HAL
open science

Graph Kernels in chemoinformatics

Benoit Gaüzère, Luc Brun, Didier Villemin

To cite this version:

Benoit Gaüzère, Luc Brun, Didier Villemin. Graph Kernels in chemoinformatics. Matthias Dehmer and Frank Emmert-Streib. Quantitative Graph Theory: Mathematical Foundations and Applications, Taylor & Francis Group, 2014, 978-1-4665-8451-8. hal-01921549

HAL Id: hal-01921549

https://hal.science/hal-01921549v1

Submitted on 5 Dec 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Graph Chapter on Graph kernels in chemoinformatics

Benoit Gaüzère, Luc Brun and Didier Villemin

December 5, 2018

Chapter 1

Graph Kernels in chemoinformatics

1.1 Introduction

Graphs provide a generic data structure widely used in bioinformatics to represent complex structures such as chemical compounds or complex interactions between proteins. Using this kind of representation, property predictions based on similarities are not trivial since many machine learning methods are defined on vectorial spaces and can not be directly applied to graphs.

Algorithms restricted to the graph domain are thus essentially restricted to k-nearest-neighbours or k-medians algorithms. These algorithms use different measures of similarity based on the set of frequent sub graphs extracted from graphs or from the size of the maximum common sub graph of two graphs. A widely used graph similarity measure is based on a measure of the distortion required to transform one graph into another. This measure of dissimilarity, called graph edit distance, is NP-hard to compute but this complexity can be reduced by computing a sub optimal, but usually effective, graph edit distance. Nevertheless, graph edit distance do not usually fit all the requirements of an Euclidean distance and hence can not be readily applied in conjunction with many machine learning methods.

Graph embedding methods aims to tackle this limitation by embedding graphs into explicit vectorial representations which allows the use of any machine learning algorithm defined on vectorial representations. However, encoding graphs as explicit vectors having a limited size induces a loss of information which may reduce prediction accuracy.

Graph kernels are defined as similarity measures between graphs. Under mild conditions, graph kernels correspond to scalar products between possibly implicit graph embeddings into an Hilbert space. Thanks to this graph embedding, machine learning methods which may be rewritten so as to use only scalar products between input data, such as SVM, can be applied on graphs. Graph

kernels thus provide a natural connection between graph space and machine learning.

A large familly of kernel methods is based on a decomposition of graphs into bags of patterns. The prototypical example of this last familly, is the complete graph kernel which is based on a decomposition of a graph into all its sub structures (sub graphs). Such a kernel being NP-Hard to compute, kernels based on bags of patterns must be restricted to the extraction of different familly of sub structures. Given a particular type of bag of pattern, each sub structure of a bag encodes a different structural information. This information may be weighted according to the property to predict. Such a weighting scheme allows to highlight relevant patterns having a high influence on a particular property.

1.1.1 General definitions

This first section aims to introduce some basics of graph theory required to define graph similarity measures used in chemoinformatics.

Definition 1 (Graph) *A non oriented unlabeled graph is a pair $G = (V, E)$ such that V corresponds to a set of nodes and $E \subset V \times V$ corresponds to a set of edges connecting nodes. The size of the graph is defined by $|V|$. If $(u, v) \in E$, u is said to be adjacent to v.*

Definition 2 (Neighbourhood) *Neighbourhood relationship is encoded by the function $\Gamma : V \to \mathcal{P}(V)$ with $\Gamma(v) = \{u \in V \mid (u, v) \in E\}$ where $\mathcal{P}(V)$ denotes all subsets of V.*

Definition 3 (Degree) *The degree of a node $v \in V$ is defined as $|\Gamma(v)|$.*

Definition 4 (Non oriented graph) *A graph is non oriented if for any pair of nodes $(u, v) \in E$, $(v, u) \in E$. In this case (u, v) denotes indifferently the oriented edges (u, v) and (v, u).*

Definition 5 (Sub graph) *A graph $G' = (V', E')$ is a sub graph of $G = (V, E)$, denoted $G' \sqsubseteq G$, if $V', \subseteq V$ and $E' \subseteq E$.*

