Kernel Graph Convolutional Neural Networks

G. Nikolentzos[1], P. Meladianos[2], A. Tixier[1], K. Skianis[1], M. Vazirgiannis[1,2]

[1]École Polytechnique, France

[2]Athens University of Economics and Business, Greece

I. Introduction

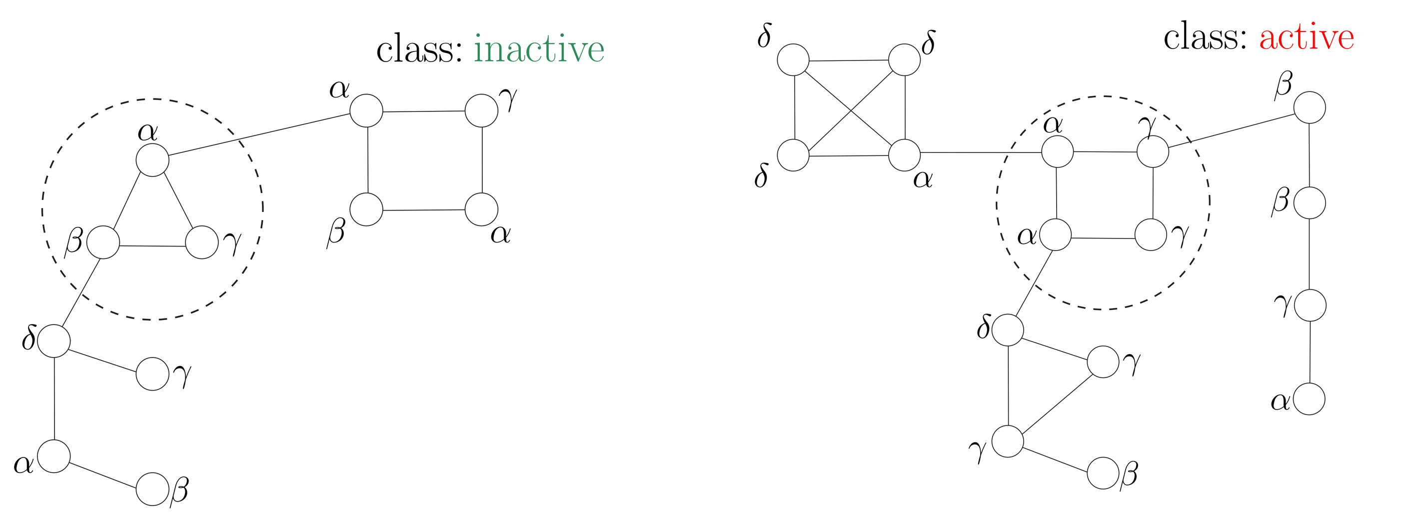
Goal: Perform graph classification

Motivation:

1. Graph classification is a very important task with numerous significant real-world applications (e.g. chemoinformatics, bioinformatics)
2. Existing algorithms generate features by considering the whole graph structure
   - However, significant subgraph patterns often confined only to small neighborhoods within the graph
   - Example: in the interaction networks of complex diseases, only specific subgraphs associated with the disease
   - Processing the entire graph may cause noise to be introduced into the generated features

Contributions:

- A graph classification approach that can identify regions in the graphs that are most predictive of the class labels
- It combines the learning potential of CNNs with the flexibility of graph kernels



