Machine learning for graphs and with graphs

Graph kernels

Titouan Vayer & Pierre Borgnat
email: titouan.vayer@inria.fr, pierre.borgnat@ens-lyon.fr

September 23, 2024

ENS DE LYON

Table of contents

Kernels for structured data

Objective

Given a dataset of graphs $(G_1, \cdots, G_n)$ can we build machine learning models to do:

- Supervised learning: each graph associated to $y_i \in \mathcal{Y}$.
- Unsupervised learning: PCA, Kernel PCA, graph embedding...

Kernels for structured data

Objective

Given a dataset of graphs $(G_1, \cdots, G_n)$ can we build machine learning models to do:

- Supervised learning: each graph associated to $y_i \in \mathcal{Y}$.
- Unsupervised learning: PCA, Kernel PCA, graph embedding...

Application of RKHS for graphs

Let $\mathcal{X} = \{$ set of all graphs $\}$ can we build interesting kernels $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$?

- For $G, G' \in \mathcal{X}, \kappa(G, G')$ is a notion of "similarity" between graphs.
- Gram matrix $\mathbf{K} = (\kappa(G_i, G_j))_{(i,j) \in [\![n]\!]^2}$.
- Then do stuff...

Kernels for structured data

Objective

Given a dataset of graphs $(G_1, \cdots, G_n)$ can we build machine learning models to do:

- Supervised learning: each graph associated to $y_i \in \mathcal{Y}$.
- Unsupervised learning: PCA, Kernel PCA, graph embedding...

Application of RKHS for graphs

Let $\mathcal{X} = \{$ set of all graphs $\}$ can we build interesting kernels $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$?

- For $G, G' \in \mathcal{X}, \kappa(G, G')$ is a notion of "similarity" between graphs.
- Gram matrix $\mathbf{K} = (\kappa(G_i, G_j))_{(i,j) \in [\![n]\!]^2}$.
- Then do stuff...

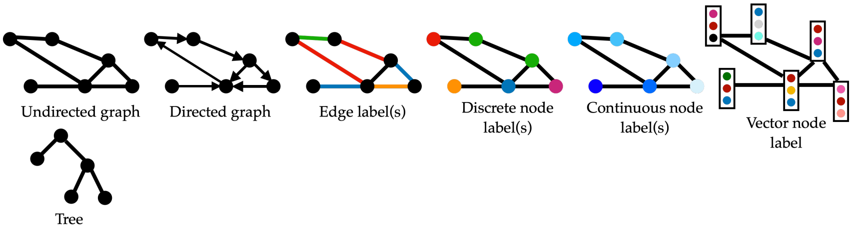
Some notations

A graph $G = (V, E)$. Labeling function if attributes/labels $\ell_G : V \cup E \to S$ (S discrete or continuous $\subset \mathbb{R}^N$)

What is a good graph kernel ?

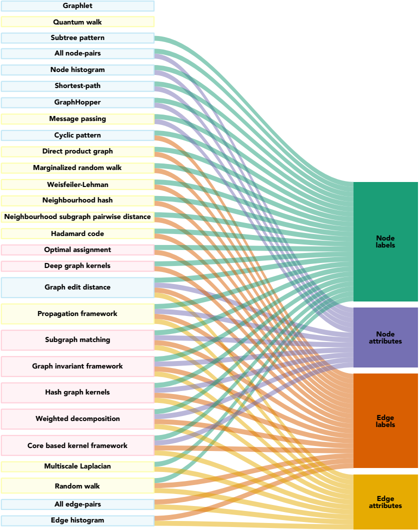
Properties of the graph kernel

- Handle graphs that are directed (or undirected) ?
- Handle node or edge labels or attributes that are present in the graphs?
- Efficient to compute ? Complexity *w.r.t.* $|V|, |E|, \dim$?
- Is there a particular relevant substructure (*e.g.* tree patterns) that would preclude the choice of a particular kernel?



