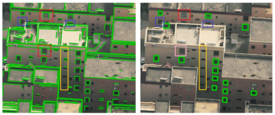
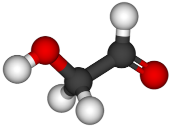
Graph Classification

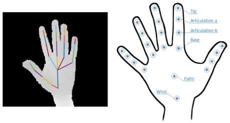
August 16, 2019

Pros

- Interest: provide a compact encoding of both :
  - a decomposition of objects into meaningful sub-parts,
  - the relationships between these sub-parts.
- Applications:
  - Image processing: segmentation, boundary detection,
  - Pattern Recognition: printed characters, documents, objects (buildings, brain structures), faces, gestures, molecules,. . . ,
  - Image registration,
  - Understanding of structured scenes.
  - . . .

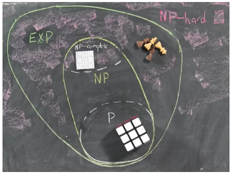






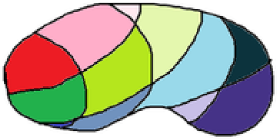
Cons

- Even simple questions are difficult:
  - Are this two graphs the same ?
    - $\mathcal{NP}$-intermediate
  - Is this graph a sub-part of this graph ?
    - $\mathcal{NP}$-complete
  - What is the distance between these two graphs ?
    - $\mathcal{NP}$-hard (for the usual distance)
  - What is the mean/median of a set of graphs ?
    - $\mathcal{NP}$-hard (for the usual distance)



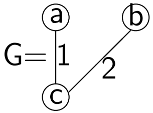
- Two main research lines to avoid exponential computation times:
  1. Use alternative metrics/embedding
  2. Use heuristics.

Graph space as an Orbifold



Orbifolds theory: basics

- Let $\mathcal{T}$ be the manifold of $n \times n$ matrices.
- One graph may have multiple matrix representations



$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & a & 0 & 1 \\ 2 & 0 & b & 2 \\ 3 & 1 & 2 & c \end{array} \quad \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & c & 1 & 2 \\ 2 & 1 & a & 0 \\ 3 & 2 & 0 & b \end{array} \quad \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & b & 2 & 0 \\ 2 & 2 & c & 1 \\ 3 & 0 & 1 & a \end{array} \ldots$$

x y z...

- Let $\mathcal{P}$ denote the set of $n \times n$ permutation matrices:

$$\forall (x,y) \in \mathcal{T}\ x \sim y \Leftrightarrow \exists P \in \mathcal{P} | x = P^t y P$$

- $\mathcal{T}/\sim$ is called an orbifold
- A graph G is encoded in $\mathcal{T}/\sim$ by $[X_G] = \{x, y, z, \ldots\}$.

Orbifolds theory: basics

- Let X and Y denote two elements of $\mathcal{T}/\sim$:

$$\left\{\begin{array}{lcl} <X,Y> & = & \max\{<x,y> \ x\in X, y\in Y\} \\ \delta(X,Y) & = & \sqrt{\|X\|^2+\|Y\|^2-2<X,Y>} \\ & = & \min_{x\in X, y\in Y}\|x-y\| \end{array}\right.$$

- Let $\mathcal{G}$ denotes the set of graphs:

$$\omega : \left(\begin{array}{ccc} \mathcal{G} & \rightarrow & \mathcal{T}/\sim \\ X & \mapsto & [X] \end{array}\right.$$

is a bijective isometry between the metric spaces $(\mathcal{G},\delta)$ and $(\mathcal{T}/\sim, d)$ (d: deduced from L_2 norm)

Orbifold theory: properties

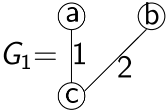
- Using real attributes, the space $(\mathcal{G}, \delta)$ is:
  - A complete metric space,
  - a geodesic space,
  - locally compact,
  - every closed bounded subset of $(\mathcal{G}, \delta)$ is compact.
- The sample mean of a set of graphs always exists. For each set of graphs $\{X_1, \dots, X_n\}$, this mean is defined by:

$$M = \sum_{i=1}^{n} x_i \text{ where } x_i \in X_i$$

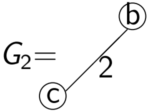
- Under some conditions, the set of sample means reduces to a singleton.
- Applications:
  - Computation of the sample mean [Jain, 2016]
  - Central clustering algorithms [Jain and Wysotzki, 2004],
  - Generalized linear classifiers [Jain, 2014]

Orbifold theory: limits

- Let us consider both graphs:



a	0	1
0	b	2
1	2	c



0	0	0
0	b	2
1	2	c

- We have:

$$\delta(G_1, G_2) = \sqrt{a^2 + 1}$$

- The cost of removing a vertex or an edge depends on its label,
- The substitution costs are induced by labels.
- Graph metric is induced by graph representation: Both concepts can not be distinguished.

