A Metric Learning Approach to Graph Edit Costs for Regression

Linlin Jia, linlin.jia@insa-rouen.fr

Florian Yger, florian.yger@dauphine.fr

Benoit Gaüzère, benoit.gauzere@insa-rouen.fr

Paul Honeine, paul.honeine@univ-rouen.fr

Normandie Université, INSA Rouen et Université de Rouen, LITIS Lab
Université Paris Dauphine-PSL, LAMSADE

Acknowledgements

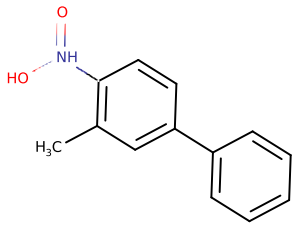
21/01/2021

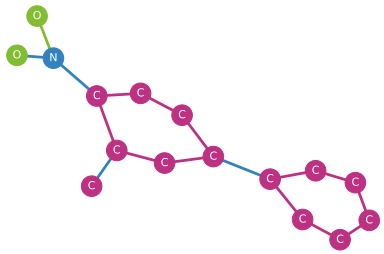
Overview

Graph data

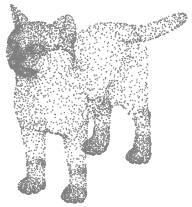
Original data **Graph representation** **Original data** **Graph representation**