The kernel jungle

Surveys: K. Borgwardt et al. 2020; Nikolentzos, Siglidis, and Vazirgiannis 2021



Graph Kernel	Exp. ϕ	Node Labels	Node Attributes	Type	Complexity
Vertex Histogram	✓	✓	✗	R-convolution	$\mathcal{O}(n)$
Edge Histogram	✓	✓	✗	R-convolution	$\mathcal{O}(m)$
Random Walk	✗†	✓	✓	R-convolution	$\mathcal{O}(n^3)$
Subtree	✗	✓	✓	R-convolution	$\mathcal{O}(n^2 4^{deg^*} h)$
Cyclic Pattern	✓	✓	✗	intersection	$\mathcal{O}((c+2)n+2m)$
Shortest Path	✗†	✓	✓	R-convolution	$\mathcal{O}(n^4)$
Graphlet	✓	✗	✗	R-convolution	$\mathcal{O}(n^k)$
Weisfeiler-Lehman Subtree	✓	✓	✗	R-convolution	$\mathcal{O}(hm)$
Neighborhood Hash	✓	✓	✗	intersection	$\mathcal{O}(hm)$
Neighborhood Subgraph Pairwise Distance	✓	✓	✗	R-convolution	$\mathcal{O}(n^2 m \log(m))$
Lovász ϑ	✓	✗	✗	R-convolution	$\mathcal{O}(n(s+\frac{nm}{\epsilon})+s^2)$
SVM-ϑ	✓	✗	✗	R-convolution	$\mathcal{O}(n(s+n^2)+s^2)$
Ordered Decomposition DAGs	✓	✓	✗	R-convolution	$\mathcal{O}(n \log n)$
Pyramid Match	✗	✓	✗	assignment	$\mathcal{O}(ndL)$
Weisfeiler-Lehman Optimal Assignment	✗	✓	✗	assignment	$\mathcal{O}(hm)$
Subgraph Matching	✗	✓	✓	R-convolution	$\mathcal{O}(kn^{k+1})$
GraphHopper	✗	✓	✓	R-convolution	$\mathcal{O}(n^4)$
Graph Invariant Kernels	✗	✓	✓	R-convolution	$\mathcal{O}(n^6)$
Propagation	✓	✓	✓	R-convolution	$\mathcal{O}(hm)$
Multiscale Laplacian	✗	✓	✓	R-convolution	$\mathcal{O}(n^5 h)$

Bag of structures

A majority of graph kernels are instances of the *convolution kernels* Haussler et al. 1999.

Principle

- Compare graphs by first dividing them into substructures of various granularity.
- E.g. vertices, subgraphs, all shortest paths of a graph.
- Defining *base kernels* at the fine granularity and combine them.
- Of the form $\kappa(G, G') = \sum_{r\in\mathcal{R}, r'\in\mathcal{R}'} \kappa_{\text{substructure}}(r, r')$.

Bag of structures

A majority of graph kernels are instances of the *convolution kernels* Haussler et al. 1999.

Principle

- Compare graphs by first dividing them into substructures of various granularity.
- E.g. vertices, subgraphs, all shortest paths of a graph.
- Defining *base kernels* at the fine granularity and combine them.
- Of the form $\kappa(G, G') = \sum_{r \in \mathcal{R}, r' \in \mathcal{R}'} \kappa_{\text{substructure}}(r, r')$.

Advantages & limitations

- Intuitive definitions + relatively good results.
- Sometimes computational limitations.
- Expressiveness limitations.
- "Diagonal dominance problem" Yanardag and Vishwanathan 2015.

All node-pairs kernel

A first idea

- Given $G = (V, E), G' = (V', E')$,
- Suppose the labels of the nodes of both graphs are in S.
- Consider a kernel on the nodes

$$\kappa_{\text{node}} : S \times S \to \mathbb{R}$$

- The all node-pairs kernel is defined by

$$\kappa(G, G') = \sum_{v \in V} \sum_{v' \in V'} \kappa_{\text{node}}(\ell_G(v), \ell_{G'}(v'))$$

All node-pairs kernel

A first idea

- Given $G = (V, E)$, $G' = (V', E')$,
- Suppose the labels of the nodes of both graphs are in S.
- Consider a kernel on the nodes

$$\kappa_{\text{node}} : S \times S \to \mathbb{R}$$

- The all node-pairs kernel is defined by

$$\kappa(G, G') = \sum_{v \in V} \sum_{v' \in V'} \kappa_{\text{node}}(\ell_G(v), \ell_{G'}(v'))$$

Remarks

- Runtime in $O(|V| \times |V'| \times \dim(S))$.
- Can handle discrete/continuous labels.
- Does not take into account the structures of the graphs.

Node histogram kernel

A baseline kernel (1/2)

- Suppose the labels are discrete over a finite alphabet

$$\Sigma = \{\sigma_1, \cdots, \sigma_{|\Sigma|}\}$$

- The node histogram kernel is defined as

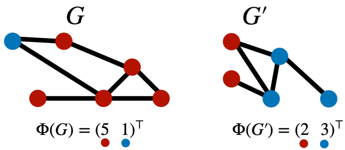
$$\kappa_{\mathrm{NH}}(G, G') = \langle \Phi(G), \Phi(G') \rangle .$$

where

$$\Phi(G) = \Big(\sum_{v \in V} \mathbf{1}_{\ell_G(v)=\sigma_1}, \cdots, \sum_{v \in V} \mathbf{1}_{\ell_G(v)=\sigma_{|\Sigma|}}\Big) .$$

- Simply corresponds to an unnormalised histogram that counts the occurrence of each node label in the graph.