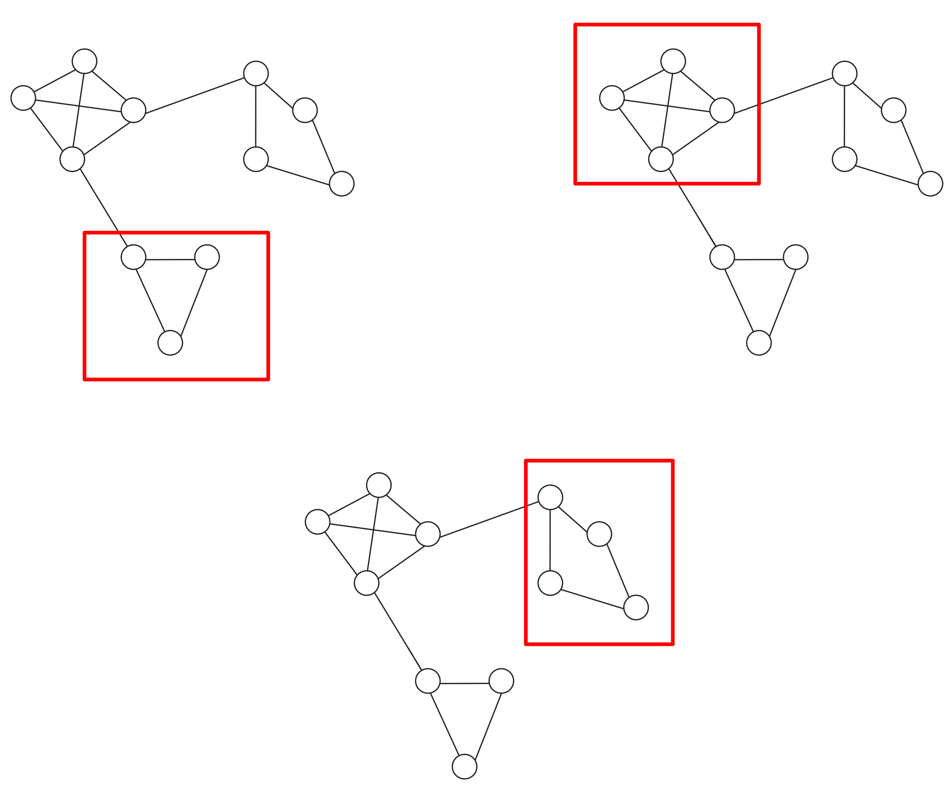
Two graphs and their corresponding subgraphs that determine class membership. Existing algorithms generate features by considering the whole graph structure which may cause noise to be introduced into the generated representations.

II. CNNs: From Images to Graphs

CNNs: Can identify indicative local predictors in a large structure, and combine them to produce a fixed size vector representation of the structure

Idea: Use CNNs to identify subgraphs that constitute strong clues regarding class membership

Example:
Figure illustrates a graph and a convolution filter that slides over all subgraphs



Problem: Standard filters cannot be used to filter graph data. How to perform convolutions on graphs?
$\hookrightarrow$ Use graphs as filters and *graph kernels* as activation functions

Graph kernels:

