A graph pre-image method based on graph edit distances

Linlin Jia
linlin.jia@insa-rouen.fr

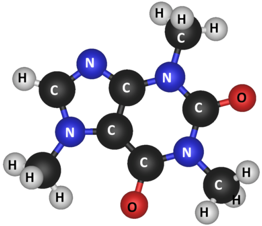
Supervisors:
Paul Honeine, paul.honeine@univ-rouen.fr
Benoit Gaüzère, benoit.gauzere@insa-rouen.fr

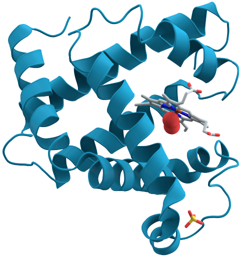
Normandie Université, INSA Rouen et Université de Rouen, LITIS Lab

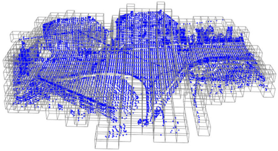
19/09/2020

Overview

Graph data



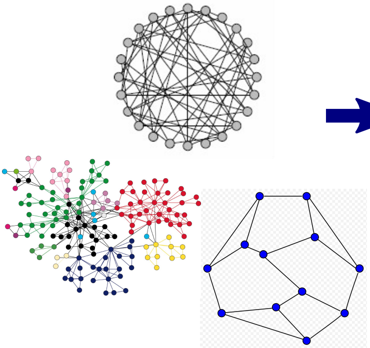


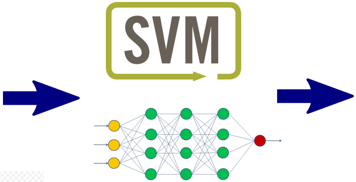


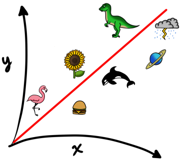
These images are from the Internet.

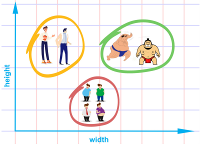
Graph structure and machine learning methods

- nodes, edges
- structural information



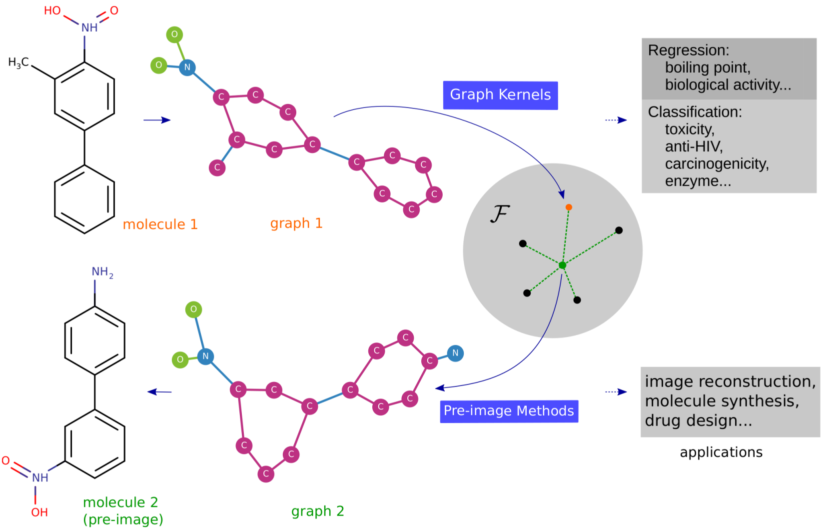






These images are from the Internet.

Graph kernels and pre-image problem



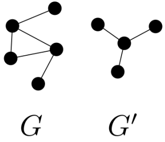
Current methods to generate graphs

- **Pre-image based on iterations:**
  - Explore the benefit of kernel space.
  - New graphs can only be generated by adding or removing edges, and only a restricted number of edges can be inserted or deleted, thus the possible graph space is not fully explored.
  - Randomly generate new graphs in each iteration, which may cause the decrease of the quality of the pre-image, and is time consuming.
  - No labelling information can be tackled.
- **Based on graph edit distance (GED):**
  - A nature way to construct graphs.
  - Only explore the graph space.