Node histogram kernel



Node histogram kernel

A baseline kernel (1/2)

- Suppose the labels are discrete over a finite alphabet

$$\Sigma = \{\sigma_1, \cdots, \sigma_{|\Sigma|}\}$$

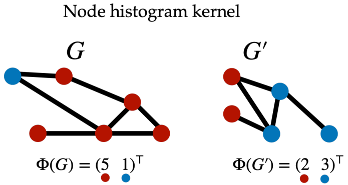
- The node histogram kernel is defined as

$$\kappa_{\mathrm{NH}}(G, G') = \langle \Phi(G), \Phi(G') \rangle .$$

where

$$\Phi(G) = \Big(\sum_{v \in V} \mathbf{1}_{\ell_G(v)=\sigma_1}, \cdots, \sum_{v \in V} \mathbf{1}_{\ell_G(v)=\sigma_{|\Sigma|}}\Big) .$$

- Simply corresponds to an unnormalised histogram that counts the occurrence of each node label in the graph.



Remarks

- Can be computed in $O(|V| + |V|')$.
- Does not take into account the structures of the graphs.

Node histogram kernel

A baseline kernel (1/2)

- Suppose the labels are discrete over a finite alphabet

$$\Sigma = \{\sigma_1, \cdots, \sigma_{|\Sigma|}\}$$

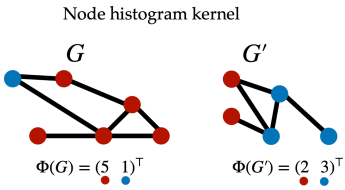
- The node histogram kernel is defined as

$$\kappa_{\text{NH}}(G, G') = \langle \Phi(G), \Phi(G') \rangle .$$

where

$$\Phi(G) = \Big(\sum_{v \in V} \mathbf{1}_{\ell_G(v) = \sigma_1}, \cdots, \sum_{v \in V} \mathbf{1}_{\ell_G(v) = \sigma_{|\Sigma|}} \Big) .$$

- Simply corresponds to an unnormalised histogram that counts the occurrence of each node label in the graph.



Remarks

- Can be computed in $O(|V| + |V|')$.
- Does not take into account the structures of the graphs.
- Of the form $\kappa_{\text{NH}}(G, G') = \sum_{v \in V, v' \in V'} \mathbf{1}_{\ell_G(v) = \ell_{G'}(v')}$.

Edge histogram kernel

A baseline kernel (2/2)

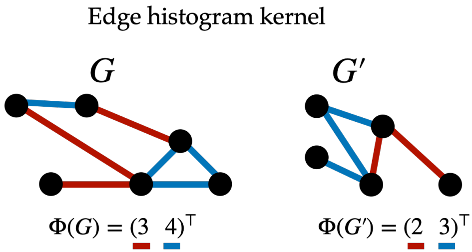
- Suppose the **edges labels** are discrete over a finite alphabet

$$\Sigma = \{\sigma_1, \cdots, \sigma_{|\Sigma|}\}$$

- The edge histogram kernel is defined as

$$\kappa_{\mathrm{EH}}(G, G') = \langle \Phi(G), \Phi(G') \rangle \,.$$

where $\Phi(G) = \left(\sum_{e \in E} \mathbf{1}_{\ell(e)=\sigma_1}, \cdots, \sum_{e \in E} \mathbf{1}_{\ell(e)=\sigma_{|\Sigma|}}\right)$.



Edge histogram kernel

A baseline kernel (2/2)

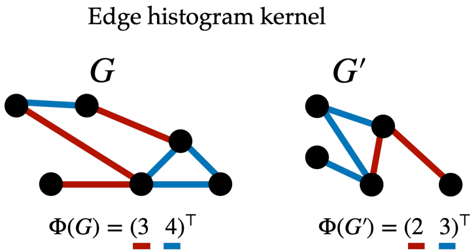
- Suppose the **edges labels** are discrete over a finite alphabet

$$\Sigma = \{\sigma_1, \cdots, \sigma_{|\Sigma|}\}$$

- The edge histogram kernel is defined as

$$\kappa_{\mathrm{EH}}(G, G') = \langle \Phi(G), \Phi(G') \rangle .$$

where $\Phi(G) = \left(\sum_{e \in E} \mathbf{1}_{\ell(e)=\sigma_1}, \cdots, \sum_{e \in E} \mathbf{1}_{\ell(e)=\sigma_{|\Sigma|}}\right)$.



