Learning Connectivity with Graph Convolutional Networks

Hichem SAHBI

CNRS Sorbonne University, Paris

ICPR 2020

Outline

- Introduction
- Learning connectivity in GCNs
- 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, and its success is reliant on the topology (structure) of input graphs.
- However, graph structures (either available or handcrafted) are powerless to optimally capture all the relationships between nodes as their setting is oblivious to the targeted applications.
- E.g., node-to-node relationships, in human skeletons, capture the intrinsic anthropometric characteristics of individuals (useful for their identification) while other connections, yet to infer, are necessary for recognizing their dynamics and actions.

Motivation

- Graph convolutional networks (GCNs) aim at generalizing deep learning to arbitrary irregular domains.
- Existing spatial GCNs follow a neighborhood aggregation scheme, and its success is reliant on the topology (structure) of input graphs.
- However, graph structures (either available or handcrafted) are powerless to optimally capture all the relationships between nodes as their setting is oblivious to the targeted applications.
- E.g., node-to-node relationships, in human skeletons, capture the intrinsic anthropometric characteristics of individuals (useful for their identification) while other connections, yet to infer, are necessary for recognizing their dynamics and actions.

Motivation

- Graph convolutional networks (GCNs) aim at generalizing deep learning to arbitrary irregular domains.
- Existing spatial GCNs follow a neighborhood aggregation scheme, and its success is reliant on the topology (structure) of input graphs.
- However, graph structures (either available or handcrafted) are powerless to optimally capture all the relationships between nodes as their setting is oblivious to the targeted applications.
- E.g., node-to-node relationships, in human skeletons, capture the intrinsic anthropometric characteristics of individuals (useful for their identification) while other connections, yet to infer, are necessary for recognizing their dynamics and actions.

Motivation

- Graph convolutional networks (GCNs) aim at generalizing deep learning to arbitrary irregular domains.
- Existing spatial GCNs follow a neighborhood aggregation scheme, and its success is reliant on the topology (structure) of input graphs.
- However, graph structures (either available or handcrafted) are powerless to optimally capture all the relationships between nodes as their setting is oblivious to the targeted applications.
- E.g., node-to-node relationships, in human skeletons, capture the intrinsic anthropometric characteristics of individuals (useful for their identification) while other connections, yet to infer, are necessary for recognizing their dynamics and actions.

Contribution : learning connectivity in GCNs

- We introduce a novel framework that learns convolutional filters on graphs together with their topological properties.
- The latter are modeled through matrix operators that capture multiple aggregates on graphs, learned using a constrained cross-entropy loss.
- We consider different *constraints* (including stochasticity, orthogonality and symmetry) acting as regularizers.
- Stochasticity implements random walk Laplacians while orthogonality models multiple aggregation operators with non-overlapping supports ; it also avoids redundancy and oversizing the learned GCNs with useless parameters. Symmetry reduces further the number of training parameters.

Contribution : learning connectivity in GCNs

- We introduce a novel framework that learns convolutional filters on graphs together with their topological properties.
- The latter are modeled through matrix operators that capture multiple aggregates on graphs, learned using a constrained cross-entropy loss.
- We consider different *constraints* (including stochasticity, orthogonality and symmetry) acting as regularizers.
- Stochasticity implements random walk Laplacians while orthogonality models multiple aggregation operators with non-overlapping supports ; it also avoids redundancy and oversizing the learned GCNs with useless parameters. Symmetry reduces further the number of training parameters.

Contribution : learning connectivity in GCNs

- We introduce a novel framework that learns convolutional filters on graphs together with their topological properties.
- The latter are modeled through matrix operators that capture multiple aggregates on graphs, learned using a constrained cross-entropy loss.
- We consider different *constraints* (including stochasticity, orthogonality and symmetry) acting as regularizers.
- Stochasticity implements random walk Laplacians while orthogonality models multiple aggregation operators with non-overlapping supports ; it also avoids redundancy and oversizing the learned GCNs with useless parameters. Symmetry reduces further the number of training parameters.