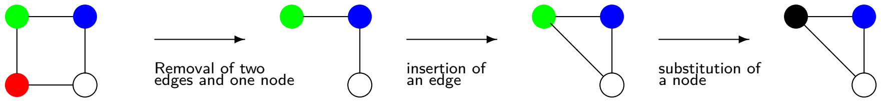
$\boxed{\text{We should search for a more flexible metric.}}$

Graph Edit distance

Edit Paths

Definition (Edit path)

Given two graphs G_1 and G_2 an **edit path** between G_1 and G_2 is a sequence of node or edge removal, insertion or substitution which transforms G_1 into G_2.



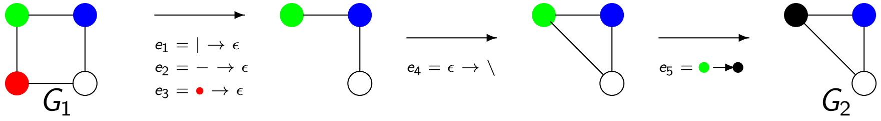
A substitution is denoted $u \rightarrow v$, an insertion $\epsilon \rightarrow v$ and a removal $u \rightarrow \epsilon$.

Alternative edit operations such as merge/split have been also proposed[Ambauen et al., 2003].

Costs

Let $c(.)$ denote the cost of any elementary operation. The cost of an edit path is defined as the sum of the costs of its elementary operations.

- All cost are positive: $c() \geq 0$,
- A node or edge substitution which does not modify a label has a 0 cost: $c(l \to l) = 0$.


G_1
$e_1 = | \to \epsilon$
$e_2 = - \to \epsilon$
$e_3 = \bullet \to \epsilon$
$e_4 = \epsilon \to \backslash$
$e_5 = \bullet \to \bullet$
G_2

If all costs are equal to 1, the cost of this edit path is equal to 5.

Conversely to graph alignment metric, costs allow to distinguish graph representation and graph metrics.

Graph edit distance

Definition (Graph edit distance)

The graph edit distance between G_1 and G_2 is defined as the cost of the less costly path within $\Gamma(G_1, G_2)$. Where $\Gamma(G_1, G_2)$ denotes the set of edit paths between G_1 and G_2.

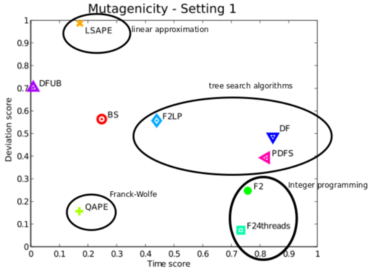
$$d(G_1, G_2) = \min_{\gamma \in \Gamma(G_1, G_2)} \sum_{e \in \gamma} c(e)$$

Graph edit distance: Main methods

- A^* like algorithms [Abu-Aisheh, 2016],
- Formulation as a quadratic problem [Bougleux et al., 2017] solved by Franck-Wolfe [Frank and Wolfe, 1956] like algorithms (see also:
  [Liu and Qiao, 2014, Boria et al., 2018, Daller et al., 2018].
- Integer Programming [Lerouge et al., 2017, Darwiche, 2018]
- Fast (and often rough)
  approximations [Riesen and Bunke, 2009, Gaüzère et al., 2014, Carletti et al., 2015, Blumenthal et al., 2018]

Graph edit distance: Comparison

- Relative comparison:



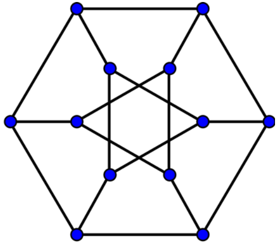
- Absolute results on PAH dataset (unlabeled chemical graphs):

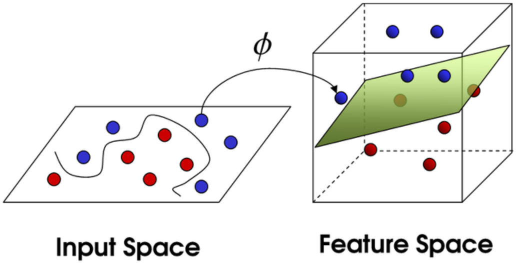
methods	mean distance	execution time
Linear approx. [Riesen and Bunke, 2009]	135.2	10^{-3}
Franck-Wolfe [Bougleux et al., 2017]	45.3	2.73
multi-start Franck-Wolfe [Daller et al., 2018]	29.8	2.96

Graph Edit distance: Conclusion

- 🙂 Allows to dissociate graph representation and graph metric.
- 🙂 Constitutes a fine and intuitive metric between graphs.
- ☹ It is $\mathcal{NP}$-hard to compute,
- ☹ It is not conditionally definite negative (Krein space).
    - Most of machine learning machinery should be adapted [Loosli et al., 2016].
    - Weak properties: E.g. the median is usually not unique.