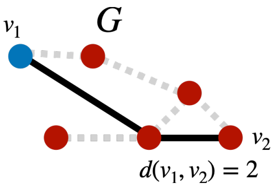
Remarks

- Can be computed in $O(|E| + |E|')$.
- Does not take into account the labels of the nodes.
- Can be combined with the previous one as

$$\kappa(G, G') = \kappa_{\mathrm{EH}}(G, G') \times \kappa_{\mathrm{NH}}(G, G')$$

The shortest-path kernel

K. M. Borgwardt and Kriegel 2005



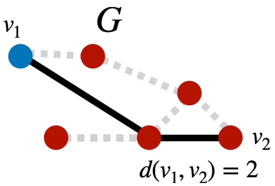
- Compute all pair-to-pair shortest-paths in G, G' with Floyd-Warshall.
- The kernel is defined as

$$\kappa_{SP}(G, G') = \sum_{(v_1, v_2) \in V} \sum_{(v_1', v_2') \in V'} \kappa_0(d(v_1, v_2), d(v_1', v_2')) \,.$$

  where $d(v_1, v_2)$ is the shortest-path distance between v_1, v_2.
- κ_0 is a kernel that compares the lengths of the two shortest-paths.
- $\kappa_0(x, y) = x \times y$ (linear kernel) or $\kappa_0(x, y) = \mathbf{1}_{x=y}$ (dirac).

The shortest-path kernel

K. M. Borgwardt and Kriegel 2005



- Compute all pair-to-pair shortest-paths in G, G' with Floyd-Warshall.
- The kernel is defined as

$$\kappa_{SP}(G, G') = \sum_{(v_1,v_2)\in V} \sum_{(v_1',v_2')\in V'} \kappa_0(d(v_1, v_2), d(v_1', v_2')) .$$

  where $d(v_1, v_2)$ is the shortest-path distance between v_1, v_2.
- κ_0 is a kernel that compares the lengths of the two shortest-paths.
- $\kappa_0(x, y) = x \times y$ (linear kernel) or $\kappa_0(x, y) = \mathbf{1}_{x=y}$ (dirac).

Remarks

- Complexity Floyd-Warshall on G, $O(|V|^3)$.
- Variants with Bellman–Ford's, Dijkstra's algorithms.
- General complexity for κ_{SP} $O(|V|^2|V'|^2)$.
- Many variants **with attributes**.

GraphHopper kernel

Undirected graphs with edge weights and node attributes.

- Even for real-valued/vector attributes Feragen et al. 2013.

- Kernel is defined as

$$\kappa_{\mathrm{GH}}(G, G') = \sum_{p \in \mathcal{P}_G} \sum_{p' \in \mathcal{P}_{G'}} \kappa_0(p, p')$$ where $\mathcal{P}_G$: set of **all shortest-paths**.

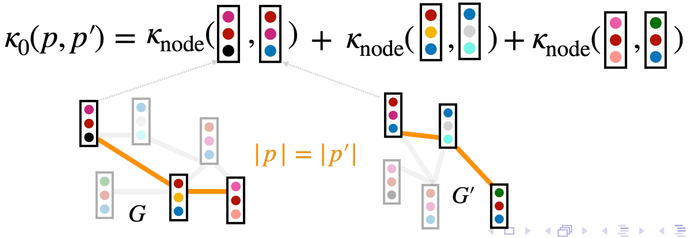
GraphHopper kernel

Undirected graphs with edge weights and node attributes.

- Even for real-valued/vector attributes Feragen et al. 2013.
- Kernel is defined as

$$\kappa_{\text{GH}}(G, G') = \sum_{p \in \mathcal{P}_G} \sum_{p' \in \mathcal{P}_{G'}} \kappa_0(p, p') \text{ where } \mathcal{P}_G\text{: set of } \textbf{all shortest-paths}.$$

- Base kernel $\kappa_0(p, p') = \begin{cases} \sum_{j=1}^{|p|} \kappa_{\text{node}}(p_j, p'_j) & \text{if equal length} |p| = |p'| \\ 0 & \text{otherwise} \end{cases}$



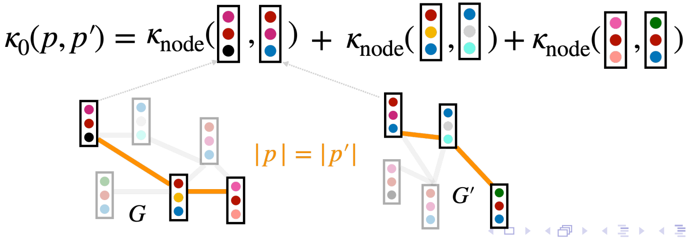
GraphHopper kernel

Undirected graphs with edge weights and node attributes.