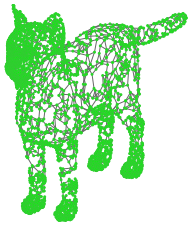
chemoinformatics: molecule



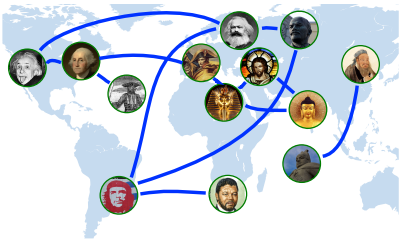


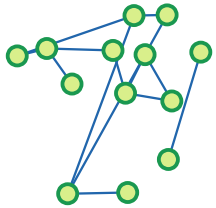
computer vision: 3D point cloud



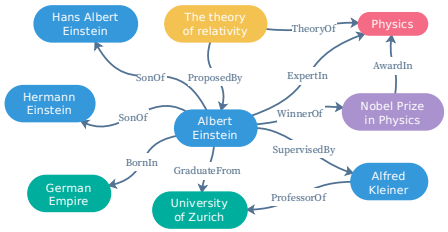


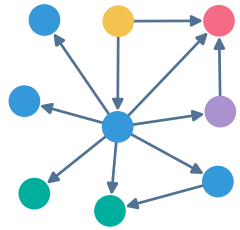
social media: social network



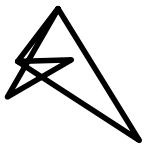


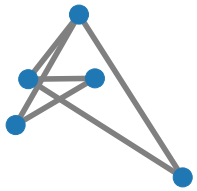
knowledge graph





computer vision: handwriting

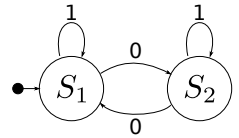




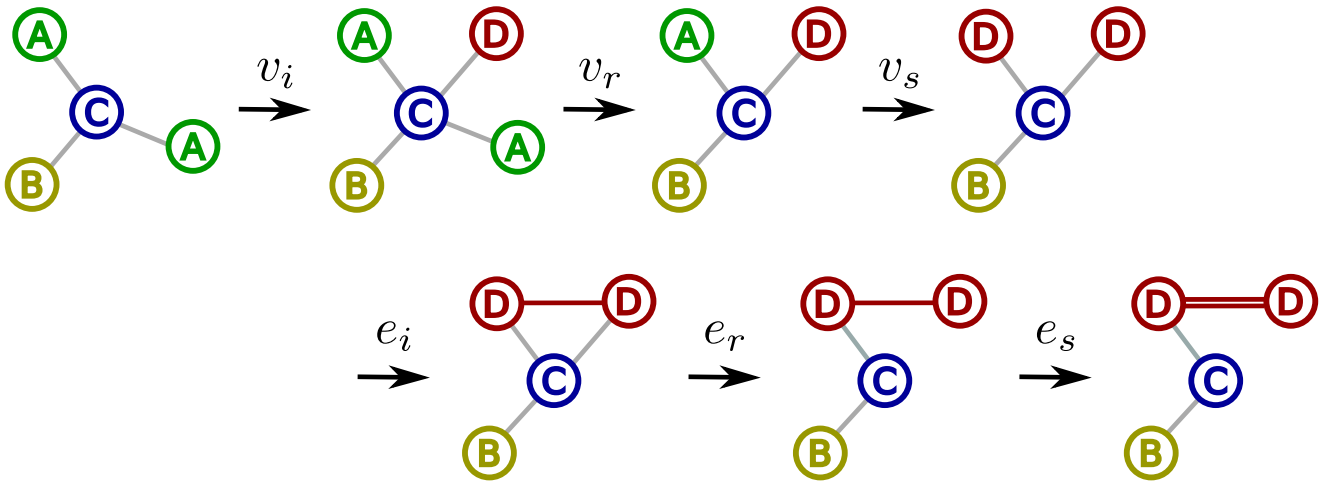
state transition

State-transition table

Current state \ Input	0	1
S_1	S_2	S_1
S_2	S_1	S_2



Graph edit distances



$$(v_{i_0}, v_{r_0}, v_{s_0}, v_{s_1}, v_{r_1} \dots) \to \pi \quad \longrightarrow \quad \texttt{ged}(G_1, G_2) = \min_{\pi_1,\dots,\pi_k \in \Pi(G_1,G_2)} \sum_{i=1}^{k} c(\pi_i)$$

$$\begin{cases} \boldsymbol{x} = [n_{vr}, n_{vi}, n_{vs}, n_{er}, n_{ei}, n_{es}]^\top \\ \boldsymbol{c} = [c_{vr}, c_{vi}, c_{vs}, c_{er}, c_{ei}, c_{es}]^\top \end{cases} \longrightarrow \quad \texttt{ged}(G_i, G_j) = \boldsymbol{x}^\top \boldsymbol{c}$$

Overview

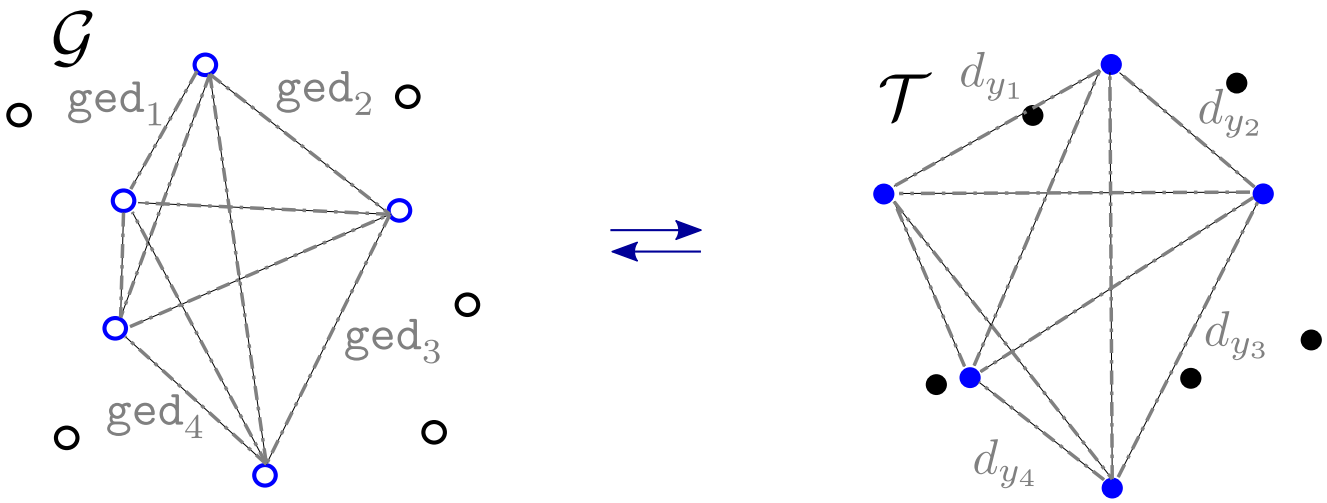
Methods to learn edit costs

- **Set manually (expert costs):**
  - domain knowledge implied a priori
  - costs required in datasets
  - data information not well explored
  - hard to generalize
- **Learn costs:**
  - high computational complexity
  - restricted to strings and trees
  - require a ground truth mapping
  - literature on metric learning for graph data is limited

-> Our method: Learn edit cost by aligning GEDs and distances between targets.

