering $\kappa$ as a symmetric p.s.d 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 $w_\theta$ 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)

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

Hichem SAHBI

Kernel-based Graph Convolutional Networks
Standard vs kernel GCNs (I)

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

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

- In spite of being agnostic to arbitrary node permutations, the above definition suffers from limited discrimination power.

- **Kernel GCN**: considering $\kappa$ as a symmetric p.s.d 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 $w_\theta$ as $\frac{1}{|V_\theta|} \sum_{i=1}^{N} \alpha_i \psi(s(v_i^\theta))$ related to the representer theorem (Wahba71, Scholkopf01)

$$K_\theta(u) = \frac{1}{|N_r(u)|.|V_\theta|} \sum_{u' \in N_r(u)} \left( \sum_{i=1}^{N} \alpha_i \kappa(u', v_i^\theta) \right).$$
Standard vs kernel GCNs (I)

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

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

- In spite of being agnostic to arbitrary node permutations, the above definition suffers from limited discrimination power.

- **Kernel GCN**: considering $\kappa$ 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 $w_\theta$ 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)

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

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

In spite of being agnostic to arbitrary node permutations, the above definition suffers from limited discrimination power.

**Kernel GCN**: considering $\kappa$ as a symmetric p.s.d 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 $w_\theta$ 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)

$$K_\theta(u) = \frac{1}{|\mathcal{N}_r(u)|.|\mathcal{V}_\theta|} \sum_{u' \in \mathcal{N}_r(u)} \left( \sum_{i=1}^{N} \alpha_i^\theta \kappa(u', v_i^\theta) \right).$$
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 $w_\theta = \frac{1}{|V_\theta|} \sum_i \alpha_i^\theta \psi(v_i^\theta)$ 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.
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 $w_\theta = \frac{1}{|V_\theta|} \sum_i \alpha_i^\theta \psi(v_i^\theta)$ 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.
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 $w_\theta = \frac{1}{|V_\theta|} \sum_i \alpha_i \psi(v_i)$ while allowing entries in $\{v_i\}$ and $\{\alpha_i\}$ 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 $w_\theta = \frac{1}{|V_\theta|} \sum_i \alpha^\theta_i \psi(v^\theta_i)$ while allowing entries in $\{v^\theta_i\}_i$ and $\{\alpha^\theta_i\}_i$ to vary as a part of the end-to-end GCN (and also kernel) learning.
Outline

1. Introduction
2. Kernel-based graph convolutional networks
3. Experiments
4. Conclusion
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 $\nu$ that decreases as $\nu \leftarrow \nu \times 0.99$ (resp. increases as $\nu \leftarrow \nu / 0.99$).
Input skeleton graphs

Motion trajectory \( (v) \)
(raw coordinates)

Temporal Chunking

\( s(v) \)
## Performances

<table>
<thead>
<tr>
<th>kernels</th>
<th>GCNs</th>
<th>Standard GCN with different # of KPCA dimensions (H)</th>
<th>Our KGCN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>Linear</td>
<td></td>
<td>92.3077</td>
<td>overdim</td>
</tr>
<tr>
<td>Poly</td>
<td></td>
<td>89.2308</td>
<td>95.3846</td>
</tr>
<tr>
<td>tanh</td>
<td></td>
<td>89.2308</td>
<td>93.8462</td>
</tr>
<tr>
<td>sigmoid</td>
<td></td>
<td>93.8462</td>
<td>90.7692</td>
</tr>
<tr>
<td>Gaussian</td>
<td></td>
<td>92.3077</td>
<td>92.3077</td>
</tr>
<tr>
<td>Laplacian</td>
<td></td>
<td>92.3077</td>
<td>93.8462</td>
</tr>
<tr>
<td>Power</td>
<td></td>
<td>90.7692</td>
<td>92.3077</td>
</tr>
<tr>
<td>IMQ</td>
<td></td>
<td>87.6923</td>
<td>92.3077</td>
</tr>
<tr>
<td>Log</td>
<td></td>
<td>93.8462</td>
<td>92.3077</td>
</tr>
<tr>
<td>Cauchy</td>
<td></td>
<td>93.8462</td>
<td>95.3846</td>
</tr>
<tr>
<td>HI</td>
<td></td>
<td>93.8462</td>
<td>92.3077</td>
</tr>
</tbody>
</table>

| 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 |

<table>
<thead>
<tr>
<th># of Filters (K)</th>
<th># of SVs (N)</th>
<th>1</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>84.6154</td>
<td>84.7552</td>
<td>85.1748</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>93.1469</td>
<td>95.3846</td>
<td>92.8671</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>92.1678</td>
<td>95.1049</td>
<td>95.1049</td>
</tr>
</tbody>
</table>
### Ablation Study

<table>
<thead>
<tr>
<th>KGCNs</th>
<th>Fixed-SV / Learned-α</th>
<th>Learned-SV / Fixed-α</th>
<th>Learned-SV / Learned-α</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>89.2308</td>
<td>90.7692</td>
<td>90.7692</td>
</tr>
<tr>
<td>Polynomial</td>
<td>84.6154</td>
<td>90.7692</td>
<td>93.8462</td>
</tr>
<tr>
<td>tanh</td>
<td>87.6923</td>
<td>90.7692</td>
<td>96.9231</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>95.3846</td>
<td>95.3846</td>
<td>95.3846</td>
</tr>
<tr>
<td>Gaussian</td>
<td>84.6154</td>
<td>93.8462</td>
<td>98.4615</td>
</tr>
<tr>
<td>Laplacian</td>
<td>84.6154</td>
<td>93.8462</td>
<td>98.4615</td>
</tr>
<tr>
<td>Power</td>
<td>92.3077</td>
<td>95.3846</td>
<td>96.9231</td>
</tr>
<tr>
<td>I. Multi-quadric</td>
<td>81.5385</td>
<td>93.8462</td>
<td>95.3846</td>
</tr>
<tr>
<td>Log</td>
<td>84.6154</td>
<td>90.7692</td>
<td>96.9231</td>
</tr>
<tr>
<td>Cauchy</td>
<td>86.1538</td>
<td>92.3077</td>
<td>98.4615</td>
</tr>
<tr>
<td>HI</td>
<td>86.1538</td>
<td>95.3846</td>
<td>96.9231</td>
</tr>
</tbody>
</table>
## Comparison

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

1. Introduction
2. Kernel-based graph convolutional networks
3. Experiments
4. Conclusion
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.