Definition 6 (Walks, Trails, Paths) *A walk of a graph $G = (V, E)$ is a sequence of nodes $W = v_1 \dots v_n$ connected by edges: $(v_i, v_{i+1}) \in E$ for any $i \in \{1, \dots, n-1\}$. If each edge (v_i, v_{i+1}) appears only once in W, W is called a trail. If each vertex (and thus each edge) appears only once, W is called a path. The length of a walk is defined*

Definition 7 (Distance between nodes) *The distance $d_G(u, v)$ between two nodes u and v of a graph $G = (V, E)$ is defined as the length of the shortest path between u and v in G.*

Definition 8 (Cycle) *A cycle is a path whose first node is equal to the last one. This node is the only one appearing twice in the sequence.*

Definition 9 (Connected graph) *A graph $G = (V, E)$ is connected if it exists a path between any pair of distinct nodes $(u, v) \in V^2$. A maximal connected subset of V is called a connected component of G.*

Definition 10 (Tree) *A tree is a connected graph without cycles.*

Definition 11 (Bridge) *A bridge in an edge whose removal increases the number of connected component of the graph. The set of bridges of a graph G is denoted $\mathcal{B}(G)$.*

Definition 12 (Labeled graph) *A non oriented labeled graph $G = (V, E, \mu, \nu)$ is a non oriented unlabeled graph $G = (V, E)$ associated to a node labeling function $\mu : V \to L_V$ and an edge labeling function $\nu : E \to L_E$, where L_V and L_E denote respectively sets of node and edge labels.*

Two graphs are considered as equal if it exists a bijection between the nodes of both graphs which respect adjacency relationships. In this case both graphs are said to be isomorphic.

Definition 13 (Graph isomorphism) *Two graphs $G = (V, E)$ and $G' = (V', E')$ are structurally isomorphic, denoted $G \simeq_s G'$, if and only if it exists a bijection*

$$f : V \to V'$$

such that:

$$(u, v) \in E \Leftrightarrow (f(u), f(v)) \in E'.$$

If G and G' correspond to labeled graphs, i.e. if $G = (V, E, \mu, \nu)$ and $G' = (V', E', \mu', \nu')$. A structural isomorphism between G and G' is called a graph isomorphism, denoted by $G \simeq G'$, if

$$\begin{array}{lll} \forall v \in V & \mu(v) & = \mu'(f(v)) \quad \textit{and} \\ \forall (u, v) \in E & \nu(u, v) & = \nu'(f(u), f(v)). \end{array}$$

Definition 14 (Partial sub graph isomorphism) *Let $G = (V, E, \mu, \nu)$ and $G' = (V', E', \mu', \nu')$ denote two graphs such that $|V| \leq |V'|$. It exists a partial structural sub graph isomorphism between G and G' if and only if it exists an injection:*

$$f : V \to V'$$

such that:

$$\forall (u, v) \in V^2 \ (u, v) \in E \Rightarrow (f(u), f(v)) \in E'.$$

A partial structural sub graph isomorphism is a partial sub graph isomorphism, denoted $G \subseteq_p G'$, if:

$$\begin{array}{lll} \forall v \in V & \mu(v) & = \mu'(f(v)) \quad \textit{and} \\ \forall (u, v) \in E & \nu(u, v) & = \nu'(f(u), f(v)). \end{array}$$

Note that the partial sub graph isomorphism relationship may also be denoted an homomorphism relationship in other references.

Definition 15 (sub graph isomorphism) *Let $G = (V, E, \mu, \nu)$ and $G' = (V', E', \mu', \nu')$ two graphs such that $|V| \leq |V'|$. It exists a structural sub graph isomorphism if it exists a partial structural sub graph isomorphism between G and G' and:*

$$\forall (u, v) \in V'^2, (u, v) \in E' \Leftrightarrow (u, v) \in E$$

A structural sub graph isomorphism is a sub graph isomorphism denoted $G \subseteqq G'$, if:

$$\begin{array}{lll} \forall v \in V & \mu(v) & = \mu'(f(v)) \quad \textit{and} \\ \forall (u, v) \in E & \nu(u, v) & = \nu'(f(u), f(v)). \end{array}$$

Hypergraphs correspond to an extension of graphs which allows to define an adjacency relationship between more than two nodes. Hypergraphs have been introduced by Claude Berge [3].