-> Our method: combine the two methods above, use GED as a direction when constructing graphs in the iteration of the pre-image method.

Overview

Linear and non-linear graph kernels



Kernel

$$k(G,G') = \langle \phi_{\mathcal{F}}(G), \phi_{\mathcal{F}}(G') \rangle$$

R-convolution kernel

$$k(G,G') = \sum_{\vec{G},\vec{G'}} \prod_{d=1}^{D} k_d(G_d, G_d)$$

sub-structures

linear patterns

non-linear patterns

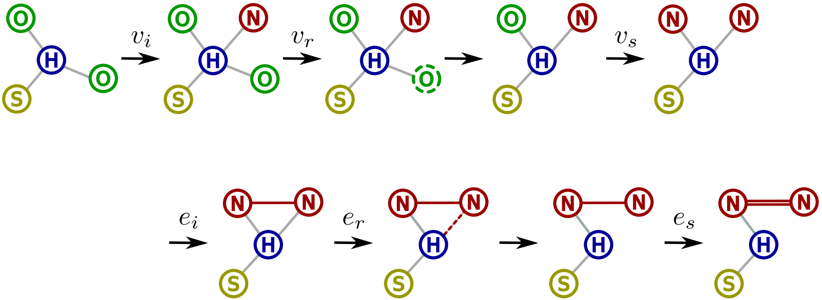
Graph kernels comparison

Table: Characteristics of graph kernels based on linear patterns and two on non-linear patterns.

Kernels	Substructures			Labeling				Directed	Edge Weighted	Computational Complexity (Gram Matrix)	Explicit Representation	Weighting
				symbolic		non-symbolic						
	linear	non-linear	cyclic	vertices	edges	vertices	edges					
Common walk	✓	✗	✗	✓	✓	✗	✗	✓	✗	$\mathcal{O}(N^2n^6)$	✗	a priori
Marginalized	✓	✗	✗	✓	✓	✗	✗	✓	✗	$\mathcal{O}(N^2rn^4)$	✗	✗
Sylvester equation	✓	✗	✗	✗	✗	✗	✗	✓	✓	$\mathcal{O}(N^2n^3)$	✗	a priori
Conjugate gradient	✓	✗	✗	✓	✓	✓	✓	✓	✓	$\mathcal{O}(N^2rn^4)$	✗	a priori
Fixed-point iterations	✓	✗	✗	✓	✓	✓	✓	✓	✓	$\mathcal{O}(N^2rn^4)$	✗	a priori
Spectral decomposition	✓	✗	✗	✗	✗	✗	✗	✓	✓	$\mathcal{O}(N^2n^2 + Nn^3)$	✗	a priori
Shortest path	✓	✗	✗	✓	✗	✓	✗	✓	✓	$\mathcal{O}(N^2n^4)$	✗	✗
Structural shortest path	✓	✗	✗	✓	✓	✓	✓	✓	✗	$\mathcal{O}(hN^2n^4 + N^2nm))$	✗	✗
Path kernel up to length h	✓	✗	✗	✓	✓	✗	✗	✓	✗	$\mathcal{O}(N^2h^2n^2d^{2h})$	✓	✓
Treelet	✓	✓	✗	✓	✓	✗	✗	✓	✗	$\mathcal{O}(N^2nd^5)$	✓	✓
WL subtree	✓	✓	✗	✓	✗	✗	✗	✓	✗	$\mathcal{O}(Nhm + N^2hn)$	✓	✗

- The "Computational complexity" column is a rough estimation for computing the Gram matrix.
- The "Explicit representation" column indicates whether the embedding of graphs in the representation space can be encoded by a vector explicitly; in other words, whether the patterns of graph kernels can be explicitly presented.
- The "Weighting" column indicates whether the substructures can be weighted in order to obtain a similarity measure adapted to a problem of particular prediction, where "a priori" indicates that the weights are set while constructing kernels.

