Graphs and Numerical Spaces

Luc Brun

GREYC – CNRS UMR 6072, University of Caen, ENSICAEN

Luc.Brun@ensicaen.fr

Structural Pattern Recognition

- ☺ Rich description of objects
- ☹ Poor properties of graph's space does not allow to readily generalize/combine sets of graphs

Statistical Pattern Recognition

- ☹ Global description of objects
- ☺ Numerical spaces with many mathematical properties (metric, vector space, ...).

Motivation

Analyse large famillies of structural and numerical objects using a **unified** framework based on pairwise similarity.

Outline

Explicit embedding

From points to Euclidean Distance matrix

- Let us consider a set of points $\{x_1, \dots, x_n\}$ in $\mathbb{R}^n$ and the $n \times n$ matrix $D = (d_{ij})$ defined by:

$$\begin{array}{rcl} d_{ij} = \|x_i - x_j\|^2 & = & < x_i - x_j, x_i - x_j > \\ & = & < x_i, x_i > + < x_j, x_j > -2 < x_i, x_j > \end{array}$$

- The diagonal of D is 0.
- Let S denote the $n \times n$ matrix with $S_{ij} =< x_i, x_j >$.
- We have :

$$D_{ij} = S_{ii} + S_{jj} - 2S_{ij}$$

- an S is definite positive ($\forall c \in \mathbb{R}^n \; c^t S c \geq 0$). Indeed:

$$\forall c \in \mathbb{R}^n \left\{ \begin{array}{rcl} c^t S c & = & \sum_{i=1}^n \sum_{j=1}^n c_i < x_i, x_j > c_j \\ & = & \sum_{i=1}^n c_i < x_i, \sum_{j=1}^n c_j x_j > \\ & = & \| \sum_{j=1}^n c_j x_j \|^2 \geq 0 \end{array} \right.$$

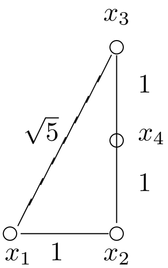
Explicit Embedding

From Matrix to points ?

- We additionally have:
  1. $\sqrt{d_{ij}} \geq 0$ — positivity
  2. $\sqrt{d_{ij}} = 0 \Rightarrow x_i = x_j$ — separation
  3. $\sqrt{d_{ij}} = \sqrt{d_{ji}}$ — symmetry
  4. $\forall k \in \{1, \dots, n\} \ \sqrt{d_{ij}} \leq \sqrt{d_{ik}} + \sqrt{d_{kj}}$ — triangular inequality
- Question:
  For any matrix D fulfilling 1, 3, 4 with a 0 diagonal can we find $\{x_1, \dots, x_n\}$ such that $d_{ij} = \|x_i - x_j\|^2$?
- The answer is: **No**.

$$D = \begin{bmatrix} 0 & 1 & 5 & d_{14} \\ 1 & 0 & 4 & 1 \\ 5 & 4 & 0 & 1 \\ d_{14} & 1 & 1 & 0 \end{bmatrix} \quad \begin{array}{l} \sqrt{d_{23}} = 2 \\ d_{24} = d_{34} = 1 \end{array}$$

Triangular inequality:
$\sqrt{5} - 1 (\approx 1.23) \leq \sqrt{d_{14}} \leq \sqrt{2} (\approx 1.41)$



Explicit Embedding

From Matrix to points: step 1

- Given a set of objects, and a set of distances between these objects, fulfilling all the requirements of a distance we can not ensure that an embedding may be associated to objects.
- Let us first note that the problem is ill posed. Indeed given $d_{ij} = \|x_i - x_j\|^2$, any translation of $(x_i)_{i\in\{1,\dots,n\}}$ would solve the problem equally.
- We should thus fix an origin. Let's choose the barycenter $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.
- We obtain after basic calculus (trust me or compute 🙂)

$$< x_i - \overline{x}, x_j - \overline{x} >= -\frac{1}{2}\left[d_{ij} - \frac{1}{n}\sum_{k=1}^{n} d_{jk} - \frac{1}{n}\sum_{l=1}^{n} d_{jl} + \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n} d_{lk}\right]$$

- This is equivalent to consider the centered matrix:

$$S^c = -\frac{1}{2}(I - ee^t)D(I - ee^t) = -\frac{1}{2}\left[D - \frac{1}{n}ee^tD - \frac{1}{n}Dee^t + \frac{1}{n^2}ee^tDee^t\right]$$

Explicit Embedding

From Matrix to points: step 1

- Given a set of objects, and a set of distances between these objects, fulfilling all the requirements of a distance we can not ensure that an embedding may be associated to objects.
- Let us first note that the problem is ill posed. Indeed given $d_{ij} = \|x_i - x_j\|^2$, any translation of $(x_i)_{i \in \{1,\dots,n\}}$ would solve the problem equally.
- We should thus fix an origin. Let's choose the barycenter $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.
- We obtain after basic calculus (trust me or compute)

$$< x_i - \overline{x}, x_j - \overline{x} >= -\frac{1}{2}\left[d_{ij} - \frac{1}{n}\sum_{k=1}^{n} d_{jk} - \frac{1}{n}\sum_{l=1}^{n} d_{jl} + \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n} d_{lk}\right]$$