- Even for real-valued/vector attributes Feragen et al. 2013.
- Interestingly averaged overall worst-case complexity $O(|V||V'|\dim(S))$.
- Kernel is defined as

$$\kappa_{\mathrm{GH}}(G, G') = \sum_{p \in \mathcal{P}_G} \sum_{p' \in \mathcal{P}_{G'}} \kappa_0(p, p') \text{ where } \mathcal{P}_G\text{: set of } \textbf{all shortest-paths}.$$

- Base kernel $\kappa_0(p, p') = \begin{cases} \sum_{j=1}^{|p|} \kappa_{\mathrm{node}}(p_j, p'_j) & \text{if equal length} |p| = |p'| \\ 0 & \text{otherwise} \end{cases}$



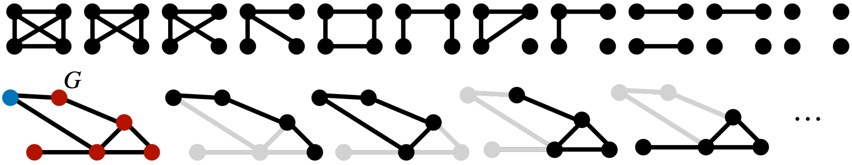
The Graphlet kernel

Principle Shervashidze, Vishwanathan, et al. 2009

- Count substructures in graphs.
- Graphlet = subgraph with k vertices.
- $\mathbb{G} := \{\mathfrak{g}_1, \cdots, \mathfrak{g}_{N_k}\}$ set of k-graphlets (asymptotically $N_k \approx 2^{\binom{k}{2}}/k!$).
- Kernel $\kappa(G, G') = \langle \Phi(G), \Phi(G') \rangle$

$$\Phi(G) \propto (|\{\mathfrak{g}_i \in G\}|, \cdots, |\{\mathfrak{g}_{N_k} \in G\}|)^\top$$

All subgraphs of size 4



Different size 4 graphlets found in G

The Graphlet kernel

Principle Shervashidze, Vishwanathan, et al. 2009

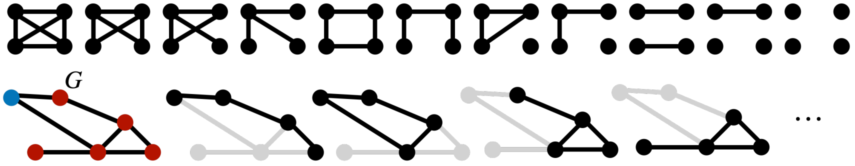
- Count substructures in graphs.
- Graphlet = subgraph with k vertices.
- $\mathbb{G} := \{\mathfrak{g}_1, \cdots, \mathfrak{g}_{N_k}\}$ set of k-graphlets (asymptotically $N_k \approx 2^{\binom{k}{2}}/k!$).
- Kernel $\kappa(G, G') = \langle \Phi(G), \Phi(G') \rangle$

$$\Phi(G) \propto (|\{\mathfrak{g}_i \in G\}|, \cdots, |\{\mathfrak{g}_{N_k} \in G\}|)^\top$$

Remarks

- Ignores all labels.
- Computational bottleneck: enumeration of all graphlets.
- Complexity in $O(|V|^k)$ time.
- Typically $k \in \{3, 4, 5\}$.
- Counting all possible subgraphs is NP-hard Gärtner, Flach, and Wrobel 2003.

All subgraphs of size 4



Different size 4 graphlets found in G

The graph isomorphism problem

Checking if two graphs are "identical"

Two graphs $G = (V, E), G' = (V', E')$ are **isomorphic** ($G \cong G'$) if there exists a **bijection** $\Psi : V \to V'$ such that

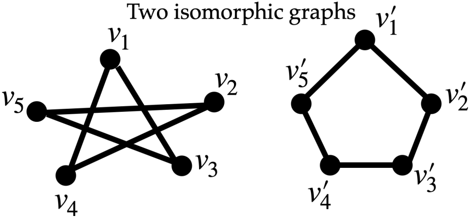
$$(u, v) \in E \iff (\Psi(u), \Psi(v)) \in E' \,.$$