Graph Kernels





Kernels : Definition

- A kernel k is a **symmetric** similarity measure on a set χ

$$\forall (x,y) \in \chi^2,\ k(x,y) = k(y,x)$$

- k is said to be **definite positive** (d.p.) iff k is symmetric and iff:

$$\left.\begin{array}{rcl} \forall (x_1,\dots,x_n) & \in & \chi^n \\ \forall (c_1,\dots,c_n) & \in & \mathbb{R}^n \end{array}\right\} \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(x_i,x_j) \geq 0$$

- $K = (k(x_i,x_j))_{(i,j)\in\{1,\dots,n\}}$ is the Gramm matrix of k. k is d.p. iff:

$$\forall c \in \mathbb{R}^n - \{0\},\ c^t K c \geq 0$$

Kernels and scalar products

Aronszajn 1950 :

A kernel k is d.p. on a space χ
if and only if
it exists

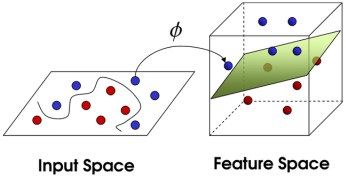
- one Hilbert space $\mathcal{H}$ and
- a function $\varphi : \chi \to \mathcal{H}$

such that:

$$k(x, y) =< \varphi(x), \varphi(y) >$$

Kernels: So what ?

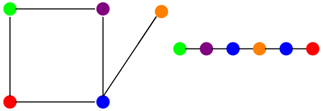
- In Machine Learning, function φ allows to implicitly embed our data in high dimensional spaces and thus to get a non linear separation/analysis of data using linear algorithms: We have its cake and eat it.



- **Much stronger interest for graphs:** It provides an implicit embedding of graphs into an Hilbert space and open the way for rich interactions between graphs and usual machine learning methods: SVM, kPCA, MKL,...

Walk kernels

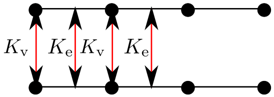
- Walks: Let $G = (V, E)$. $W = (v_1, \dots, v_n)$ is a walk iff $(v_i, v_{i+1}) \in E, \forall i \in \{1, \dots, n-1\}$.



is a walk

- Kernel between walks

$$K(h, h') = \begin{cases} 0 & \text{if } |h| \neq |h'| \\ K_v(v_1, v_1').\Pi_{i=1}^{|h|} K_e(e_i, e_i') K_v(v_{i+1}, v_{i+1}') & \text{otherwize} \end{cases}$$



Walks kernels

- Walk kernels :

$$K(G_1, G_2) = \sum_{h \in \mathcal{W}(G_1)} \sum_{h' \in \mathcal{W}(G_2)} K(h, h') \lambda_{G_1}(h) \lambda_{G_2}(h')$$

- Covers different Graph kernels[Vert 2007,Vishwanathan et al. 2010]:

$$If\ \lambda_G(h) = \begin{cases} 1\ iff\ |h| = n & K \text{is a nth order walk kernel} \\ P_G(h)(Markov\ RW) & K \text{is a random walk/marginaliz} \\ \beta^{|h|} & K \text{is a geometric kernel} \end{cases}$$

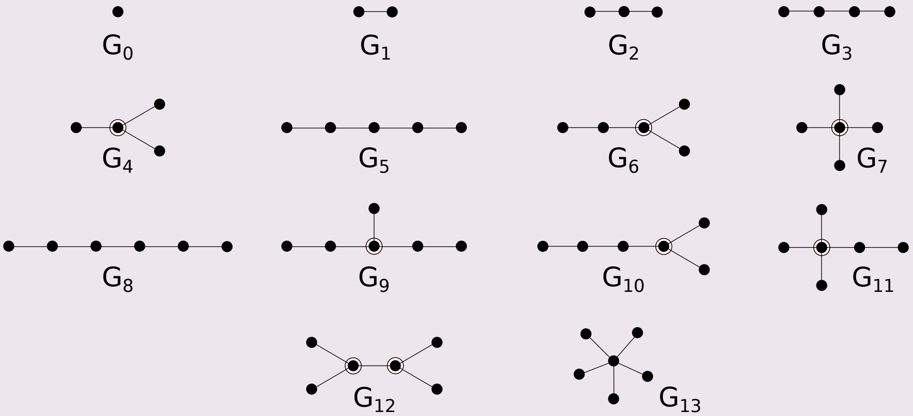
$$P_G(h) = p_s(h_1)\Pi_{i=1}^{n} p_t(h_i|h_{i-1}) p_q(h_n) \text{ with } |h| = n$$