- This is equivalent to consider the centered matrix:

$$S^c = -\frac{1}{2}(I - ee^t)D(I - ee^t) = -\frac{1}{2}\left[D - \frac{1}{n}ee^tD - \frac{1}{n}Dee^t + \frac{1}{n^2}ee^tDee^t\right]$$

Explicit Embedding

From Matrix to points: step 1

- Given a set of objects, and a set of distances between these objects, fulfilling all the requirements of a distance we can not ensure that an embedding may be associated to objects.
- Let us first note that the problem is ill posed. Indeed given $d_{ij} = \|x_i - x_j\|^2$, any translation of $(x_i)_{i\in\{1,\dots,n\}}$ would solve the problem equally.
- We should thus fix an origin. Let's choose the barycenter $\overline{x} = \frac{1}{n}\sum_{i=1}^n x_i$.
- We obtain after basic calculus (trust me or compute)

$$< x_i - \overline{x}, x_j - \overline{x} >= -\frac{1}{2}\left[d_{ij} - \frac{1}{n}\sum_{k=1}^n d_{jk} - \frac{1}{n}\sum_{l=1}^n d_{jl} + \frac{1}{n^2}\sum_{k=1}^n\sum_{l=1}^n d_{lk}\right]$$

- This is equivalent to consider the centered matrix:

$$S^c = -\frac{1}{2}(I - ee^t)D(I - ee^t) = -\frac{1}{2}\left[D - \frac{1}{n}ee^tD - \frac{1}{n}Dee^t + \frac{1}{n^2}ee^tDee^t\right]$$

Explicit Embedding

From Matrix to points: step 2

- Let us considered a distance matrix D (e.g. coming from an edit distance) and its centered version $S^c = -\frac{1}{2}(I - ee^t)D(I - ee^t)$.
- Matrix S^c is symmetric. If S^c is semi definite positive. By a basic SVD we obtain $S^c = V\Lambda V^t$. V matrix of eigenvectors, Λ matrix of **positive** eigenvalues.
- Let $X = V\sqrt{\Lambda}$. We have $S^c = XX^t$ and matrix S^c is indeed a matrix of scalar products.
- **Each line of X may be interpreted as the embedding into $\mathbb{R}^n$ of the corresponding object.**
- Pb: No one said to matrix S^c that it has to be semi definite positive. We do so using (for e.g.) the constant shift:

$$\tilde{S}^c = S^c - \lambda_n(S^c)I$$

  where $\lambda_n(S^c)$ is the minimal eigenvalue of S^c.
- Such a transformation modifies the initial metric:

$$\tilde{D} = D - 2\lambda_n(S^c)(ee^t - I)$$

Explicit Embedding

From Matrix to points: step 2

- Let us considered a distance matrix D (e.g. coming from an edit distance) and its centered version $S^c = -\frac{1}{2}(I - ee^t)D(I - ee^t)$.
- Matrix S^c is symmetric. If S^c is semi definite positive. By a basic SVD we obtain $S^c = V\Lambda V^t$. V matrix of eigenvectors, Λ matrix of **positive** eigenvalues.
- Let $X = V\sqrt{\Lambda}$. We have $S^c = XX^t$ and matrix S^c is indeed a matrix of scalar products.
- **Each line of X may be interpreted as the embedding into $\mathbb{R}^n$ of the corresponding object.**
- Pb: No one said to matrix S^c that it has to be semi definite positive. We do so using (for e.g.) the constant shift:

$$\tilde{S^c} = S^c - \lambda_n(S^c)I$$

where $\lambda_n(S^c)$ is the minimal eigenvalue of S^c.
- Such a transformation modifies the initial metric:

$$\tilde{D} = D - 2\lambda_n(S^c)(ee^t - I)$$

Explicit Embedding

Alternatives and Applications

- The above solution corresponds to the minimization of:

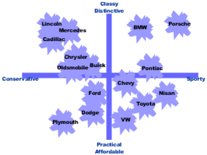
$$\min_X \|D_{ij}^2 - d_{ij}(X)^2\|^2$$

- Alternative optimizations exist such as Smacof minimization:

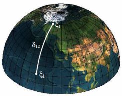
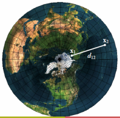
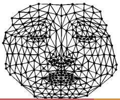
$$\min_X \|D_{ij} - d_{ij}(X)\|^2$$

which unfortunately has no analytical solution $\rightarrow$ solved by iterations.

- Applications:
  - Data visualization (marketing , geography, chemistry,....)



  - Mesh processing (flatening, texture maping,. . .)