Contribution : learning connectivity in GCNs

- We introduce a novel framework that learns convolutional filters on graphs together with their topological properties.
- The latter are modeled through matrix operators that capture multiple aggregates on graphs, learned using a constrained cross-entropy loss.
- We consider different *constraints* (including stochasticity, orthogonality and symmetry) acting as regularizers.
- Stochasticity implements random walk Laplacians while orthogonality models multiple aggregation operators with non-overlapping supports ; it also avoids redundancy and oversizing the learned GCNs with useless parameters. Symmetry reduces further the number of training parameters.

Outline

Problem statement

- Considering $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ endowed with $\{s(u)\}_u$ and (ii) $\mathbf{A}$,

$$(\mathcal{G} \star \mathcal{F})_{\mathcal{V}} = f\big(\mathbf{A}\ \mathbf{U}^\top\ \mathbf{W}\big).$$

- Here $\mathbf{A}\mathbf{U}^\top$ acts as a feature extractor which collects non-differential and differential statistics including means and variances of node neighbors, before applying convolutions.
- We use the chain rule in order to derive the gradient $\frac{\partial E}{\partial \mathbf{vec}(\mathbf{A})}$ and update $\mathbf{A}$ using SGD ; we upgrade the latter to learn both the convolutional parameters $\mathbf{W}$ together with $\mathbf{A}$ while implementing *orthogonality, stochasticity and symmetry*.
- Orthogonality allows us to design $\mathbf{A}$ with a minimum number of parameters, stochasticity normalizes nodes by their degrees and allows learning random walk Laplacians, while symmetry reduces further the number of training parameters.

Problem statement

- Considering $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ endowed with $\{s(u)\}_u$ and (ii) $\mathbf{A}$,

$$(\mathcal{G} \star \mathcal{F})_{\mathcal{V}} = f\big(\mathbf{A}\ \mathbf{U}^\top\ \mathbf{W}\big).$$

- Here $\mathbf{A}\mathbf{U}^\top$ acts as a feature extractor which collects non-differential and differential statistics including means and variances of node neighbors, before applying convolutions.
- We use the chain rule in order to derive the gradient $\frac{\partial E}{\partial \mathbf{vec}(\mathbf{A})}$ and update $\mathbf{A}$ using SGD ; we upgrade the latter to learn both the convolutional parameters $\mathbf{W}$ together with $\mathbf{A}$ while implementing *orthogonality, stochasticity and symmetry*.
- Orthogonality allows us to design $\mathbf{A}$ with a minimum number of parameters, stochasticity normalizes nodes by their degrees and allows learning random walk Laplacians, while symmetry reduces further the number of training parameters.

Problem statement

- Considering $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ endowed with $\{s(u)\}_u$ and (ii) $\mathbf{A}$,

$$(\mathcal{G} \star \mathcal{F})_{\mathcal{V}} = f\big(\mathbf{A}\ \mathbf{U}^\top\ \mathbf{W}\big).$$

- Here $\mathbf{AU}^\top$ acts as a feature extractor which collects non-differential and differential statistics including means and variances of node neighbors, before applying convolutions.
- We use the chain rule in order to derive the gradient $\frac{\partial E}{\partial \mathbf{vec}(\mathbf{A})}$ and update $\mathbf{A}$ using SGD ; we upgrade the latter to learn both the convolutional parameters $\mathbf{W}$ together with $\mathbf{A}$ while implementing *orthogonality, stochasticity and symmetry*.
- Orthogonality allows us to design $\mathbf{A}$ with a minimum number of parameters, stochasticity normalizes nodes by their degrees and allows learning random walk Laplacians, while symmetry reduces further the number of training parameters.

Problem statement

- Considering $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ endowed with $\{s(u)\}_u$ and (ii) $\mathbf{A}$,

$$(\mathcal{G} \star \mathcal{F})_{\mathcal{V}} = f\big(\mathbf{A}\ \mathbf{U}^\top\ \mathbf{W}\big).$$

- Here $\mathbf{AU}^\top$ acts as a feature extractor which collects non-differential and differential statistics including means and variances of node neighbors, before applying convolutions.
- We use the chain rule in order to derive the gradient $\frac{\partial E}{\partial \mathbf{vec}(\mathbf{A})}$ and update $\mathbf{A}$ using SGD ; we upgrade the latter to learn both the convolutional parameters $\mathbf{W}$ together with $\mathbf{A}$ while implementing *orthogonality, stochasticity and symmetry*.
- Orthogonality allows us to design $\mathbf{A}$ with a minimum number of parameters, stochasticity normalizes nodes by their degrees and allows learning random walk Laplacians, while symmetry reduces further the number of training parameters.