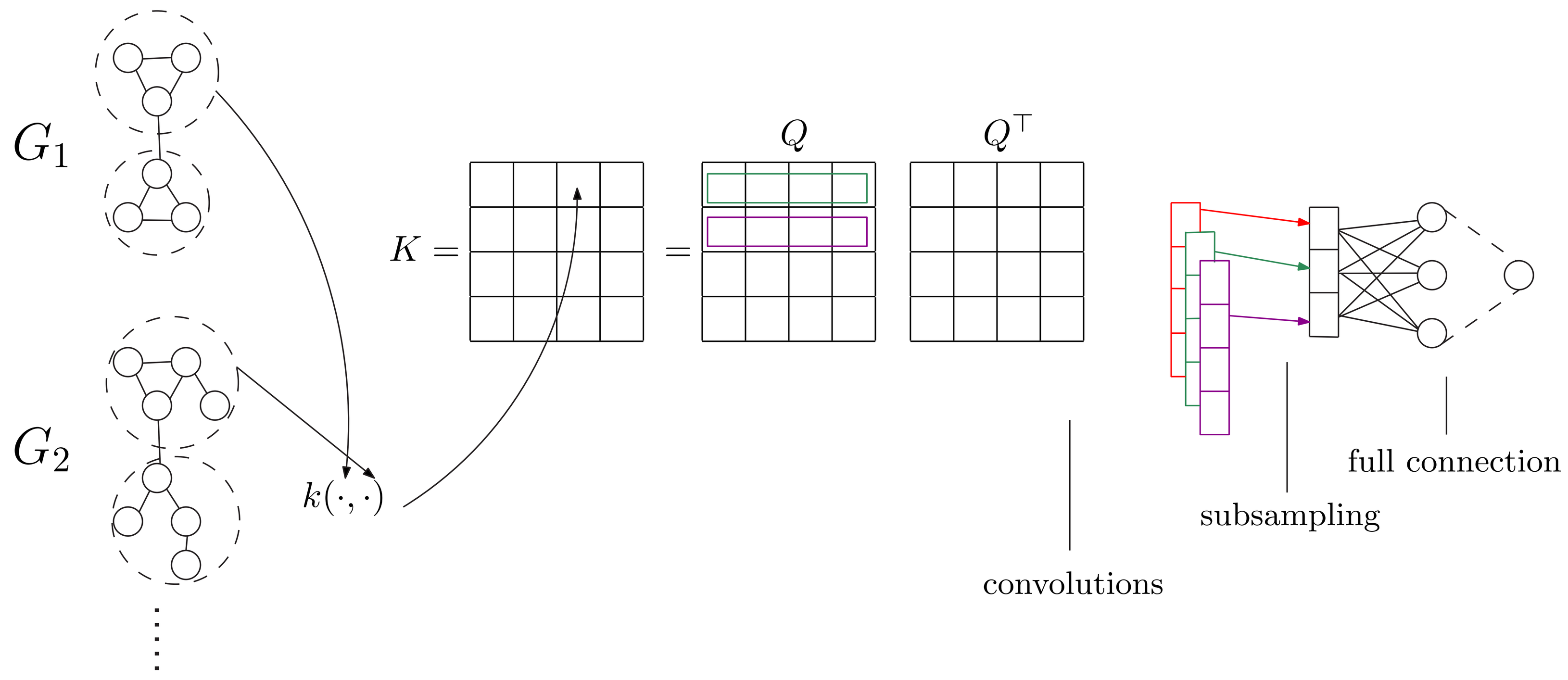
- Symmetric positive semidefinite functions on the set of graphs $\mathcal{G}$
- For any graph kernel $k : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}$
   - There exists a map $\phi : \mathcal{G} \rightarrow \mathcal{H}$ into a Hilbert space $\mathcal{H}$
   - It holds that $k(G, G') = \langle \phi(G), \phi(G') \rangle_{\mathcal{H}}$ for all $G, G' \in \mathcal{G}$

Updating graph filters during backpropagation is challenging
$\hookrightarrow$ Use graph kernels to *normalize* subgraphs (i.e., transform them to vectors)

III. Kernel Graph Convolutional Neural Network (KCNN)

Main steps:

- Extract a set of subgraphs that will play the role of patches (i.e. using community detection algorithms)
- Use graph kernels to generate kernel matrix between subgraphs (or approximate it using Nystrom)
- Decompose kernel matrix to get subgraph representation
- These representations are convolved with the filters of a 1-d CNN
- A pooling layer is followed by a fully-connected one to output class probabilities



Models:

Two single channel models:

- KCNN SP employs the shortest path kernel **[Borgwardt and Kriegel, ICDM '05]**
- KCNN WL employs the Weisfeiler-Lehman subtree kernel **[Shervashidze et al., JMLR '09]**

A model with two channels:

- KCNN SP+WL employs both kernels as different channels

IV. Graph Classification Results

Real-world Datasets

Method \ Dataset	ENZYMES	NCI1	PROTEINS	PTC-MR	D&D
SP	40.10 (± 1.50)	73.00 (± 0.51)	75.07 (± 0.54)	58.24 (± 2.44)	> 3 days
GR	26.61 (± 0.99)	62.28 (± 0.29)	71.67 (± 0.55)	57.26 (± 1.41)	78.45 (± 0.26)
RW	24.16 (± 1.64)	> 3 days	74.22 (± 0.42)	57.85 (± 1.30)	> 3 days
WL	53.15 (± 1.14)	80.13 (± 0.50)	72.92 (± 0.56)	56.97 (± 2.01)	77.95 (± 0.70)
Deep Kernels	**53.43** (± 0.91)	**80.31** (± 0.46)	75.68 (± 0.54)	60.08 (± 2.55)	NA
PSCN $k = 10$	NA	76.34 (± 1.68)	75.00 (± 2.51)	62.29 (± 5.68)	76.27 (± 2.64)
KCNN SP	46.35 (± 0.39)	75.70 (± 0.31)	74.27 (± 0.22)	<u>**62.94**</u> (± 1.69)	76.63 (± 0.09)
KCNN WL	43.08 (± 0.68)	75.83 (± 0.25)	<u>**75.76**</u> (± 0.28)	61.52 (± 1.41)	75.80 (± 0.07)
KCNN SP + WL	<u>48.12</u> (± 0.23)	<u>77.21</u> (± 0.22)	73.79 (± 0.29)	62.05 (± 1.41)	<u>**78.83**</u> (± 0.29)

Method \ Dataset	IMDB BINARY	IMDB MULTI	REDDIT BINARY	REDDIT MULTI-5K	COLLAB
GR	65.87 (± 0.98)	43.89 (± 0.38)	77.34 (± 0.18)	41.01 (± 0.17)	72.84 (± 0.28)
Deep GR	66.96 (± 0.56)	44.55 (± 0.52)	78.04 (± 0.39)	41.27 (± 0.18)	73.09 (± 0.25)
PSCN $k = 10$	71.00 (± 2.29)	45.23 (± 2.84)	**86.30** (± 1.58)	49.10 (± 0.70)	72.60 (± 2.15)
KCNN SP	69.60 (± 0.44)	45.99 (± 0.23)	77.23 (± 0.15)	44.86 (± 0.24)	70.78 (± 0.12)
KCNN WL	70.46 (± 0.45)	46.44 (± 0.24)	<u>81.85</u> (± 0.12)	<u>**50.04**</u> (± 0.19)	<u>**74.93**</u> (± 0.14)
KCNN SP + WL	<u>**71.45**</u> (± 0.15)	<u>**47.46**</u> (± 0.21)	78.35 (± 0.11)	44.63 (± 0.18)	74.12 (± 0.17)

10-fold cross validation average classification accuracy (± standard deviation) of the proposed models and the baselines. Best performance per dataset in **bold**, among the variants of Kernel CNN <u>underlined</u>.

- On 7/10 datasets, the proposed models outperformed the baselines
- The multi-channel architecture (KCNN SP + WL) led to better results on 5/10 datasets

Synthetic Dataset

Constructed to empirically verify that KCNN can identify the significant subgraph patterns inside a graph:

- Step 1: generate an Erdos-Rényi graph
   - number of vertices sampled from $\{100, 101, \ldots, 200\}$
   - edge probability 0.1
- Step 2: generate randomly either a 10-clique (class -1) or a 10-star graph (class 1)
- Step 3: connect pairs of vertices of the two graphs with probability 0.1

Method \ Dataset	Synthetic
SP	75.47
GR	69.34
WL	65.88
KCNN SP	98.20
KCNN WL	97.25
KCNN SP+WL	**98.40**

10-fold cross validation average classification accuracy of the proposed models and the baselines on the synthetic dataset.

- All three variants achieved accuracies greater than 97%
- Conversely, graph kernels failed to discriminate between the two categories

Implementation available at: `https://github.com/giannisnik/cnn-graph-classification`

 nikolentzos@lix.polytechnique.fr