Explicit Embedding

Discussion

- 🙂This method allows to associate vectors to structural objects (strings, graphs) such that the distance between vectors is close or equal to the initial distance between objects.
- 🙂Vectors may serve as input of any classification/regression algorithm.
- ☹️ The embedding should be computed on the whole set → forbid the use of a train set and on line classification.
- ☹️ The embedding is relative to each set. Not exactly what we want, we would like an embedding which reflect the properties of our distance independently of the considered set.

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:

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

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

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

Examples

If $\chi = \mathbb{R}^n$, classical kernels include:

- Linear kernel:
$$K(x,y) = x^t y$$
- Polynomial kernel
$$K(x,y) = (x^t y)^d + c,\ c \in \mathbb{R}, d \in \mathbb{N}$$
- Cosinus kernel:
$$K(x,y) = \frac{x^t y}{\|x\|\|y\|}$$
- Rational kernel:
$$K(x,y) = 1 - \frac{\|x-y\|^2}{\|x-y\|^2 + b}, b \in \mathbb{R} - \{0\}$$
- Gaussian Kernel
$$K(x,y) = exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right), \sigma \in \mathbb{R} - \{0\}$$

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 $\varphi : \chi \rightarrow \mathcal{H}$

such that:

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

Reproducing kernel Hilbert Space

- To each d.p. kernel is associated a functional Hilbert space $\mathcal{H}$ called the *Reproducing kernel Hilbert Space.*
- $\mathcal{H}$ is composed of function of the form:

$$f(.) = \sum_{i=1}^{n} \alpha_i k(x_i, .)$$

  $\mathcal{H}$ is composed of functions mapping real values to objects $x \in \chi$.
- For any $f = \sum_{i=1}^{n} \alpha_i k(x_i, .)$ and $g = \sum_{i=1}^{m} \beta_i k(y_i, .)$:

$$< f, g >= \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k(x_i, y_j)$$

- The norm induced by the scalar product on $\mathcal{H}$ is defined as:

$$\|f\|_K^2 = \sum_{i=1}^{n} \sum_{i=1}^{n} \alpha_i \alpha_j k(x_i, x_j)$$

Support Vector Regression/Classification

- Given a set of observations $(x_i, y_i) \in \chi \times \mathbb{R}$, support vector regression/classification methods aim to find $f^* \in \mathcal{H}$ such that:

$$f^* = \underset{f \in \mathcal{H}}{\operatorname{argmin}}\, C.Loss(x_i, y_i, f(x_i)) + \|f\|_K$$

- Where:
  - $Loss()$: is the loss function (attach to data) and
  - $\|f\|_K$: is a regularization term. Encodes the smoothness of f.
  - C is the tradeoff between both terms.
- Kernel design determines how classification/regression algorithms generalize from the training set.

A basic example

- Let $\chi = \mathbb{R}^2$ and $k(x, y) = (x^t y)^2 + 1$
- For any $x = (x_1, x_2)$ and $y = (y_1, y_2)$ we have:

$$\begin{array}{rcl} k(x, y) & = & (x_1 y_1 + x_2 y_2)^2 + 1 \\ & = & 1 + x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 \end{array}$$

- The function φ from $\mathbb{R}^2$ to $\mathbb{R}^4$ defined by:

- Satisfies

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

A basic example

- Let $\chi = \mathbb{R}^2$ and $k(x, y) = (x^t y)^2 + 1$
- For any $x = (x_1, x_2)$ and $y = (y_1, y_2)$ we have:

$$\begin{array}{rcl} k(x, y) & = & (x_1 y_1 + x_2 y_2)^2 + 1 \\ & = & 1 + x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 \end{array}$$

- The function φ from $\mathbb{R}^2$ to $\mathbb{R}^4$ defined by:

$$\varphi(x) = \left(\begin{array}{c} 1 \\ x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \end{array} \right)$$

- Satisfies

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

A basic example

- Let $\chi = \mathbb{R}^2$ and $k(x,y) = (x^t y)^2 + 1$
- For any $x = (x_1, x_2)$ and $y = (y_1, y_2)$ we have:

$$\begin{array}{rcl} k(x,y) & = & (x_1 y_1 + x_2 y_2)^2 + 1 \\ & = & 1 + x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 \end{array}$$

- The function φ from $\mathbb{R}^2$ to $\mathbb{R}^4$ defined by:

$$\varphi(x) = \begin{pmatrix} 1 \\ x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \end{pmatrix}$$

- Satisfies

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

- Remark: An hyperplane in $\mathcal{H} = \mathbb{R}^4$ corresponds to a quadric of $\mathbb{R}^2$.

$$< \varphi(x), n >= K \Rightarrow n_1 + n_2 x_1^2 + n_3 x_2^2 + n_4 \sqrt{2} x_1 x_2 = K$$