Stochasticity

- Stochasticity requires adding equality and inequality constraints in SGD, i.e., $\mathbf{A}_{ij} \in [0,1]$ and $\sum_q \mathbf{A}_{qj} = 1$.
- We consider a reparametrization of the learned matrices, as $\mathbf{A}_{ij} = h(\hat{\mathbf{A}}_{ij}) / \sum_q h(\hat{\mathbf{A}}_{qj})$.
- During backpropagation, the gradient of the loss E (now w.r.t $\hat{\mathbf{A}}$) is updated using the chain rule as

$$\frac{\partial E}{\partial \hat{\mathbf{A}}_{ij}} = \sum_p \frac{\partial E}{\partial \mathbf{A}_{pj}} . \frac{\partial \mathbf{A}_{pj}}{\partial \hat{\mathbf{A}}_{ij}} \quad \text{with} \quad \frac{\partial \mathbf{A}_{pj}}{\partial \hat{\mathbf{A}}_{ij}} = \frac{h'(\hat{\mathbf{A}}_{ij})}{\sum_q h(\hat{\mathbf{A}}_{qj})} . (\delta_{pi} - \mathbf{A}_{pj}).$$

- In practice $h(.) = \exp(.)$ and the new gradient (w.r.t $\hat{\mathbf{A}}$) is obtained by multiplying the original one by the Jacobian $\mathbf{J}_{\text{stc}} = \left[\frac{\partial \mathbf{A}_{pj}}{\partial \hat{\mathbf{A}}_{ij}}\right]_{p,i=1}^{n}$ which merely reduces to $[\mathbf{A}_{ij} . (\delta_{pi} - \mathbf{A}_{pj})]_{p,i}$.

Stochasticity

- Stochasticity requires adding equality and inequality constraints in SGD, i.e., $\mathbf{A}_{ij} \in [0,1]$ and $\sum_q \mathbf{A}_{qj} = 1$.
- We consider a reparametrization of the learned matrices, as $\mathbf{A}_{ij} = h(\hat{\mathbf{A}}_{ij}) / \sum_q h(\hat{\mathbf{A}}_{qj})$.
- During backpropagation, the gradient of the loss E (now w.r.t $\hat{\mathbf{A}}$) is updated using the chain rule as

$$\frac{\partial E}{\partial \hat{\mathbf{A}}_{ij}} = \sum_p \frac{\partial E}{\partial \mathbf{A}_{pj}} . \frac{\partial \mathbf{A}_{pj}}{\partial \hat{\mathbf{A}}_{ij}} \quad \text{with} \quad \frac{\partial \mathbf{A}_{pj}}{\partial \hat{\mathbf{A}}_{ij}} = \frac{h'(\hat{\mathbf{A}}_{ij})}{\sum_q h(\hat{\mathbf{A}}_{qj})} . (\delta_{pi} - \mathbf{A}_{pj}).$$

- In practice $h(.) = \exp(.)$ and the new gradient (w.r.t $\hat{\mathbf{A}}$) is obtained by multiplying the original one by the Jacobian $\mathbf{J}_{\text{stc}} = \left[\frac{\partial \mathbf{A}_{pj}}{\partial \hat{\mathbf{A}}_{ij}}\right]_{p,i=1}^{n}$ which merely reduces to $[\mathbf{A}_{ij} . (\delta_{pi} - \mathbf{A}_{pj})]_{p,i}$.