- Walks may induce totering problems: Walks with arbitrary length on the same set of edges and vertices.
- Framework extended to tree-pattern [Vert 2006, Bach 2007]

Bag of patterns : unlabeled treelets

- One step beyong bag of paths : bags of treelets (trees of depth at most 6).

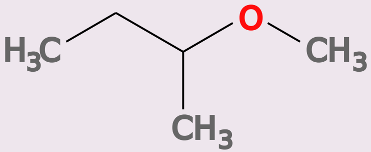
Treelets



Bag of labeled treelets

Treelet code

- Two parts :
  - Structural part : Treelet type $(G_0, G_1, \dots)$
  - Label part : Canonical traversal of the treelet

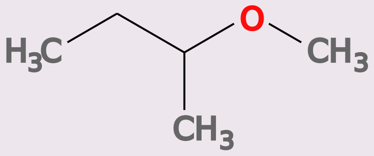


Code : G09-C1C1C1O1C1C

- $Code(G) = Code(G') \Leftrightarrow G \simeq G'$

Kernel definition

From graph to histogram



(a) Molecule

$\tau(f_\tau(G))$		
C(5)	O(1)	
C-C(3)	C-O(2)	
C-C-C(2)	C-C-O(2)	C-O-C(1)
C-$\underset{\substack{\vert \\ \mathrm{C}}}{\mathrm{C}}$-O(1)		
	$\vdots$	

(b) Histogram

Kernel definition

$$K(G_1, G_2) = \sum_{\tau \in \mathcal{B}(G_1) \cap \mathcal{B}(G_2)} K(f_\tau(G_1), f_\tau(G_2))$$

Selection of relevant patterns

- Some Facts :
  - Our first solution considers bags containing all patterns,
  - The relevance of a given treelet should be fixed a posteriori.
- MKL [Sonnenburg et al., 2006, Rakotomamonjy et al., 2007] :

$$\text{If : } \; K(x,y) = \sum_{i=1}^{n} w_i K_i(x,y)$$

  MKL allows to fix $(w_i)_{i\in\{1,\dots,n\}}$ optimally.
- Our kernel :

$$K(G_1,G_2) = \sum_{\tau\in\mathcal{B}(G_1)\cap\mathcal{B}(G_2)} K(f_\tau(G_1), f_\tau(G_2)) \overset{not.}{=} \sum_{\tau\in\mathcal{B}(G_1)\cap\mathcal{B}(G_2)} K_\tau(G_1,G_2)$$

- A direct application of the MKL method provides the new kernel:

$$K(G_1,G_2) = \sum_{\tau\in\mathcal{B}(G_1)\cap\mathcal{B}(G_2)} w_\tau K(f_\tau(G_1), f_\tau(G_2))$$

  where w_τ is defined using MKL.

Some results

Method	RMSE		Familly
Gaussian edit distance	10.27		Graph edit distance
Random Walks	18.72		Infinite bag of walks
Tree Pattern Kernel	11.02		Infinite bag of tree patterr
Treelet Kernel (TK)	8.10		Finite bag of tree patterns
TK + MKL	**5.24**		

Boiling point prediction on acyclic molecule dataset using 90% of the dataset as train set and remaining 10% as test set.