Overview

Align graph space and the target space



$$\mathtt{ged}_1 = d_{y_1}, \quad \mathtt{ged}_2 = d_{y_2}, \quad \mathtt{ged}_3 = d_{y_3}, \quad \mathtt{ged}_4 = d_{y_4}, \quad \dots$$

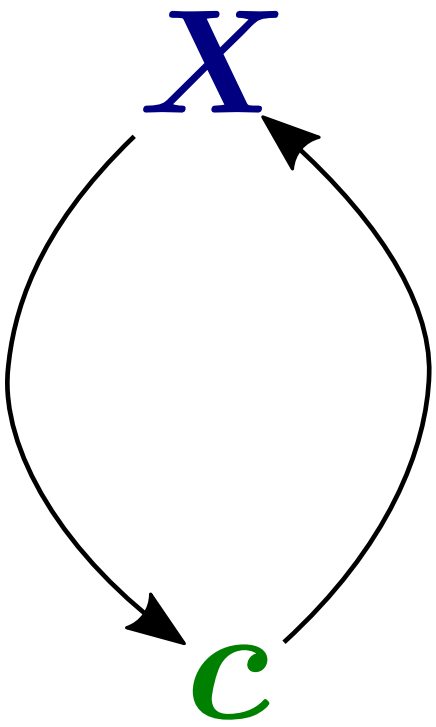
$$\begin{cases} \mathtt{ged}(G_i, G_j) = \boldsymbol{x}^\top \boldsymbol{c} \\ d_y(G_i, G_j) = \|y(G_i) - y(G_j)\|_2 \end{cases}$$

$$\longrightarrow \quad \underset{\boldsymbol{c}, \boldsymbol{x}}{\arg\min} \sum_{i,j=1}^{N} \left(\mathtt{ged}^{i,j} - d_y^{i,j} \right)^2$$

Align graph space and the target space

$$\arg\min_{\boldsymbol{c}>0} \quad \|\boldsymbol{X}^\top \boldsymbol{c} - \boldsymbol{d}_y\|^2$$

`CVXPY, scipy`



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

$$\forall\, G_i, G_j,$$

$$\mathsf{ged}(G_i, G_j) = \boldsymbol{x}(i,j)^\top \boldsymbol{c}$$

`bipartite, IPFP`

- $\boldsymbol{X}^\top$: the N^2-by-6 matrix with rows $\boldsymbol{x}(i,j)^\top$;
- $\boldsymbol{d}_y$: the vector of N^2 entries $d_y(G_i, G_j)$, for $i, j = 1, \ldots, N$.