Definition 16 (Hypergraph) *An hypergraph $H = (V, E)$ is a pair of sets V, encoding hypergraph nodes, and $E = (e_i)_{i \in I} \subseteq \mathcal{P}(V)$ encoding hyperedges:*

- $\forall i \in \{1, \ldots, |E|\}, e_i \neq \emptyset$,
- $\cup_{i \in I} \; e_i = V$

As for graphs, the size of an hypergraph $H = (V, E)$ is defined as $|V|$ and two nodes u and v are adjacent if it exists $e \in E$ such that $\{u, v\} \subset e$.

Oriented hypergraph [11] is an hypergraph where hyperedges connect two sets of nodes (figure 1.1):

Definition 17 (Oriented hypergraph) *An oriented hypergraph $H = (V, \vec{E})$ is a pair of sets V, encoding hypergraph nodes, and $\vec{E} = (e_i)_{i \in I} \subseteq \mathcal{P}(V) \times \mathcal{P}(V)$ encoding oriented hyperedges. An oriented hyperedge $e = (s_u, s_v) \in \vec{E}$ with $s_u = \{u_1, \ldots, u_i\} \subset \mathcal{P}(V)$ and $s_v = \{v_1, \ldots, v_j\} \subset \mathcal{P}(V)$ encodes an adjacency relationship between set of nodes s_u et s_v (figure 1.1).*

In the remaining part of this chapter we will consider that an hyperedge (s_v, s_u) exists in $\vec{E}$ for each $(s_u, s_v) \in \vec{E}$ and will thus not differentiate (s_u, s_v) from (s_v, s_u). Note anyway that an oriented hypergraph remains different from an usual hypergraph since within an oriented hypergraph an edge connects two sets of vertices while within the usual hypergraph framework an edge is composed of a single set.

An usual molecular representation is defined by the molecular graph [14, 45] (figure 1.2). A molecular graph is a non oriented labeled graph encoding adjacency relationships between atoms of a molecule. The set of nodes encodes the set of atoms of a molecula while the set of edges encodes atomic bonds connecting these atoms. Each vertex is labeled by the chemical element of the corresponding atom and each edge is labeled by the type of the atomic bond (simple, double, triple or aromatic) connecting two atoms. An usual convention consists in not encoding hydrogen atoms within molecular graph since they are implicitly encoded by the valency of other atoms. An other used convention is to not explicitly represent carbon atoms within graphical representation of a molecular graph (figure 1.2).

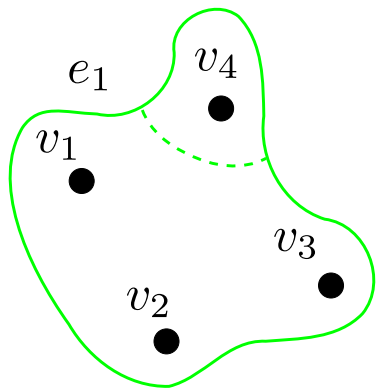


Figure 1.1: Oriented hypergraph with oriented hyperedge $e_1 = (\{v_1, v_2\}, \{v_3\})$.

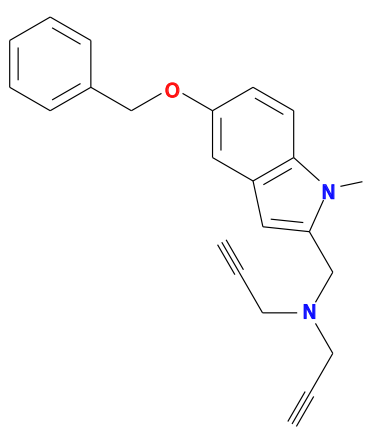


Figure 1.2: A molecular graph

1.2 Graph similarity measures

Most of existing methods used in chemoinformatics are based on the similarity principle which states that similar compounds have similar properties. If we consider the molecular representation given by the molecular graph, comparing molecular compounds thus relies on comparing molecular graphs. Therefore, in order to be able to predict physical or biological properties of molecular compounds, we have to define molecular graph similarity measures.