Some results

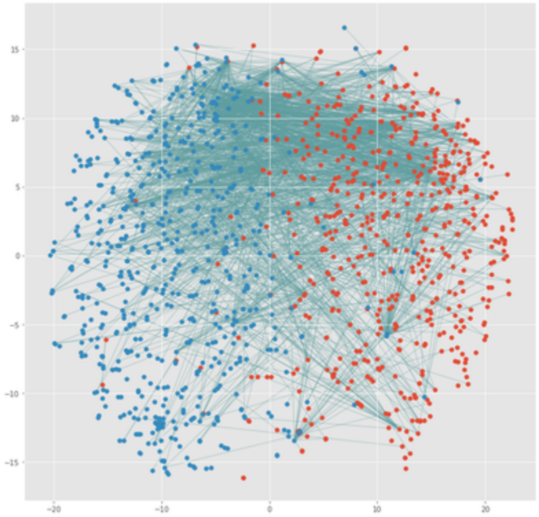
Method	RMSE	Times(s)	Familly
Gaussian edit distance	10.27	1.35	Graph edit distance
Random Walks	18.72	19.10	Infinite bag of walks
Tree Pattern Kernel	11.02	4.98	Infinite bag of tree pattern
Treelet Kernel (TK)	8.10	**0.07**	Finite bag of tree patterns
TK + MKL	**5.24**		

Boiling point prediction on acyclic molecule dataset using 90% of the dataset as train set and remaining 10% as test set.

Graph kernels:

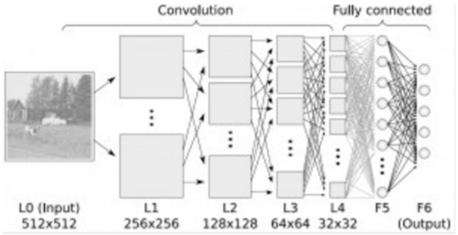
- Pros:
  - Graph kernels provide an implicit embedding of graphs,
  - Its open the way to the application of many statistical tools to graphs,
- Cons:
  - Graph kernels are usually based on a notion of bag which only provides a rough similarity measure.
  - The graph feature extraction process has been moved to the design of a similarity measure (the kernel). Such a measure remains largely “handcrafted”.

Graph Neural Network

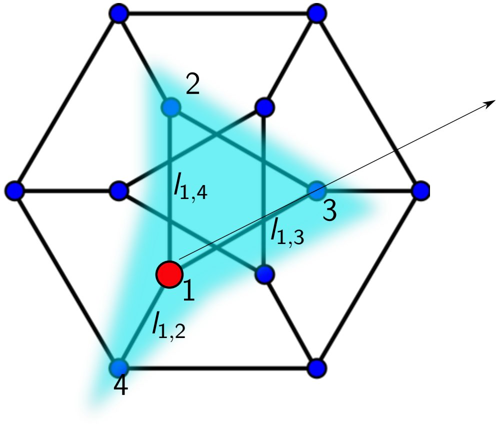


Graph Neural Networks: Three main steps

1. Agregation,
2. Decimation,
3. Pooling



Agregation: The problem



$$h_1 = f_w \left(l_1, \{l_{1,2}, l_{1,3}, l_{1,4}\}, \{h_2, h_3, h_4\}, \{l_2, l_3, l_4\} \right)$$

Graph Convolution

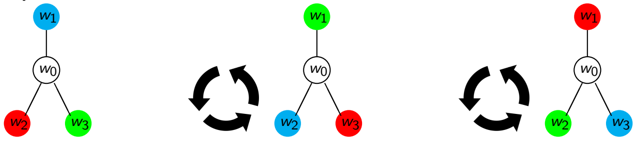
The problem

- Using images we learn $w_0 \ldots, w_8$:

w_5	w_1	w_6
w_3	w_0	w_4
w_7	w_2	w_8

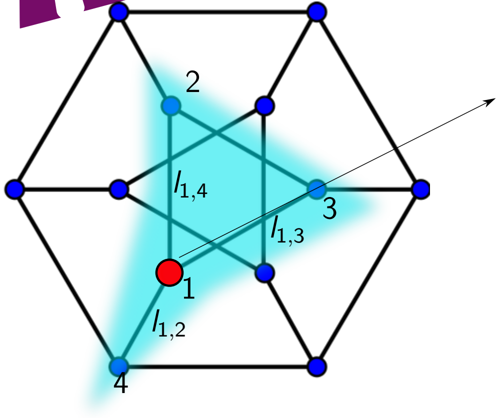
w_1 denotes the weigh of the pixel above the central pixel.

- Using graphs:



Without embedding nothing distinguishes the cyan,red and green neighbors.