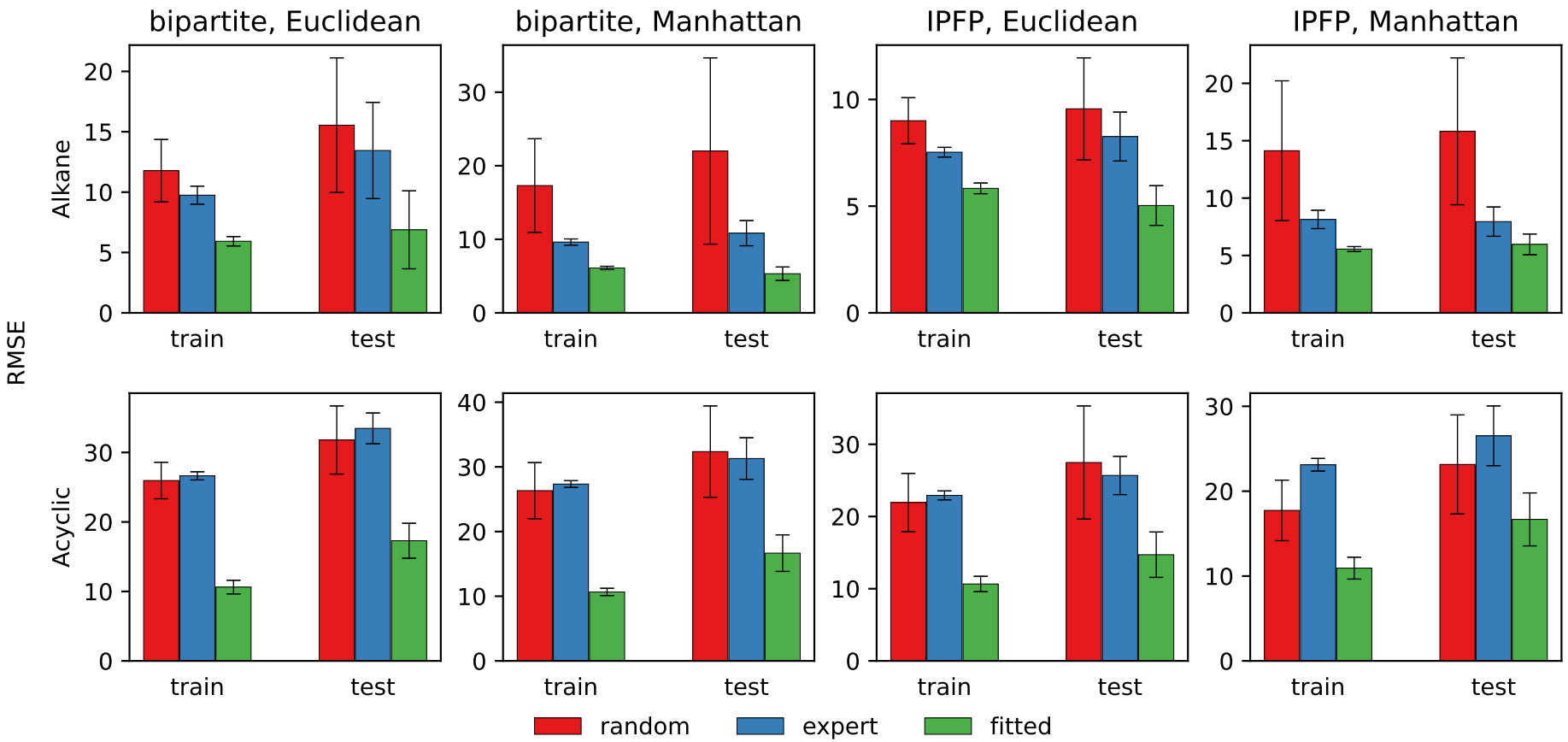
Overview

Experiment settings

- Datasets:
  - molecules and their boiling points as targets
  - the `Alkane` dataset: unlabeled
  - the `Acyclic` dataset: nodes with discrete labels
- Configurations:
  - K-NN regression problem
  - m`bipartite` and m`IPFP` with $m = 40$
  - outer CV: 90% training + 10% test, inner CV: 5-fold
  - k optimized in inner CV over $\{3, 5, 7, 9, 11\}$
  - maximum number of iterations: 5

Performance

Results on each dataset in terms of RMSE for the 10 splits, measured on the training and on the test sets:



Performance

A different representation of the same results:

Dataset	Distance	Method	bipartite		IPFP	
			Train errors	Test errors	Train errors	Test errors
Alkane	Euclidean	random	11.77±2.59	15.54±5.58	9.001±1.08	9.56±2.39
		expert	9.75±0.75	13.45±3.98	7.53±0.23	8.26±1.15
		fitted	**5.93±0.39**	**6.88±3.23**	**5.83±0.25**	**5.03±0.93**
	Manhattan	random	17.30±6.36	22.00±12.66	14.13±6.09	15.82±6.40
		expert	9.63±0.42	10.83±1.71	8.14±0.80	7.95±1.28
		fitted	**6.10±0.23**	**5.32±0.91**	**5.56±0.22**	**5.97±0.90**
Acyclic	Euclidean	random	25.96±2.63	31.79±4.90	21.94±4.03	27.48±7.84
		expert	26.63±0.59	33.46±2.22	22.92±0.62	25.68±2.65
		fitted	**10.62±0.98**	**17.29±2.52**	**10.66±1.06**	**14.71±3.14**
	Manhattan	random	26.33±4.34	32.36±7.06	17.73±3.57	23.15±5.83
		expert	27.34±0.52	31.29±3.22	23.11±0.74	26.53±3.52
		fitted	**10.66±0.56**	**16.66±2.83**	**10.93±1.29**	**16.68±3.12**

Performance

Table: Average and standard deviation of fitted edit costs values

Dataset	Edit cost	Distance	c_{ni}	c_{nr}	c_{ns}	c_{ei}	c_{er}	c_{es}
Alkane	bipartite	Euclidean	26.45±0.48	26.24±0.60	-	0.13±0.06	0.14±0.09	-
		Manhattan	26.67±0.37	26.63±0.58	-	0.11±0.04	0.11±0.06	-
	IPFP	Euclidean	26.12±0.24	25.88±0.25	-	0.74±0.23	0.78±0.23	-
		Manhattan	25.94±0.38	25.71±0.44	-	0.89±0.30	0.77±0.29	-
Acyclic	bipartite	Euclidean	13.81±0.48	13.83±0.80	10.46±0.40	1.37±0.46	1.45±0.46	1.41±0.09
		Manhattan	13.76±0.39	14.14±0.57	10.28±0.44	1.44±0.20	1.45±0.19	1.45±0.07
	IPFP	Euclidean	11.61±0.45	11.68±0.43	11.07±0.53	4.49±0.30	4.46±0.24	4.48±0.18
		Manhattan	11.52±0.40	11.40±0.40	10.61±0.52	4.50±0.31	4.50±0.31	4.50±0.10

- Insertion and deletion costs are almost similar, hence showing the symmetry of these operations.
- Deletion and insertion costs are more important than substitution costs, which shows that the number of atoms is more important than the atom itself.

Overview

- Conclusion:
  - learn optimal graph edit costs for regression
  - metric learning, space alignment
  - alternated optimization strategy
  - experiments show promising results
- Future work:
  - comparison to other methods
  - other criteria (besides the distance-preserving criterion)
  - convergence proof
  - to classification problem
  - to non-constant costs

Questions

Thank you.

Any questions?