Stochasticity

- Stochasticity requires adding equality and inequality constraints in SGD, i.e., $\mathbf{A}_{ij} \in [0,1]$ and $\sum_q \mathbf{A}_{qj} = 1$.
- We consider a reparametrization of the learned matrices, as $\mathbf{A}_{ij} = h(\hat{\mathbf{A}}_{ij}) / \sum_q h(\hat{\mathbf{A}}_{qj})$.
- During backpropagation, the gradient of the loss E (now w.r.t $\hat{\mathbf{A}}$) is updated using the chain rule as

$$\frac{\partial E}{\partial \hat{\mathbf{A}}_{ij}} = \sum_p \frac{\partial E}{\partial \mathbf{A}_{pj}} . \frac{\partial \mathbf{A}_{pj}}{\partial \hat{\mathbf{A}}_{ij}} \quad \text{with} \quad \frac{\partial \mathbf{A}_{pj}}{\partial \hat{\mathbf{A}}_{ij}} = \frac{h'(\hat{\mathbf{A}}_{ij})}{\sum_q h(\hat{\mathbf{A}}_{qj})} . (\delta_{pi} - \mathbf{A}_{pj}).$$

- In practice $h(.) = \exp(.)$ and the new gradient (w.r.t $\hat{\mathbf{A}}$) is obtained by multiplying the original one by the Jacobian $\mathbf{J}_{\text{stc}} = \left[\frac{\partial \mathbf{A}_{pj}}{\partial \hat{\mathbf{A}}_{ij}}\right]_{p,i=1}^{n}$ which merely reduces to $[\mathbf{A}_{ij} . (\delta_{pi} - \mathbf{A}_{pj})]_{p,i}$.

Stochasticity

- Stochasticity requires adding equality and inequality constraints in SGD, i.e., $\mathbf{A}_{ij} \in [0,1]$ and $\sum_q \mathbf{A}_{qj} = 1$.
- We consider a reparametrization of the learned matrices, as $\mathbf{A}_{ij} = h(\hat{\mathbf{A}}_{ij}) / \sum_q h(\hat{\mathbf{A}}_{qj})$.
- During backpropagation, the gradient of the loss E (now w.r.t $\hat{\mathbf{A}}$) is updated using the chain rule as

$$\frac{\partial E}{\partial \hat{\mathbf{A}}_{ij}} = \sum_p \frac{\partial E}{\partial \mathbf{A}_{pj}} . \frac{\partial \mathbf{A}_{pj}}{\partial \hat{\mathbf{A}}_{ij}} \quad \text{with} \quad \frac{\partial \mathbf{A}_{pj}}{\partial \hat{\mathbf{A}}_{ij}} = \frac{h'(\hat{\mathbf{A}}_{ij})}{\sum_q h(\hat{\mathbf{A}}_{qj})} . (\delta_{pi} - \mathbf{A}_{pj}).$$

- In practice $h(.) = \exp(.)$ and the new gradient (w.r.t $\hat{\mathbf{A}}$) is obtained by multiplying the original one by the Jacobian $\mathbf{J}_{\text{stc}} = \left[\frac{\partial \mathbf{A}_{pj}}{\partial \hat{\mathbf{A}}_{ij}}\right]_{p,i=1}^{n}$ which merely reduces to $[\mathbf{A}_{ij} . (\delta_{pi} - \mathbf{A}_{pj})]_{p,i}$.

Orthogonality

- Learning multiple $\{\mathbf{A}_k\}_k$ allows us to capture different graph topologies when achieving aggregation and convolution.

$$(\mathcal{G} \star \mathcal{F})_{\mathcal{V}} = f\left(\sum_{k=1}^{K} \mathbf{A}_k \mathbf{U}^\top \mathbf{W}_k\right)$$

$$\begin{array}{ll} \min_{\{\mathbf{A}_k\}_k, \mathbf{W}} & E\big(\mathbf{A}_1, \ldots, \mathbf{A}_K; \mathbf{W}\big) \\ \text{s.t.} & \mathbf{A}_k \odot \mathbf{A}_k > \mathbf{0}_n, \ \mathbf{A}_k \odot \mathbf{A}_{k'} = \mathbf{0}_n \quad \forall k, k' \neq k. \end{array}$$

- We consider $\exp(\gamma \hat{\mathbf{A}}_k) \oslash (\sum_{r=1}^{K} \exp(\gamma \hat{\mathbf{A}}_r))$ as a softmax reparametrization of $\mathbf{A}_k$, with $\{\hat{\mathbf{A}}_k\}_k$ free parameters in $\mathbb{R}^{n \times n}$.
- By choosing a large value of γ, it becomes possible to implement ϵ-orthogonality ; a surrogate property where only one entry $\mathbf{A}_{kij} \gg 0$ while all others $\{\mathbf{A}_{k'ij}\}_{k' \neq k}$ vanish.
- The setting of γ and updated Jacobians are in the paper.