Agregation: The problem

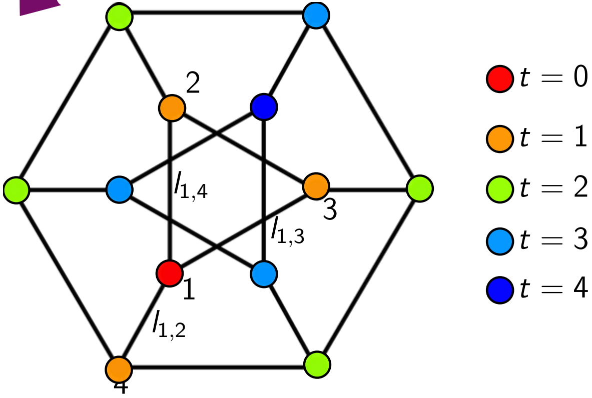


$$h_1 = f_w \left(l_1, \{l_{1,2}, l_{1,3}, l_{1,4}\}, \{h_2, h_3, h_4\}, \{l_2, l_3, l_4\} \right)$$

$$\begin{cases} h_v = f_w(l_v, l_{CON(v)}, h_{\mathcal{N}(v)}, l_{\mathcal{N}(v)}) \\ o_v = g_w(h_v, l_v) \end{cases}$$

with $CON(v) = \{(v, v')\ v' \in \mathcal{N}(v)\}$

Iteration



$$\begin{cases} h_v^t = f_w(l_v, l_{CON(v)}, h_{\mathcal{N}(v)}^{t-1}, l_{\mathcal{N}(v)}) \\ o_v = g_w(h_v^T, l_v) \end{cases}$$

How to become permutation invariant

$$h_v^t = f_w(l_v, l_{CON(v)}, h_{\mathcal{N}(v)}^{t-1}, l_{\mathcal{N}(v)})$$

$$h_v^t = \sum_{v' \in \mathcal{N}(v)} f(l_v, l_{v,v'}, l_{v'}, h_{v'}^{(t-1)})$$

where f may be:

- An affine function [Scarselli et al., 2009],

$$f(l_v, l_{v,v'}, l_{v'}, h_{v'}^{(t-1)}) = A^{(l_v, l_{v,v'}, l_{v'})} h_{v'}^{(t-1)} + b^{(l_v, l_{v,v'}, l_{v'})}$$

- A MLP [Massa et al., 2006]

More complex agregation functions

- A long Short-term Memory [Hochreiter and Schmidhuber, 1997, Peng et al., 2017, Zayats and Ostendorf, 2018]
- A Gated Reccurent Unit [Li et al., 2016]

$$\begin{aligned}
h_v^{(1)} &= [x_v^T, 0] && (1)\\
a_v^{(t)} &= A_v^T [h_1^{(t-1)\ T}, \dots, h_{|V|}^{(t-1)\ T}]^T + b && (2)\\
z_v^t &= \sigma(W^z a_v^{(t)} + U^z h_v^{(t-1)}) && (3)\\
r_v^t &= \sigma(W^r a_v^{(t)} + U^r h_v^{(t-1)}) && (4)\\
\tilde{h}_v^{(t)} &= \tanh\left(W a_v^{(t)} + U\left(r_v^t \odot h_v^{(t-1)}\right)\right) && (5)\\
h_v^t &= (1 - z_v^t) \odot h_v^{(t-1)} + z_v^t \odot \tilde{h}_v^t && (6)
\end{aligned}$$

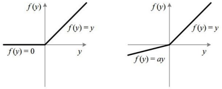
z_v^t: update gate, r_v^t: reset gate, A_v: weight by edges types.

- Learned weight by edge type: $a_v^{(t)} = \sum_{w \in \mathcal{N}(v)} A_{l_{v,w}} h_w^{(t-1)}$ [Gilmer et al., 2017]

Graph attention Networks

- Not all neighbors have a same importance for update:

$$\alpha_{v,v'} = softmax_j(e_{v,v'}) = \frac{exp(e_{v,v'})}{\sum_{v''\in\mathcal{N}_i} exp(e_{v,v''})}$$



- With : $e_{v,v'} = LeakyReLU(a^T[Wh_v||Wh_{v'}])$
  a, W : weight vector and matrix.
- Update rule:

$$h'_v = \sigma(\sum_{v'\in\mathcal{N}_v} \alpha_{v,v'} Wh_{v'})$$

- With K features:

$$h'_v = ||_{k=1}^{K} \sigma(\sum_{v'\in\mathcal{N}_v} \alpha^k_{v,v'} W^k h_{v'})$$

Graph Convolution

The spectral approach [Defferrard et al., 2016]

- Graph Laplacian:

$$L = D - A \text{ with } D_{ii} = \sum_{j=1}^{n} A_{ij}$$

  A adjacency matrix of a graph G.
- Matrix L is real symmetric semi definite positive:

$$L = U \Lambda U^T$$

  U orthogonal, Λ real(positive) diagonal matrix.
- A classical result from signal processing:

$$x * y = \mathcal{F}^{-1}(\hat{x}.\hat{y})$$

  *: convolution operation, $\mathcal{F}^{-1}$ inverse Fourrier transform, $\hat{x}$ fourrier transform of x, '.' term by term multiplication.