Graph edit distances



$$(v_{i_0}, v_{r_0}, v_{s_0}, v_{s_1}, v_{r_1} \ldots) \rightarrow \pi$$

$$GED(G_1, G_2) = \min_{\pi_1,\ldots,\pi_k \in \Pi(G_1,G_2)} \sum_{i=1}^{k} c(\pi_i)$$

Graph edit distances

$$c_{vr} \rightarrow n_{vr}, \qquad c_{vi} \rightarrow n_{vi}, \qquad c_{vs} \rightarrow n_{vs}$$

$$c_{er} \rightarrow n_{er}, \qquad c_{ei} \rightarrow n_{ei}, \qquad c_{es} \rightarrow n_{es}$$

$$\longrightarrow$$

$$\boldsymbol{\omega} = \left[n_{vr}, n_{vi}, n_{vs}, n_{er}, n_{ei}, n_{es}\right]^\top$$

$$\boldsymbol{c} = \left[c_{vr}, c_{vi}, c_{vs}, c_{er}, c_{ei}, c_{es}\right]^\top$$

$$d_{GED}(G_i, G_j) = \boldsymbol{\omega}^\top \boldsymbol{c}$$

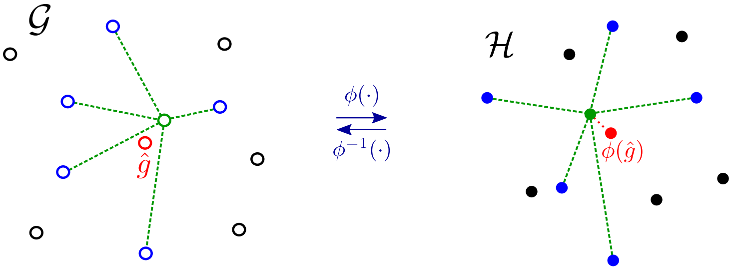
References

- Graph kernels:
  - Jia, L., Gaüzère, B., Honeine, P., 2019. Graph Kernels Based on Linear Patterns: Theoretical and Experimental Comparisons. URL: https://hal-normandie-univ.archives-ouvertes.fr/hal-02053946. working paper or preprint.
  - Jia, L., Gaüzère, B., Honeine, P., 2020. graphkit-learn: A Python Library for Graph Kernels Based on Linear Patterns. (submitted to *Pattern Recognition Letters*.)

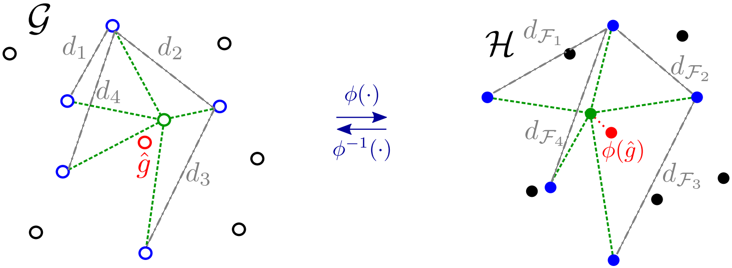
- Graph edit distances:
  - Blumenthal, D.B., Boria, N., Gamper, J., Bougleux, S., Brun, L., 2019a. Comparing heuristics for graph edit distance computation. The VLDB Journal, 140.

Overview

Goal: construct graph pre-image



Align graph space and kernel space



$$d_1 = d_{\mathcal{F}_1}, \quad d_2 = d_{\mathcal{F}_2}, \quad d_3 = d_{\mathcal{F}_3}, \quad d_4 = d_{\mathcal{F}_4}, \quad \ldots$$

Align graph space and kernel space

$$d_{GED}(G_i, G_j) = \boldsymbol{\omega}^\top \boldsymbol{c}$$

$$=$$

$$d_{\mathcal{F}}(\phi(G_i), \phi(G_j)) = \sqrt{k(G_i, G_i) + k(G_j, G_j) - 2k(G_i, G_j)}$$

$$\longrightarrow$$

$$\arg\min_{\boldsymbol{c},\boldsymbol{\omega}} \sum_{i,j=1}^{N} \left(d_{GED}^{i,j} - d_{\mathcal{F}}^{i,j} \right)^2$$