The Kernel Trick

- Linear classifier become non-linear using kernels

- Problem: φ is usually unknown.
- Many methods only need scalar product between data (not explicit coordinates) $\Rightarrow$ replace scalar product by kernel.
- E.g. k-NN:

$$
\begin{aligned}
d_K^2(x_1,x_2) &= \|\varphi(x_1)-\varphi(x_2)\|^2 \\
&= <\varphi(x_1)-\varphi(x_2),\varphi(x_1)-\varphi(x_2)> \\
&= <\varphi(x_1),\varphi(x_1)> + <\varphi(x_2),\varphi(x_2)> -2<\varphi(x_1),\varphi(x_2)> \\
d_K(x_1,x_2) &= k(x_1,x_1)+k(x_2,x_2)-2k(x_1,x_2)
\end{aligned}
$$

- Kernel trick
  - Algorithm defined in $\mathcal{H} \Rightarrow$ (linear methods,non linear separation),
  - Data stored in χ.

Interesting but so what...

Outline

Kernel and strutured data

The kernel trick provides an implicit embedding whose metric is defined from our similarity criterion (the kernel).

Do we absolutely need definite positiveness

- **No.**([Haasdonk, 2005]).
- results of SVM may be interpreted even with a non definite positive kernel (distance to convex hull).
- Several methods have been specifically designed to deal with non definite positive kernels.
- **But** ,
  - Results are usually more difficult to interpret (Krein space),
  - Mathematical properties are usually weaker,

Global alignment kernel

Recall

- Let $\mathcal{A}(n,m)$ denote all locals alignments from string of lenght n to strings of length m.
- The Dynamic Time warping (DTW) between x and y is defined as:

$$DTW(x,y) = \min_{\pi \in \mathcal{A}(n,m)} D_{x,y}(\pi)$$

with

$$D_{x,y}(\pi) = \sum_{i=1}^{|\pi|} \varphi(x_{\pi_1(i)}, y_{\pi_2(i)})$$

- φ any difference operator.
- Example: $x = restauration$, $y = restaurant$:

	1	2	3	4	5	6	7	8	9	10
π_1 :	1	2	3	4	5	6	7	8	12	13
	r	e	s	t	a	u	r	a	n	$-$
π_2 :	1	2	3	4	5	6	7	8	9	10
	r	e	s	t	a	u	r	a	n	t

Global alignment kernel

Design (Cuturi 2011)

- $DTW(x,y)$ is not a valid distance function (due to the min operator). We thus define our kernel as follows:

$$k_{GA}(x,y) = \sum_{\pi \in \mathcal{A}(n,m)} e^{-D_{x,y}(\pi)}$$

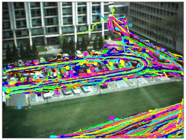
- which is equivalent to:

$$k_{GA}(x,y) = \sum_{\pi \in \mathcal{A}(n,m)} \Pi_{i=1}^{|\pi|} \kappa(x_{\pi_1}(i), y_{\pi_2}(i)).$$

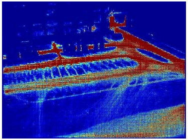
- Few results:
  - k_{GA} is d.p. iff κ and $\frac{\kappa}{1+\kappa}$ are d.p.
  - k_{GA} may be computed with the same computational scheme than the e.d. with $M_{0,0} = 1, M_{0,i} = M_{j,0} = 0$ and

$$M_{ij} = \kappa(x_i, y_j)\left(M_{i-1,j-1} + M_{i,j-1} + M_{i-1,j}\right)$$

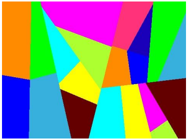
Global Alignment kernel



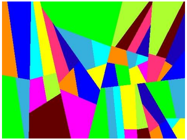
Trajectories



Heat Map



Clustering K=20



Clustering K=50

We pass from:

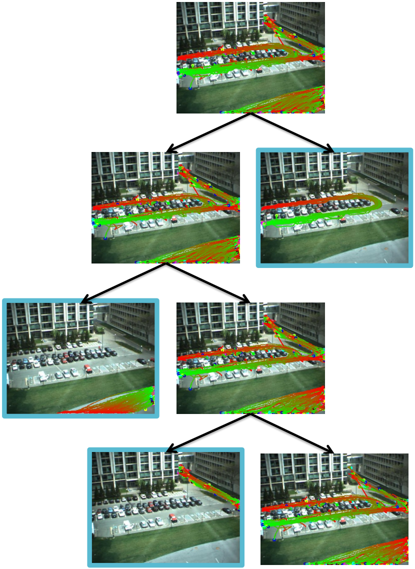
$$t = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \dots \begin{pmatrix} x_n \\ y_n \end{pmatrix}$$

To:

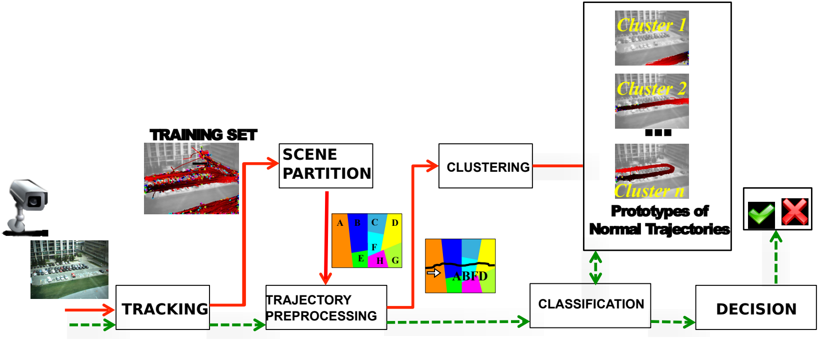
$$t = z_1 \dots z_p \text{ with } p << n$$

Global Alignment kernel

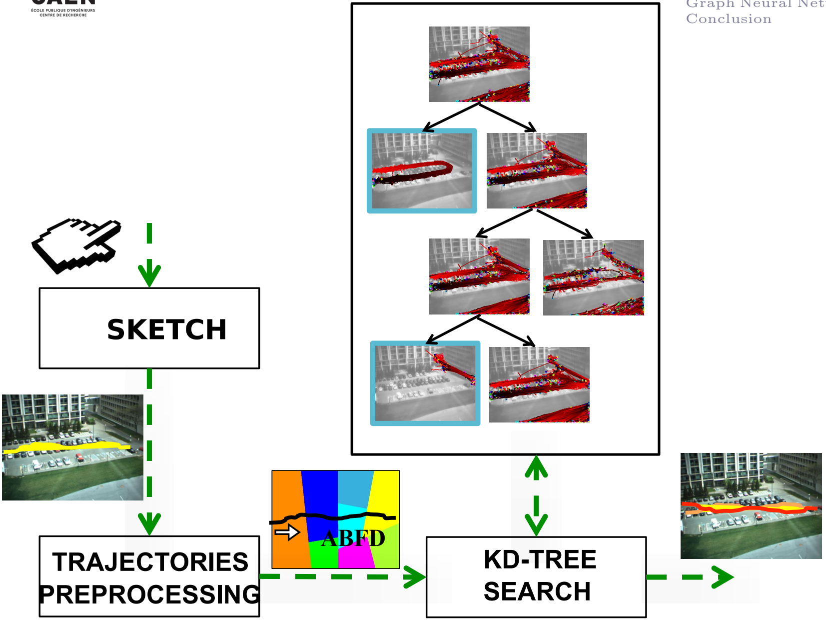
Clustering of trajectories



Detection of Abnormal trajectories



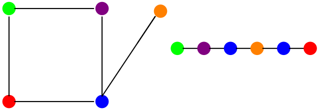
Retreival of most similar trajectories



Walk kernels

Basement paper: [Haussler, 1999].

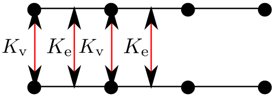
- Walks: Let $G = (V, E)$. $W = (v_1, \ldots, v_n)$ is a walk iff $(v_i, v_{i+1}) \in E, \forall i \in \{1, \ldots, n-1\}$.



is a walk

- Kernel between walks

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



Walks kernels

- Walk kernels [Kashima et al., 2003] :

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

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

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

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

- Walks may induce totering problems: Walks with arbitrary length on the same set of edges and vertices.
- Framework extended to tree-pattern [Mahé and Vert, 2009].

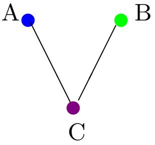
Walk kernel

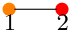
- Kronecker product
  Given two real matrices $A^{n\times m}$ and $B^{p\times q}$, the Kronecker product of A and B ($A \otimes B$) belongs to $\mathbb{R}^{np\times mq}$ and is defined as:

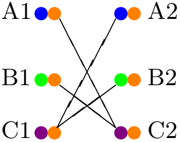
$$A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \dots & a_{1,m}B \\ \vdots & \vdots & & \vdots \\ a_{n,1}B & a_{n,2}B & \dots & a_{n,m}B \end{bmatrix}$$

- Tensor product graph Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the tensor product graph $G = G_1 \otimes G_2 = (V, E)$ is defined by:

$$\begin{array}{rcl} V & = & \{(v_1, v_2) \in V_1 \times V_2\} \\ E & = & \{((v_1, v_2), (v_1', v_2')) \mid (v_1, v_1') \in E_1 \text{ and } (v_2, v_2') \in E_2\} \end{array}$$