1.2.1 Maximum common subgraphs

A first similarity measure between graphs based on graph theory is defined from the maximum common subgraph. Intuitively, two graphs are considered as similar if they share a large common structure and dissimilar otherwise. In order to define the notion of maximum common subgraph, let us first define the notion of common subgraph:

Definition 18 (Common subgraph) *A graph G is a common subgraph of two graphs G_1 et G_2 if it exists two graphs $\hat{G}_1$ and $\hat{G}_2$ of G_1 and G_2 such that:*

$$G \simeq \hat{G}_1 \simeq \hat{G}_2$$

Common sub graphs correspond thus to sets of nodes and edges common to two graphs. In order to formally define maximum common subgraphs, let us additionally define the notion of maximal common subgraph:

Definition 19 (Maximal common subgraph) *A common sub graph of two graphs G_1 and G_2 is maximal if it is a common subgraph of G_1 and G_2 and if it is not itself a subgraph of another common subgraph of G_1 et G_2.*

The notion of maximal common subgraph corresponds to set of nodes and edges common to two graphs which can not be enlarged. Using maximal common subgraphs, we can now define maximum common subgraph:

Definition 20 (Maximum common subgraph) *A common subgraph G is maximum if it is a maximal common subgraph of G_1 and G_2 and it does not exist a larger common subgraph of G_1 and G_2.*

A maximum common subgraph corresponds thus to the largest maximal common subgraph.

The size of a maximum common subgraph of two graphs may be seen as a similarity measure between these two graphs. Methods based on maximum common subgraph consider two graphs as similar if they share a large common structure. Conversely, graphs sharing a lot of small common structures but not a large one are not considered as similar. This particularity may alter the accuracy of methods based on maximum common subgraphs on chemical prediction problems where molecular activity may be due to a large number of small structures.

1.2.2 Frequent graphs

A widely used approach in chemoinformatics consists in finding sub structures which are responsible of a particular activity. This family of methods is thus more dedicated to activity prediction problems since from a chemical point of view, a molecular compound may have a particular property if it contains a sub structure, called a pharmacophore. This chemical consideration has sustained the emergence of a familly of methods based on the discovery of frequent sub graphs [9, 52, 34]. These methods are based on the following assumptions:

- A sub structure is considered as frequent if its number of occurences is greater than a frequency threshold σ within a set of positive molecular compounds and is insignificant within a set of negative molecular compounds;
- A substructure of size k may be frequent if it is composed of at least two frequent substructures of size k-1

Among the set of methods using such an approach, methods described in [9, 52] are based on an iterative algorithm (algorithm 1) which aims to find frequent sub graphs of size k from the set of frequent subgraphs of size $k-1$.

Algorithm 1: Generic algorithm used by frequent subgraphs methods

```
Find a set of subgraphs S_2 having a size k = 2 and a frequency greater
  than σ. ;
for k = 3 → k-max do
    Find a set of candidates C_k of size k from S_{k-1}. ;
    S_k ← frequent subgraphs in C_k;
```

These approaches obtain a good prediction accuracy on many datasets within a reasonable computational time. However, the main drawback of these approaches is the frequent subgraphs hypothesis. Indeed, the hypothesis that one pharmacophore is responsible of an activity is no longer valid when the activity of molecular compounds is induced by a set of differents pharmacophores. In this case, each pharmacophore is associated to few molecular compounds and each pharmacophore does not reach the minimal frequency threshold in order to be considered as a frequent sub structure. Such configurations induce thus a poor prediction model.

1.2.3 Graph edit distance

Graph edit distance aims to define a distance between graphs. This distance is based on a measure of distortion induced by a transformation of one graph into another. Graph edit distance between two graphs is defined by the minimal cost associated to an edit path which transform the first graph into the second one. This distance is defined as a sequence of edit operations, each of these elementary operations being defined as:

- a node/edge insertion: Adding one node/edge to the graph;
- a node/edge deletion: Removing one node/edge to the graph;
- a node/edge substitution: Replace one node/edge label.