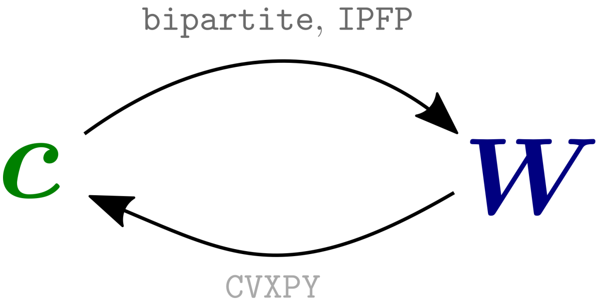
Align graph space and kernel space

$$\longrightarrow$$

$$\begin{aligned} \underset{\boldsymbol{c},\boldsymbol{W}}{\arg\min} \quad & \|\boldsymbol{W}^\top \boldsymbol{c} - \boldsymbol{d}_{\mathcal{F}}\|^2 \\ \text{subject to} \quad & \boldsymbol{c} > \mathbf{0} \\ & c_{vr} + c_{vi} \geq c_{vs} \\ & c_{er} + c_{ei} \geq c_{es} \end{aligned}$$

- $\boldsymbol{W}^\top$: the N^2-by-6 matrix with rows $\boldsymbol{\omega}(i,j)^\top$;
- $\boldsymbol{d}_{\mathcal{F}}$: the vector of N^2 entries $d_{\mathcal{F}}(\phi(G_i), \phi(G_j))$, for $i, j = 1, \ldots, N$.

Align graph space and kernel space



$$\longrightarrow \quad \boldsymbol{c}_{optimized}$$

Generate graph pre-image

Using $\boldsymbol{c}_{optimized}$, alternately iterate the following procedures:

$$\begin{tikzcd}
\bar{g} \arrow[r, bend left, "\texttt{bipartite, IPFP}"] & \bar{\pi}_p \arrow[l, bend left, "\texttt{BCU updater}"]
\end{tikzcd}$$

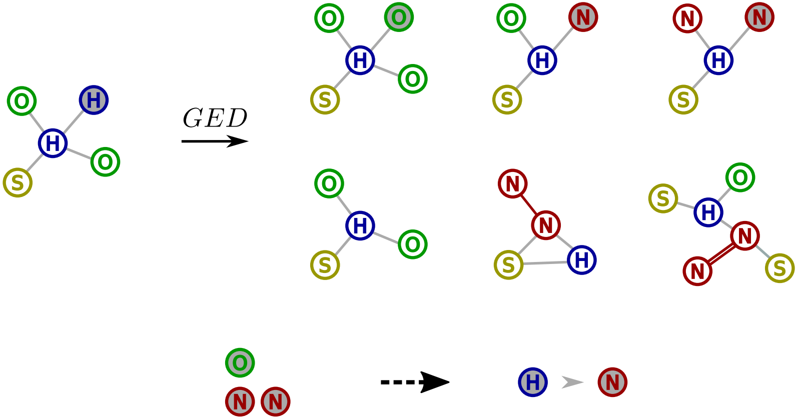
$$\longrightarrow \quad g^{\star}$$

Generate median graph via Block Coordinate Update (BCU)

$$\bar{g} = (\bar{\varphi}, \bar{A}, \bar{\Phi}) \leftarrow \arg \min_{\substack{\varphi \in \mathbb{F}_v^{\bar{n}} \\ A \in \{0,1\}^{\bar{n} \times \bar{n}} \\ \Phi \in \mathbb{F}_e^{\bar{n} \times \bar{n}}}} \sum_{p=1}^{|\mathcal{G}|} \left(c_v(\bar{\pi}_p, \varphi, \varphi_p) + \frac{1}{2} c_e(\bar{\pi}_p, A, \Phi, A_p, \Phi_p) \right)$$

$$\forall p \in \{1, \dots, |\mathcal{G}|\}, \bar{\pi}_p \leftarrow \arg \min_{\pi_p \in \Pi(\bar{g}, g_p)} c(\pi_p, \bar{G}, G_p)$$

Block Coordinate Update (BCU)



Block Coordinate Update (BCU)

- Nodes:
  - Each node is labeled with one of the most present labels among the ones substituted to it.
  - The optimal attribute for a node is given by the mean attribute of the nodes substituted to it.
- Edges:
  - Each edge is labeled with one of the most present labels among the ones substituted to it.
  - The optimal attribute for an edge is given by the mean attribute of the edges substituted to it.