Walk kernel

- Any walk in $G_1 \otimes G_2$ corresponds to a common walk in G_1 and G_2 and vice versa.
- If M_1 is the adjacency matrix of G_1 and M_2 the one of G_2, $M_1 \otimes M_2$ is the adjacency matrix of $G_1 \otimes G_2$.
- $\left((M_1 \otimes M_2)^k\right)_{i,j}$ encodes the number of common walks of length k between nodes $i = (v_1, v_2)$ and $j = (v_1', v_2)$.
- Matrices:

$$(I - \lambda M_1 \otimes M_2)^{-1} = \sum_{k=0}^{+\infty} \lambda^k (M_1 \otimes M_2)^k \text{ or } exp(\lambda M_1 \otimes M_2) = \sum_{k=0}^{+\infty} \frac{\lambda^k}{k!} (M_1 \otimes M_2)^k$$

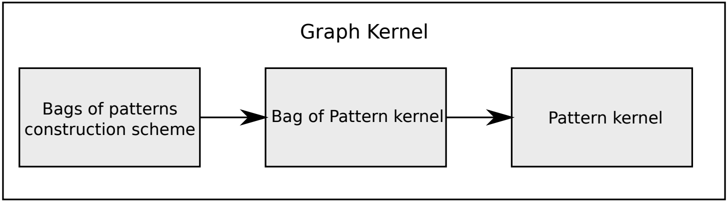
  encode the numbers of all common walks of G_1 and G_2 weighted by a factor depending of their length (λ^k or $\frac{\lambda^k}{k!}$).
- Kernel (e.g.):

$$K(G_1, G_2) = \sum_{i=1}^{np} \sum_{j=1}^{mq} exp(\lambda M_1 \otimes M_2)_{i,j}$$

Finite Bag kernels

$$\left.\begin{array}{ccc} G & \rightarrow & B(G) \\ G' & \rightarrow & B(G') \end{array}\right\} K(G,G') = K(B(G),B(G'))$$

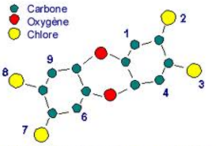
Three independent step to design a graph kernel.



Graph kernels and chemoinformatics

- Aims : Predict the physical or biological properties of a molécule using the similarity principle :

  Two structurally similar molecules should have similar properties.



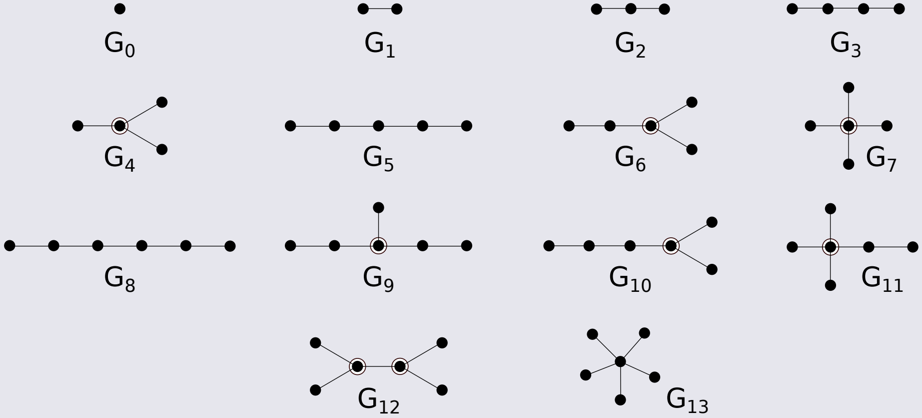
2,3,7,8-tétrachlorodibenzo-*p*-dioxine

- Graph definition : $G = (V, E, \mu, \nu)$
  - μ encodes atom's type,
  - ν encodes types of bonds.

Bag of patterns : unlabeled treelets

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

Treelets



Treelet code

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

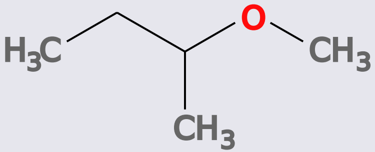


Figure: Code : G09-C1C1C1O1C1C

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

Bag of labeled treelets

Treelet code

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

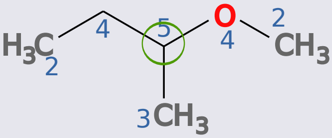


Figure: Code : G09-C1C1C1O1C1C

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

Bag of labeled treelets

Treelet code

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

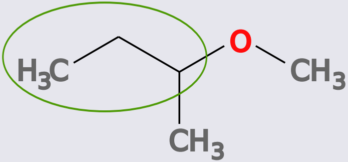


Figure: Code : G09-C1C1C1O1C1C

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

Bag of labeled treelets

Treelet code

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

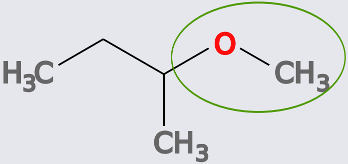


Figure: Code : G09-C1C1C1O1C1C

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

Bag of labeled treelets

Treelet code

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

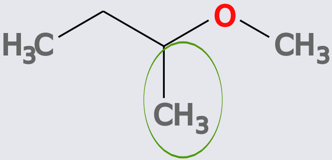
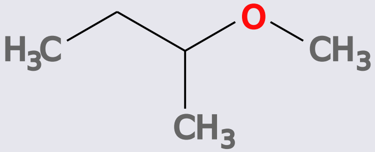


Figure: Code : C1C1C1O1C1C

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

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-C-O(1) C		
$\vdots$		

(b) Histogram

Kernel definition

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

where $K(.,.)$ is any kernel between real numbers (e.g. $K(x,y) = e^{-\frac{(x-y)^2}{\sigma^2}}$).