Orthogonality

- Learning multiple $\{\mathbf{A}_k\}_k$ allows us to capture different graph topologies when achieving aggregation and convolution.

$$(\mathcal{G} \star \mathcal{F})_{\mathcal{V}} = f\left(\sum_{k=1}^{K} \mathbf{A}_k \mathbf{U}^\top \mathbf{W}_k\right)$$

$$\begin{array}{ll} \min_{\{\mathbf{A}_k\}_k, \mathbf{W}} & E\big(\mathbf{A}_1, \ldots, \mathbf{A}_K; \mathbf{W}\big) \\ \text{s.t.} & \mathbf{A}_k \odot \mathbf{A}_k > \mathbf{0}_n, \ \mathbf{A}_k \odot \mathbf{A}_{k'} = \mathbf{0}_n \quad \forall k, k' \neq k. \end{array}$$

- We consider $\exp(\gamma \hat{\mathbf{A}}_k) \oslash (\sum_{r=1}^{K} \exp(\gamma \hat{\mathbf{A}}_r))$ as a softmax reparametrization of $\mathbf{A}_k$, with $\{\hat{\mathbf{A}}_k\}_k$ free parameters in $\mathbb{R}^{n \times n}$.
- By choosing a large value of γ, it becomes possible to implement ϵ-orthogonality ; a surrogate property where only one entry $\mathbf{A}_{kij} \gg 0$ while all others $\{\mathbf{A}_{k'ij}\}_{k' \neq k}$ vanish.
- The setting of γ and updated Jacobians are in the paper.

Orthogonality

- Learning multiple $\{\mathbf{A}_k\}_k$ allows us to capture different graph topologies when achieving aggregation and convolution.

$$(\mathcal{G} \star \mathcal{F})_{\mathcal{V}} = f\left(\sum_{k=1}^{K} \mathbf{A}_k \mathbf{U}^\top \mathbf{W}_k\right)$$

$$\begin{array}{ll} \min_{\{\mathbf{A}_k\}_k, \mathbf{W}} & E\big(\mathbf{A}_1, \ldots, \mathbf{A}_K; \mathbf{W}\big) \\ \text{s.t.} & \mathbf{A}_k \odot \mathbf{A}_k > \mathbf{0}_n, \ \mathbf{A}_k \odot \mathbf{A}_{k'} = \mathbf{0}_n \quad \forall k, k' \neq k. \end{array}$$

- We consider $\exp(\gamma \hat{\mathbf{A}}_k) \oslash (\sum_{r=1}^{K} \exp(\gamma \hat{\mathbf{A}}_r))$ as a softmax reparametrization of $\mathbf{A}_k$, with $\{\hat{\mathbf{A}}_k\}_k$ free parameters in $\mathbb{R}^{n \times n}$.
- By choosing a large value of γ, it becomes possible to implement ϵ-orthogonality ; a surrogate property where only one entry $\mathbf{A}_{kij} \gg 0$ while all others $\{\mathbf{A}_{k'ij}\}_{k' \neq k}$ vanish.
- The setting of γ and updated Jacobians are in the paper.

Orthogonality

- Learning multiple $\{\mathbf{A}_k\}_k$ allows us to capture different graph topologies when achieving aggregation and convolution.

$$(\mathcal{G} \star \mathcal{F})_{\mathcal{V}} = f\left(\sum_{k=1}^{K} \mathbf{A}_k \mathbf{U}^\top \mathbf{W}_k\right)$$

$$\begin{array}{ll} \min_{\{\mathbf{A}_k\}_k, \mathbf{W}} & E\big(\mathbf{A}_1, \ldots, \mathbf{A}_K; \mathbf{W}\big) \\ \text{s.t.} & \mathbf{A}_k \odot \mathbf{A}_k > \mathbf{0}_n, \ \mathbf{A}_k \odot \mathbf{A}_{k'} = \mathbf{0}_n \quad \forall k, k' \neq k. \end{array}$$

- We consider $\exp(\gamma \hat{\mathbf{A}}_k) \oslash (\sum_{r=1}^{K} \exp(\gamma \hat{\mathbf{A}}_r))$ as a softmax reparametrization of $\mathbf{A}_k$, with $\{\hat{\mathbf{A}}_k\}_k$ free parameters in $\mathbb{R}^{n \times n}$.
- By choosing a large value of γ, it becomes possible to implement ϵ-orthogonality ; a surrogate property where only one entry $\mathbf{A}_{kij} \gg 0$ while all others $\{\mathbf{A}_{k'ij}\}_{k' \neq k}$ vanish.
- The setting of γ and updated Jacobians are in the paper.