The graph isomorphism problem

Checking if two graphs are "identical"

Two graphs $G = (V, E), G' = (V', E')$ are **isomorphic** ($G \cong G'$) if there exists a **bijection** $\Psi : V \to V'$ such that

$$(u, v) \in E \iff (\Psi(u), \Psi(v)) \in E' \, .$$



Remark

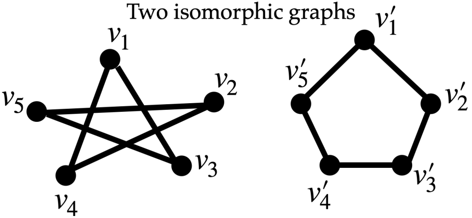
- Same graphs up to a permutation.
- Currently no known polynomial-time algorithms for solving this problem.
- Not known to be NP-complete.
- Quasi-polynomial algorithm Babai 2016.

The graph isomorphism problem

Checking if two graphs are "identical"

Two graphs $G = (V, E), G' = (V', E')$ are **isomorphic** ($G \cong G'$) if there exists a **bijection** $\Psi : V \to V'$ such that

$$(u, v) \in E \iff (\Psi(u), \Psi(v)) \in E' .$$



Remark

- Same graphs up to a permutation.
- Currently no known polynomial-time algorithms for solving this problem.
- Not known to be NP-complete.
- Quasi-polynomial algorithm Babai 2016.

Weisfeiler-Lehman test of isomorphism Leman and Weisfeiler 1968

On the board

Table of contents

Multi-set vs set

Key differences

Without being too formal.

- A set $X = \{a, b\}$ is equal to $Y = \{b, a\}$ because $x \in X \iff x \in Y$: **order is irrelevant**.
- A set $Z = \{a, a, b\}$ is also equal to X: **the same element can appear more than once**.
- A **multi-set** denoted with $\{\{\cdots\}\}$ is a "set" where elements can appear more that once.
- The order is still irrelevant.
- For example $\{\{a, a, b\}\}$.
- Formal definition: a multiset is a couple (X, m) where X is a set and a $m : X \to \mathbb{N}$ counts the multiplicity of each element.

Weisfeiler–Lehman kernel

A very popular graph kernel based on Shervashidze, Schweitzer, et al. 2011

- Originally handle graphs with discrete labels.
- Uses **iterative label refinement**.
- Concepts from the Weisfeiler-Lehman test of isomorphism.

Weisfeiler–Lehman kernel

A very popular graph kernel based on Shervashidze, Schweitzer, et al. 2011

- Originally handle graphs with discrete labels.
- Uses **iterative label refinement**.
- Concepts from the Weisfeiler-Lehman test of isomorphism.

Graphs relabeling/refinement

- Recursively refine the node labels by applying local transformations

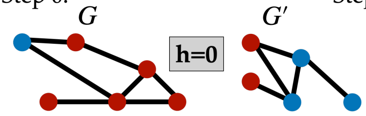
$$a_v = \text{AGGREGATE}\left(\{\{\ell_G^{(\text{old})}(v'); v' \in \mathcal{N}(v)\}\}\right)$$

$$\text{and } \ell_G^{(\text{new})}(v) = \text{COMBINE}\left(\ell_G^{(\text{old})}(v), a_v\right) .$$

- This general idea can give rise to a multitude of distinct graph kernels:
- (i) the specific form of COMBINE, AGGREGATE.
- (ii) which kernels are used to compare the resulting modified graphs.
- (iii) how the graph at multiple scales are aggregated into a single value.

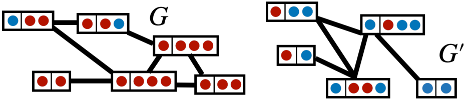
Weisfeiler–Lehman kernel

Step 0:



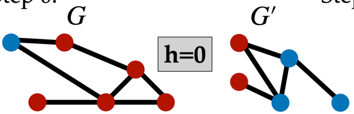
$\Phi(G) = (5 \quad 1)^\top$ $\Phi(G') = (2 \quad 3)^\top$

Step 1: « Enrich » the labels with neighbors



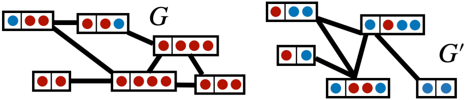
Weisfeiler–Lehman kernel

Step 0:

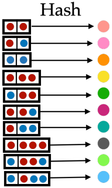


$\Phi(G) = (5 \;\; 1)^\top$ $\Phi(G') = (2 \;\; 3)^\top$

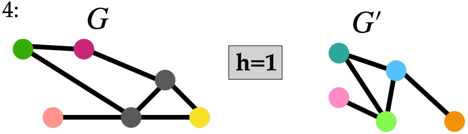
Step 1: « Enrich » the labels with neighbors