Lets go back on bag definition

- Some Facts :
  - Our first solution considers bags containing all patterns,
  - We do not have as in shape recognition a measure a priori of the relevance of a pattern,
  - The relevance of a given treelet should be fixed a posteriori given a dataset.
- MKL (Multiple Kernel Learning) method :

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

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

- Our kernel :

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

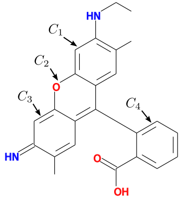
- A direct application of the MKL method provides the new kernel:

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

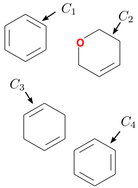
where w_τ is defined using MKL.

Cycle information

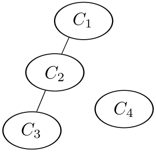
- Given $G = (V, E)$, define a graph $C = (V_C, E_C)$ such that:
  - $c \in V_C$ encodes a cycle of G.
  - $e \in E_C$ encodes an adjacency relationship between two cycles.
- Apply our treelet kernel on C and combine it with the one on G.



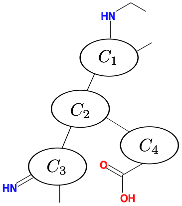
Molecular graph.



Relevant cycles(RC).



RC graph.



RC hypergraph.

Some results

Method	RMSE		Familly
Gaussian edit distance	10.27		Graph edit distance
Graph Embedding (Bunke)	10.19		Graph edit distance
Graph Embedding (EDM)	12.2		Graph edit distance
Kmean	12.24		Finite bag of paths
Random Walks	18.72		Infinite bag of walks
Tree Pattern Kernel	11.02		Infinite bag of tree patterns
Treelet Kernel (TK)	8.10		Finite bag of tree patterns
TK + Forward Selection	7.05		
TK + Backward Elimination	6.75		
Inter treelet kernel	5.89		
TK + MKL	**5.24**		

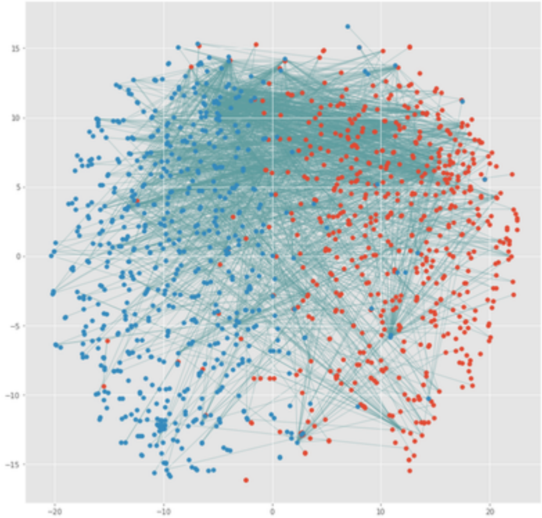
Table: Boiling point prediction on acyclic molecule dataset using 90% of the dataset as train set and remaining 10% as test set.

Some results

Method	RMSE	Times(s)	Familly
Gaussian edit distance	10.27	1.35	Graph edit distance
Graph Embedding (Bunke)	10.19		Graph edit distance
Graph Embedding (EDM)	12.2	7.21	Graph edit distance
Kmean	12.24		Finite bag of paths
Random Walks	18.72	19.10	Infinite bag of walks
Tree Pattern Kernel	11.02	4.98	Infinite bag of tree patterns
Treelet Kernel (TK)	8.10	**0.07**	Finite bag of tree patterns
TK + Forward Selection	7.05		
TK + Backward Elimination	6.75		
Inter treelet kernel	5.89		
TK + MKL	**5.24**		

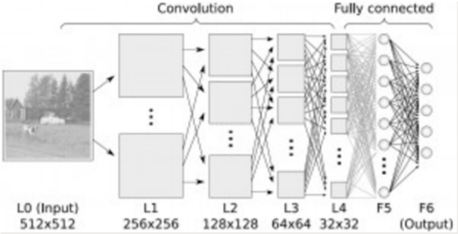
Table: Boiling point prediction on acyclic molecule dataset using 90% of the dataset as train set and remaining 10% as test set.