Graph Convolution

The spectral approach

- If x is a signal on G, $\hat{x} = U^T x$ can be considered as its "Fourrier" transform. We have:

$$U\hat{x} = UU^T x = x$$

  U is thus the inverse Fourrier transform.
- By analogy:

$$z * x = U(\hat{z} \odot \hat{x}) = U\left(U^T z \odot U^T x\right) = U\left(diag(U^T z)U^T y\right)$$

  $\odot$: Hadamard product.
- Let $g_\theta(\Lambda)$ be a diagonal matrix. The filtering of x by g_θ is:

$$y = U\left(g_\theta(\Lambda)U^T x\right) = \left(Ug_\theta(\Lambda)U^T\right)x$$

Graph convolution

The spectral approach

- If:

$$g_\theta(\Lambda) = \sum_{i=0}^{K-1} \theta_i \Lambda^i$$

Then:

$$y = \left(U g_\theta(\Lambda) U^T\right) x = U \left(\sum_{i=0}^{K-1} \theta_i \Lambda^i\right) U^T x = \left(\sum_{i=0}^{K-1} \theta_i L^i\right) x$$

- One parameter per ring:
    - Lx : one step (direct) neighborhood,
    - L^2x : two step neighborhood (idem for $L^3, L^4, \dots$)
- Problem: Computing L^i for $i \in \{0, \dots, K-1\}$ is problematic for large matrices (SVD computation)

Graph convolution

The spectral approach

- Let us consider Chebyshev polynomial $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0 = 1$ and $T_1(x) = x$.

$$g_\theta(\Lambda) = \sum_{i=0}^{K-1} \theta_i \Lambda^i \rightarrow g_\theta(\Lambda) = \sum_{i=0}^{K-1} \theta_i T_i(\tilde{\Lambda})$$

  $\tilde{\Lambda}$ normalized version of Λ.

- we have:

$$\tilde{x}_k = 2\tilde{L}\tilde{x}_{k-1} - \tilde{x}_{k-2} \text{ with } \tilde{x}_0 = x \text{ and } \tilde{x}_1 = \tilde{L}x$$

  $\mathcal{O}(K|\mathcal{E}|)$ operations to get $\tilde{x}_k$.

- If $K = 2$ it simplifies to [Kipf and Welling, 2017]: $y = \theta L' x$ where L' is a regularized version of the normalized Laplacian.

Graph convolution

Non spectral approaches

- [Simonovsky and Komodakis, 2017]

$$y_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} F_\theta(L(j,i)) x_j + b$$

F: Parametric function of θ which associates one weigh to each edge label $L(j,i)$.

- [Verma et al., 2017]:

$$y_i = \frac{1}{|\mathcal{N}(i)|} \sum_{m=1}^{M} \sum_{j \in \mathcal{N}(i)} q_{\theta_m}(x_j, x_i) W_m x_j + b$$

$q_{\theta_m}(.,.)$ m^{th} learned soft-assignment function. W_m weight matrix.

Graph Propagation

Recurent networks	Convolution
[Hochreiter and Schmidhuber, 1997]	
[Massa et al., 2006]	
[Scarselli et al., 2009]	[Bruna et al., 2014]
[Li et al., 2016]	[Defferrard et al., 2016]
[Gilmer et al., 2017]	[Kipf and Welling, 2017]
[Peng et al., 2017]	[Simonovsky and Komodakis, 2017]
[Zayats and Ostendorf, 2018]	[Verma et al., 2017]

Agregation

What's next ?

- Graph Downsampling, Graph pooling, Graph final decision: Some solutions but still the jungle.

Conclusion

- Three graph metrics associated to different topologies:

Metric	Space	Explicable
Graph alignment metric	Orbifold	✓
Graph edit distance	Krein Space	✓
Graph kernels	Hilbert space	via MKL.

- Graph neural network:
  - Still in their infancy,
  - Not so much explicable,
  - Unknown metric and topological space,
  - A great potential.

Bibliography

Abu-Aisheh, Z. (2016). *Anytime and distributed Approaches for Graph Matching*. PhD thesis, Université François Rabelais, Tours.

Ambauen, R., Fischer, S., and Bunke, H. (2003). Graph edit distance with node splitting and merging, and its application to diatom identification. In *Graph Based Representations in Pattern Recognition: 4th IAPR International Workshop, GbRPR 2003 York, UK, June 30 – July 2, 2003 Proceedings*, pages 95–106, Berlin, Heidelberg. Springer Berlin Heidelberg.