Symmetry and combination

- Symmetry is guaranteed by considering the reparametrization of each matrix as $\mathbf{A}_k = \frac{1}{2}(\hat{\mathbf{A}}_k + \hat{\mathbf{A}}_k^\top)$ with $\hat{\mathbf{A}}_k$ being free.
- Symmetry is maintained by multiplying the original gradient $\frac{\partial E}{\partial \mathbf{vec}(\{\mathbf{A}_k\}_k)}$ by the Jacobian

$$\mathbf{J}_{\text{sym}} = \frac{1}{2}\left[1_{\{k=k'\}} \cdot 1_{\{(i=i',j=j')\vee(i=j',j=i')\}}\right]_{ijk,i'j'k'}.$$

- One may combine symmetry with all the aforementioned constraints by multiplying the underlying Jacobians, so the final gradient is obtained by multiplying the original one as

$$\frac{\partial E}{\partial \mathbf{vec}(\{\hat{\mathbf{A}}_k\}_k)} = \mathbf{J}_{(\text{sym or stc})} \cdot \mathbf{J}_{\text{orth}} \cdot \frac{\partial E}{\partial \mathbf{vec}(\{\mathbf{A}_k\}_k)}.$$

Symmetry and combination

- Symmetry is guaranteed by considering the reparametrization of each matrix as $\mathbf{A}_k = \frac{1}{2}(\hat{\mathbf{A}}_k + \hat{\mathbf{A}}_k^\top)$ with $\hat{\mathbf{A}}_k$ being free.
- Symmetry is maintained by multiplying the original gradient $\frac{\partial E}{\partial \mathbf{vec}(\{\mathbf{A}_k\}_k)}$ by the Jacobian

$$\mathbf{J}_{\text{sym}} = \frac{1}{2}\left[1_{\{k=k'\}}.1_{\{(i=i',j=j')\vee(i=j',j=i')\}}\right]_{ijk,i'j'k'}.$$

- One may combine symmetry with all the aforementioned constraints by multiplying the underlying Jacobians, so the final gradient is obtained by multiplying the original one as

$$\frac{\partial E}{\partial \mathbf{vec}(\{\hat{\mathbf{A}}_k\}_k)} = \mathbf{J}_{(\text{sym or stc})}.\mathbf{J}_{\text{orth}}.\frac{\partial E}{\partial \mathbf{vec}(\{\mathbf{A}_k\}_k)}.$$

Symmetry and combination

- Symmetry is guaranteed by considering the reparametrization of each matrix as $\mathbf{A}_k = \frac{1}{2}(\hat{\mathbf{A}}_k + \hat{\mathbf{A}}_k^\top)$ with $\hat{\mathbf{A}}_k$ being free.
- Symmetry is maintained by multiplying the original gradient $\frac{\partial E}{\partial \mathbf{vec}(\{\mathbf{A}_k\}_k)}$ by the Jacobian

$$\mathbf{J}_{\text{sym}} = \frac{1}{2}\left[1_{\{k=k'\}}.1_{\{(i=i',j=j')\vee(i=j',j=i')\}}\right]_{ijk,i'j'k'}.$$

- One may combine symmetry with all the aforementioned constraints by multiplying the underlying Jacobians, so the final gradient is obtained by multiplying the original one as