Step 3:



Step 4:



$\Phi(G) = (5 \;\; 1 \,\vdots\, 1 \;\; 0 \;\; 0 \;\; 1 \;\; 1 \;\; 1 \;\; 0 \;\; 2 \;\; 0 \;\; 0)^\top$ $\Phi(G') = (2 \;\; 3 \,\vdots\, 0 \;\; 1 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 1 \;\; 1)^\top$

-> Step 0 …

-> Terminates after h iterations

Weisfeiler–Lehman kernel

The Weisfeiler–Lehman kernel

- The function AGGREGATE sorts in alphabetic order.
- The function COMBINE hashes to compress the tuple into a single integer-valued label.
- Produces a sequence of graphs $(G_0, \cdots, G_h)$.
- The Weisfeiler–Lehman kernel is

$$\kappa_{\mathrm{WL}}(G, G') = \sum_{i=0}^{h} \kappa_0(G_i, G_i'),$$

for a base kernel κ_0.

Weisfeiler–Lehman kernel

The Weisfeiler–Lehman kernel

- The function AGGREGATE sorts in alphabetic order.
- The function COMBINE hashes to compress the tuple into a single integer-valued label.
- Produces a sequence of graphs $(G_0, \cdots, G_h)$.
- The Weisfeiler–Lehman kernel is

$$\kappa_{\mathrm{WL}}(G, G') = \sum_{i=0}^{h} \kappa_0(G_i, G_i'),$$

  for a base kernel κ_0.
- Most common κ_0 *subtree kernel*: $\Phi(G) =$ number of occurrences of each label in the alphabet of all compressed labels at each step.
- Complexity: for one graph $O(|E| \times h)$.
- Runtime scales only linearly with the number of edges !

Optimal assignment kernel

General setting (Kriege, Giscard, and Wilson 2016)

- Different than "bag of structure" kernels.
- Let $X, Y \subset \Omega$ with $|X| = |Y|$.

$$\kappa_{OA}(X, Y) = \max_{B \in \mathcal{B}(X,Y)} \sum_{x \in X} \kappa_0(x, B(y)) \text{ where } \mathcal{B}(X, Y) = \text{all bijections.}$$

- κ is a valid PSD kernel if $\kappa_0 : \Omega \times \Omega \to \mathbb{R}_+$ is *strong*:

$$\kappa_0(x, y) \geq \min\{\kappa_0(x, z), \kappa_0(z, y)\} \; \forall (x, y, z).$$

- Assign the parts of one objects to the parts of the other *s.t.* the total similarity is maximum possible.

Optimal assignment kernel

General setting (Kriege, Giscard, and Wilson 2016)

- Different than "bag of structure" kernels.
- Let $X, Y \subset \Omega$ with $|X| = |Y|$.

$$\kappa_{OA}(X,Y) = \max_{B \in \mathcal{B}(X,Y)} \sum_{x \in X} \kappa_0(x, B(y)) \text{ where } \mathcal{B}(X,Y) = \text{all bijections.}$$

- κ is a valid PSD kernel if $\kappa_0 : \Omega \times \Omega \to \mathbb{R}_+$ is *strong*:

$$\kappa_0(x,y) \geq \min\{\kappa_0(x,z), \kappa_0(z,y)\} \ \forall (x,y,z).$$

- Assign the parts of one objects to the parts of the other *s.t.* the total similarity is maximum possible.

Weisfeiler-Lehman optimal assignment kernel

- $i \in [\![h]\!], \tau_i(v)$ denotes the color of vertex v at step i of the WL process.
- The base kernel is $\kappa_0(v, v') = \sum_{i=0}^{h} \mathbf{1}_{\tau_i(v)=\tau_i(v')}$ + padding.
- Can also be computed in $O(hm)$.