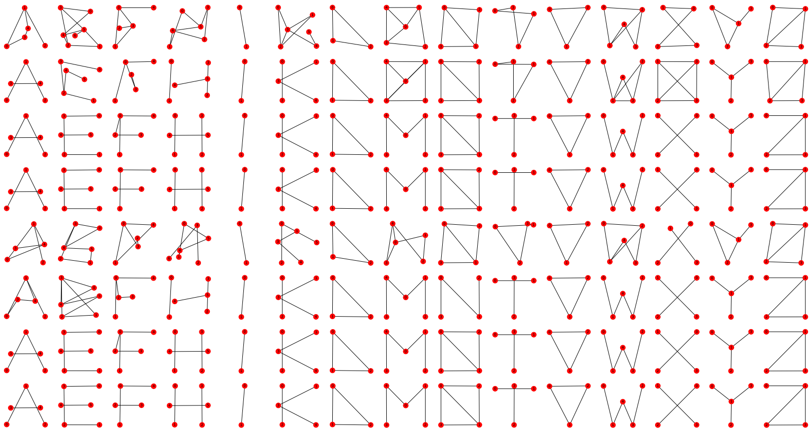
Overview

Distances in kernel space

Datasets	Graph Kernels	Algorithms	$d_{\mathcal{F}}$ GM	Running Times		
				Optimization	Generation	Total
Letter-high	shortest paths	From median set	0.469	-	-	-
		IAM (random costs)	0.467	-	142.59	142.59
		IAM (expert costs)	**0.451**	-	30.31	30.31
		IAM (optimized costs)	0.460	5968.92	26.55	5995.47
	structural sp	From median set	0.459	-	-	-
		IAM (random costs)	0.435	-	30.22	30.22
		IAM (expert costs)	**0.391**	-	29.71	29.71
		IAM (optimized costs)	**0.394**	24.79	25.60	50.39
Letter-med	shortest paths	From median set	0.469	-	-	-
		IAM (random costs)	0.303	-	25.61	25.61
		IAM (expert costs)	**0.288**	-	26.93	26.93
		IAM (optimized costs)	**0.288**	23.72	24.79	48.52
	structural sp	From median set	0.478	-	-	-
		IAM (random costs)	0.286	-	24.77	24.77
		IAM (expert costs)	**0.248**	-	27.51	27.51
		IAM (optimized costs)	**0.248**	27.06	29.24	56.30
Letter-low	shortest paths	From median set	0.166	-	-	-
		IAM (random costs)	**0.116**	-	26.47	26.47
		IAM (expert costs)	**0.116**	-	24.87	24.87
		IAM (optimized costs)	**0.116**	26.35	29.97	56.31
	structural sp	From median set	0.148	-	-	-
		IAM (random costs)	0.103	-	30.22	30.22
		IAM (expert costs)	**0.086**	-	29.43	29.43
		IAM (optimized costs)	0.104	21.95	24.59	46.53

Pre-images constructed by different algorithms

Pre-images generated as median graphs for each letter of *Letter-high* dataset using random costs, expert costs and optimized costs:



Overview

- Conclusion:
  - We proposed a novel method to estimate graph pre-images by aligning kernel space and graph space.
  - Our method is able to generate better pre-images than other methods, as demonstrated on the *Letter-high* dataset.

- Future work:
  - Generalize our method to construct pre-images as arbitrary graphs.
  - Improve both BCU algorithm and the distance alignment.

Questions

Thank you.

Any questions?