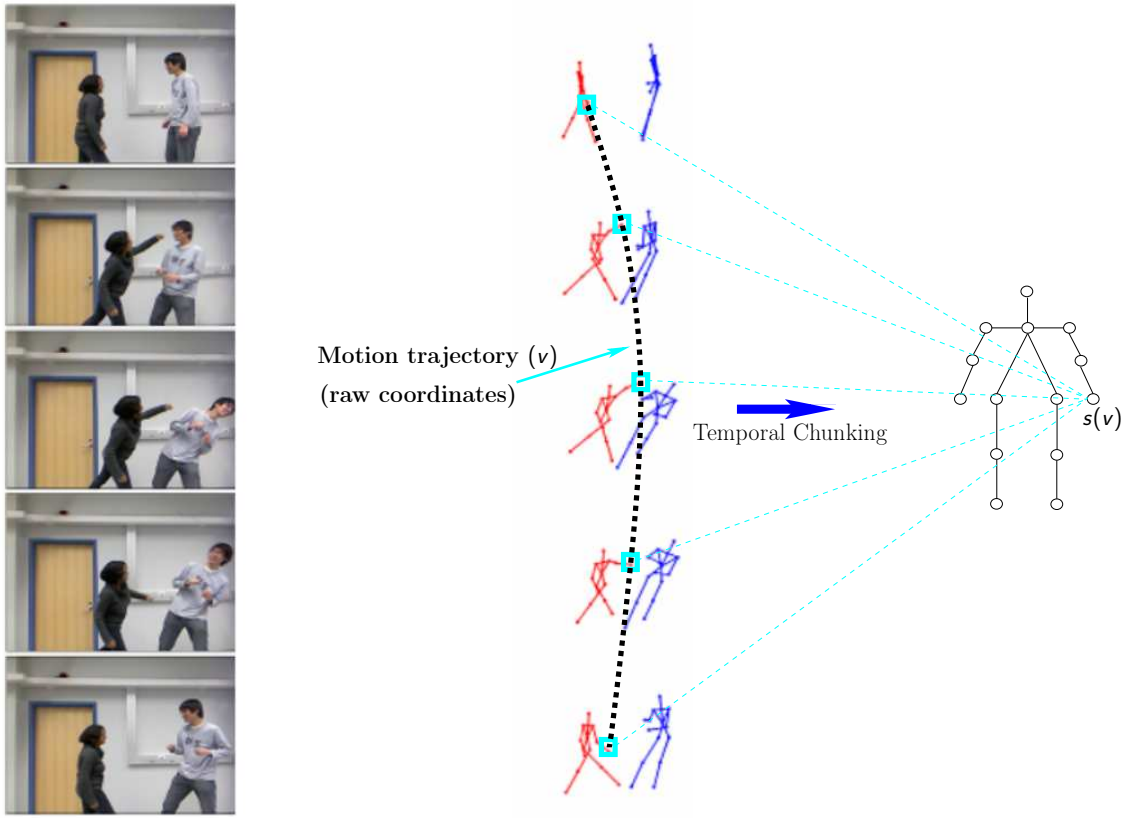
$$\frac{\partial E}{\partial \mathbf{vec}(\{\hat{\mathbf{A}}_k\}_k)} = \mathbf{J}_{(\text{sym or stc})}.\mathbf{J}_{\text{orth}}.\frac{\partial E}{\partial \mathbf{vec}(\{\mathbf{A}_k\}_k)}.$$

Outline

Database and settings

- We evaluate our GCN 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 our GCNs for 3000 epochs, with a batch size of 200, 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