Graph Neural Network

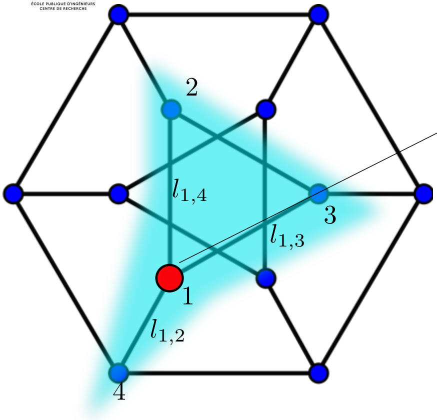


Graph Neural Networks: Three main steps

1. Agregation,
2. Decimation,
3. Pooling



Agregation: The problem

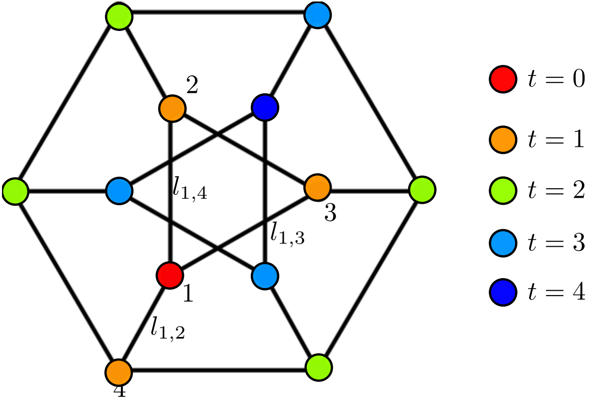


$$h_1 = f_w \left(l_1, \{l_{1,2}, l_{1,3}, l_{1,4}\}, \{h_2, h_3, h_4\}, \{l_2, l_3, l_4\} \right)$$

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

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

Iteration



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

Graph Convolution

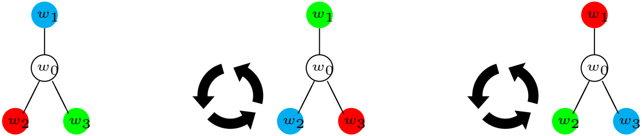
The problem

- 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.

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]

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

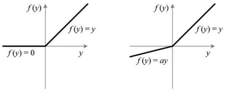
z_v^t: update gate, r_v^t: reset gate, A_v: weight by edges types.

- 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 [Velickovic et al., 2018]

- Not all neighbors have a same importance for update:

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



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

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

- 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

Input image

Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map

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:
$$z * x = U(\hat{z} \odot \hat{x}) = U\left(U^T z \odot U^T x\right) = U\left(diag(U^T z)U^T x\right)$$
$\odot$: Hadamard product.
- Let $g_\theta(\Lambda)$ be a diagonal matrix. The filtering of x by g_θ is:
$$y = U\left(g_\theta(\Lambda)U^T x\right) = \left(U g_\theta(\Lambda)U^T\right)x$$

Graph convolution

- 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,
  - $L^2 x$: 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

- 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

- [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

Conclusion

Recurent networks

[Hochreiter and Schmidhuber, 1997]

[Massa et al., 2006]

[Scarselli et al., 2009]

[Li et al., 2016]

[Gilmer et al., 2017]

[Peng et al., 2017]

[Zayats and Ostendorf, 2018]

Convolution

[Bruna et al., 2014]

[Defferrard et al., 2016]

[Kipf and Welling, 2017]

[Simonovsky and Komodakis, 2017]

[Verma et al., 2017]

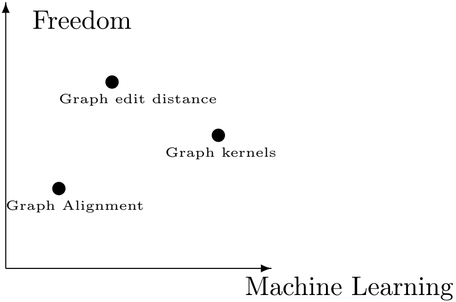
Agregation

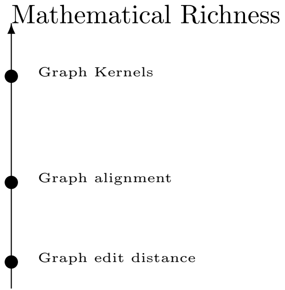
What's next ?

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

Conclusion

- Three graph metrics





- Graph neural network:
  - Still in their infancy,
  - A great potential.

Bibliography

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.*

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3837–3845.

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.

Haasdonk, B. (2005). Feature space interpretation of svms with indefinite kernels. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(4):482–492.

Haussler, D. (1999). Convolution kernels on discrete structures ucsc crl. Technical Report UCSC-CRL-99-10, Department of Computer Science, University of California at Santa Cruz, Santa Cruz, CA 95064.

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

Kashima, H., Tsuda, K., and Inokuchi, A. (2003). Marginalized kernels between labeled graphs. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 321–328.

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.*

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.*

Mahé, P. and Vert, J.-P. (2009). Graph kernels based on tree patterns for molecules. *Machine Learning*, 75(1):3–35.

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.

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.

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.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

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

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. (2010). Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242.

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