Continuous alternative to Weisfeiler–Lehman

Hash graph kernel Morris et al. 2016

- Let κ be a graph kernel (such as WL).
- $\mathfrak{H} = \{\mathfrak{h}_1, \mathfrak{h}_2 \cdots\}$ a family of hash functions.
- $\mathfrak{h}_i : \mathbb{R}^d \to \mathbb{N}$ is a hash function.
- $\mathfrak{h}_i(G)$: the discretised graph resulting from applying $\mathfrak{h}_i$ to continuous attributes of the graph.
- The kernel is defined as

$$\kappa_{\text{HGK}}(G, G') = \frac{1}{|\mathfrak{H}|} \sum_{i \in \mathfrak{H}} \kappa(\mathfrak{h}_i(G), \mathfrak{h}_i(G'))\,.$$

Continuous alternative to Weisfeiler–Lehman

Hash graph kernel Morris et al. 2016

- Let κ be a graph kernel (such as WL).
- $\mathfrak{H} = \{\mathfrak{h}_1, \mathfrak{h}_2 \cdots\}$ a family of hash functions.
- $\mathfrak{h}_i : \mathbb{R}^d \to \mathbb{N}$ is a hash function.
- $\mathfrak{h}_i(G)$: the discretised graph resulting from applying $\mathfrak{h}_i$ to continuous attributes of the graph.
- The kernel is defined as

$$\kappa_{\mathrm{HGK}}(G, G') = \frac{1}{|\mathfrak{H}|} \sum_{i \in \mathfrak{H}} \kappa(\mathfrak{h}_i(G), \mathfrak{h}_i(G')) .$$

Example of hash functions

- Locality-sensitive hashing schemes Datar et al. 2004.
- Idea: if $\mathbf{x}, \mathbf{y}$ are "close" then $\mathbb{P}[\mathfrak{h}_1(\mathbf{x}) = \mathfrak{h}_2(\mathbf{y})]$ is "high" and conversely.
- More collusion for nearby points.
- e.g. $\mathfrak{h}(\mathbf{x}) = \lfloor \frac{\langle \mathbf{x}, \mathbf{a} \rangle + b}{r} \rfloor, \mathbf{a} \sim \mu, b \sim \mathrm{unif}([0, r])$

Table of contents

Conclusion

- Graph kernels are very simple but powerful way of using all the ML machinery on graphs.
- The big question is to choose the "right" kernel.
- No straight answer, it depends on the task.
- In practice: always use simple graph kernels as baselines.

References I

Aronszajn, Nachman (1950). "Theory of reproducing kernels". In: *Transactions of the American mathematical society* 68.3, pp. 337–404.

Babai, László (2016). "Graph isomorphism in quasipolynomial time". In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 684–697.

Borgwardt, Karsten et al. (2020). "Graph kernels: State-of-the-art and future challenges". In: *Foundations and Trends® in Machine Learning* 13.5-6, pp. 531–712.

Borgwardt, Karsten M and Hans-Peter Kriegel (2005). "Shortest-path kernels on graphs". In: *Fifth IEEE international conference on data mining (ICDM'05)*. IEEE, 8–pp.

Datar, Mayur et al. (2004). "Locality-sensitive hashing scheme based on p-stable distributions". In: *Proceedings of the twentieth annual symposium on Computational geometry*, pp. 253–262.

Feragen, Aasa et al. (2013). "Scalable kernels for graphs with continuous attributes". In: *Advances in neural information processing systems* 26.

References II

Gärtner, Thomas, Peter Flach, and Stefan Wrobel (2003). "On graph kernels: Hardness results and efficient alternatives". In: *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings*. Springer, pp. 129–143.

Haussler, David et al. (1999). *Convolution kernels on discrete structures*. Tech. rep. Citeseer.

Kriege, Nils M, Pierre-Louis Giscard, and Richard Wilson (2016). "On valid optimal assignment kernels and applications to graph classification". In: *Advances in neural information processing systems* 29.

Leman, AA and Boris Weisfeiler (1968). "A reduction of a graph to a canonical form and an algebra arising during this reduction". In: *Nauchno-Technicheskaya Informatsiya* 2.9, pp. 12–16.

Morris, Christopher et al. (2016). "Faster kernels for graphs with continuous attributes via hashing". In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, pp. 1095–1100.

References III

Nikolentzos, Giannis, Giannis Siglidis, and Michalis Vazirgiannis (2021). "Graph kernels: A survey". In: *Journal of Artificial Intelligence Research* 72, pp. 943–1027.

Shervashidze, Nino, Pascal Schweitzer, et al. (2011). "Weisfeiler-lehman graph kernels.". In: *Journal of Machine Learning Research* 12.9.

Shervashidze, Nino, SVN Vishwanathan, et al. (2009). "Efficient graphlet kernels for large graph comparison". In: *Artificial intelligence and statistics*. PMLR, pp. 488–495.

Steinwart, Ingo and Andreas Christmann (2008). *Support vector machines*. Springer Science & Business Media.

Wendland, Holger (2004). *Scattered data approximation*. Vol. 17. Cambridge university press.

Yanardag, Pinar and SVN Vishwanathan (2015). "Deep graph kernels". In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374.