Blumenthal, D., Bougleux, S., Gamper, J., and Brun, L. (2018). Ring based approximation of graph edit distance. In *Proceedins of SSPR'2018*, Beijing. IAPR, Springer.

Boria, N., Bougleux, S., and Brun, L. (2018). Approximating ged using a stochastic generator and multistart ipfp. In Bai, X., Hancock, E. R., Ho, T. K., Wilson, R. C., Biggio, B., and Robles-Kelly, A., editors, *Proceedings of SSPR'2018*, pages 460–469. IAPR, Springer International Publishing.

Bougleux, S., Brun, L., Carletti, V., Foggia, P., Gaüzère, B., and Vento, M. (2017). Graph edit distance as a quadratic assignment problem. *Pattern Recognition Letters*, 87:38–46. Impact factor : 1.586.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2014). Spectral networks and locally connected networks on graphs. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Carletti, V., Gaüzère, B., Brun, L., and Vento, M. (2015). *Graph-Based Representations in Pattern Recognition: 10th IAPR-TC-15 International Workshop, GbRPR 2015, Beijing, China, May 13-15, 2015. Proceedings*, chapter Approximate Graph Edit Distance Computation Combining Bipartite Matching and Exact Neighborhood Substructure Distance, pages 188–197. Springer International Publishing, Cham.

Daller, E., Bougleux, S., Gaüzère, B., and Brun, L. (2018). Approximate graph edit distance by several local searches in parallel. In Fred, A., editor, *7th Internation Conference on Pattern Recognition Applications and Methods*.

Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*. PhD thesis, University of Tours, tours, France.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 3844–3852. Curran Associates, Inc.

Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

Gaüzère, B., Bougleux, S., Riesen, K., and Brun, L. (2014). *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, S+SSPR 2014, Joensuu, Finland, August 20-22, 2014. Proceedings*, chapter Approximate Graph Edit Distance Guided by Bipartite Matching of Bags of Walks, pages 73–82. Springer Berlin Heidelberg, Berlin, Heidelberg.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1263–1272.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Jain, B. J. (2014). Margin perceptrons for graphs. In *22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014*, pages 3851–3856.

Jain, B. J. (2016). Statistical graph space analysis. *Pattern Recognition*, 60:802 – 812.

Jain, B. J. and Wysotzki, F. (2004). Central clustering of attributed graphs. *Machine Learning*, 56(1-3):169–207.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Lerouge, J., Abu-Aisheh, Z., Raveaux, R., Héroux, P., and Adam, S. (2017). New binary linear programming formulation to compute the graph edit distance. *Pattern Recognition*, 72:254 – 265.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. S. (2016). Gated graph sequence neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Liu, Z. and Qiao, H. (2014). GNCCP - graduated nonconvexityand concavity procedure. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(6):1258–1267.

Loosli, G., Canu, S., and Ong, C. S. (2016). Learning SVM in kreĭn spaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(6):1204–1216.

Massa, V. D., Monfardini, G., Sarti, L., Scarselli, F., Maggini, M., and Gori, M. (2006). A comparison between recursive neural networks and graph neural networks. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2006, part of the IEEE World Congress on Computational Intelligence, WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006*, pages 778–785.

Peng, N., Poon, H., Quirk, C., Toutanova, K., and Yih, W. (2017). Cross-sentence n-ary relation extraction with graph lstms. *TACL*, 5:101–115.

Rakotomamonjy, A., Bach, F. R., Canu, S., and Grandvalet, Y. (2007). More efficiency in multiple kernel learning. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 775–782.

Riesen, K. and Bunke, H. (2009). Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(7):950 – 959. 7th IAPR-TC15 Workshop on Graph-based Representations (GbR 2007).

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80.

Serratosa, F. (2014). Fast computation of bipartite graph matching. *Pattern Recognition Letters*, 45:244–250.

Serratosa, F. (2015). Speeding up fast bipartite graph matching through a new cost matrix. *Int. Journal of Pattern Recognition*, 29(2).

Simonovsky, M. and Komodakis, N. (2017). Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 29–38.

Sonnenburg, S., Rätsch, G., Schäfer, C., and Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565.

Verma, N., Boyer, E., and Verbeek, J. (2017). Dynamic filters in graph convolutional networks. *CoRR*, abs/1706.05206.

Zayats, V. and Ostendorf, M. (2018). Conversation modeling on reddit using a graph-structured LSTM. *TACL*, 6:121–132.