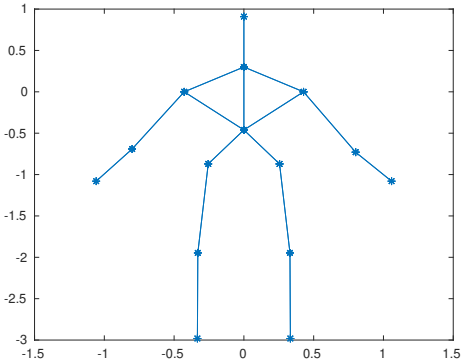


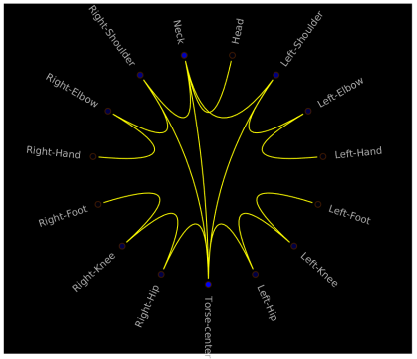
Performances

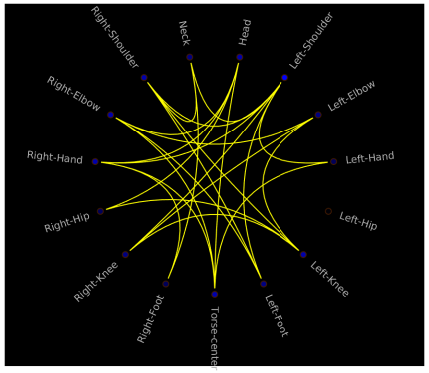
Oper \ Const		none	sym	orth	stc	sym+orth	orth+stc	Mean
HPM.	$K = 1$	89.2308	92.3077	–	89.2308	–	–	90.2564
	$K = 4$	87.6923	89.2308	89.2308	87.6923	90.7692	92.3077	89.4872
	$K = 8$	90.7692	95.3846	92.3077	90.7692	92.3077	92.3077	92.3077
	Mean	89.2308	92.3077	90.7692	89.2308	91.5384	92.3077	90.7692
LPM.	$K = 1$	92.3077	87.6923	–	95.3846	–	–	91.7949
	$K = 4$	92.3077	92.3077	93.8462	95.3846	90.7692	96.9231	93.5897
	$K = 8$	95.3846	90.7692	87.6923	93.8462	93.8462	92.3077	92.3077
	Mean	93.3333	90.2564	90.7692	94.8718	92.3077	94.6154	92.7180
Our	$K = 1$	95.3846	93.8462	–	95.3846	–	–	94.8718
	$K = 4$	93.8462	95.3846	95.3846	96.9231	93.8462	**98.4615**	95.6410
	$K = 8$	92.3077	93.8462	95.3846	90.7692	95.3846	90.7692	93.0769
	Mean	93.8462	94.3590	95.3846	94.3590	94.6154	94.6154	94.4615

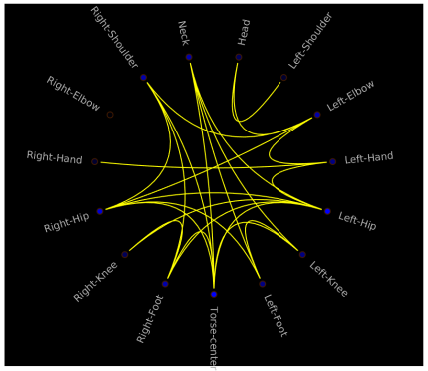
HPM : handcrafted power map connectivity, LPM : learned power map connectivity, sym : symmetry, orth : orthogonality, stc : stochasticity.

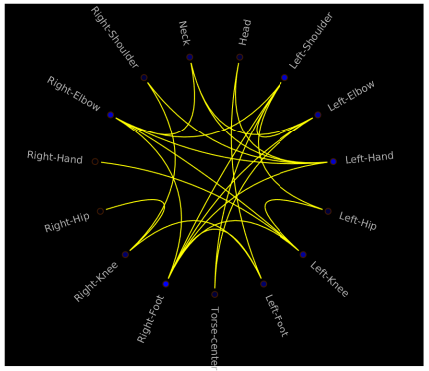
Learned connectivity (examples)

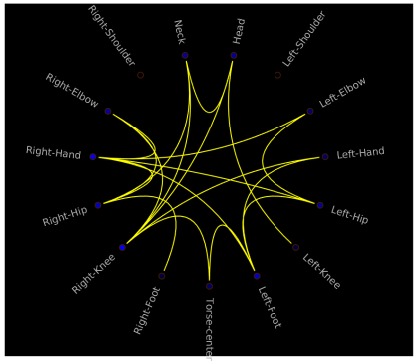












Comparison

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 GCN model	**98.46**

Outline

Conclusion

- We introduce in this paper a novel method which learns connectivity that "optimally" defines the support of aggregations and convolutions in GCNs.
- We investigate different settings which allow extracting non-differential and differential features as well as their combination before applying convolutions.
- We also consider different constraints (including orthogonality and stochasticity) which act as regularizers on the learned matrix operators.
- Experiments conducted on the challenging task of skeleton-based action recognition show the clear gain of the proposed method w.r.t different baselines as well as the related work.