Each of these edit operations is associated to a cost $c(.) \in \mathbb{R}^+$. Considering these edit costs, graph edit distance is defined as:

$$d_{edit}(G, G') = \min_{(e_1,\ldots,e_k)\in\mathcal{C}_h(G,G')} \sum_{i=1}^{k} c(e_i) \tag{1.1}$$

where $\mathcal{C}_h(G, G')$ encodes all edit paths transforming G into G'. Each edit path encodes a sequence of edit operations $e_1, \ldots, e_k$, each operation e_i being associated to an edit cost $c(e_i)$.

Graph edit distance may be computed using A^* algorithm [19]. This algorithm consists in building a rooted tree where each path from root to a leaf encodes a possible edit path. Then, the root to leaf path corresponding to the lowest sum of edit operation costs is defined as the optimal edit path. However, the computational complexity induced by this method is exponential according to the number of graphs' nodes which limits its application to very small graphs.

Graph edit distance approximation

In order to reduce the complexity required by the computation of graph edit distance, Riesen proposed a polynomial algorithm [38] which reduces graph edit distance complexity. Given two graphs $G = (V, E)$ of size $n = |V|$ and $G' = (V', E')$ of size $m = |V'|$, this algorithm is based on a complete bipartite graph $G_a = (V_a, E_a)$ where $V_a = V_G \cup V_{G'}$, $V_G = \{V \cup \{\varepsilon_1, \ldots, \varepsilon_m\}\}$, $V_{G'} = \{V' \cup \{\varepsilon'_1, \ldots, \varepsilon'_n\}\}$ and $E_a = \{(u, v) \mid u \in V_G \text{ et } v \in V_{G'}, \forall (u, v) \in V_G \times V_{G'}\}$. Bipartite graph G_a encodes a node to node matching between the two sets V_G and $V_{G'}$. A matching $u \in V \rightarrow v \in V'$ encodes a substitution whereas a matching $u \in V \rightarrow \varepsilon'_i$ (resp. $\varepsilon_i \rightarrow v \in V'$) encodes a node deletion (resp. a node insertion). Each edge $(u, v) \in V_G \times V_{G'}$ is weighted by the cost associated to the edit operation encoded by matching $u \rightarrow v$. Therefore, each matching $\varepsilon_i \rightarrow \varepsilon'_j$ is associated to zero cost since they do not encode any edit operation.

Optimal matching, i.e. matching from V to V' having the lowest sum of edge cost, is computed by means of Munkre's algorithm (or Hungarian algorithm) [32]. This optimal matching is then associated to an edit path, itself being associated to a cost. However, the computed cost does not correspond necessarily to graph edit distance since there is no guaranty that the computed edit path is the optimal one. On the first hand, the main drawback of this algorithm is that the computed dissimilarity measure does not correspond to exact graph edit distance, but only to a sub optimal one. On the other hand, this dissimilarity measure can be computed in polynomial time which allows the use of this approach on chemoinformatics problem.

Graph edit distance and machine learning methods

If we consider that costs associated to elementary edit operations define a distance (proposition 21), graph edit distance fulfills the four properties of a metric and thus defines a distance [33].

Definition 21 (Distance) *A function $d : \mathcal{X}^2 \to \mathbb{R}^+$ is a distance over $\mathcal{X}$ if, for any $(x_i, x_k) \in \mathcal{X}^2$, it fulfills the four following properties:*

- *nonnegativity: $d(x_i, x_j) \geq 0$;*
- *self distance: $d(x_i, x_j) = 0 \Leftrightarrow x_i \simeq x_j$;*
- *symmetry: $d(x_i, x_j) = d(x_j, x_i)$;*
- *triangle inequality: for any $x_k \in \mathcal{X}$, $d(x_i, x_j) \leq d(x_i, x_k) + d(x_k, x_j)$.*

However, graph edit distance does not fulfill the fifth property of an euclidean distance(Definition 22) [8] and hence does not correspond to a distance in an euclidean space. From a practical point of view, this liitation induces that the metric space defined by the edit distance on the set of graphs can not be isometrically embeded (neither explicitly nor implicitly) into an Hilbert space. This last point restrict drastically the set of machine learning algorithm which may be used in conjunction with the graph edit distance.

Definition 22 (Relative angle inequality [8]) *For any $i, j, l \neq k$, $k = \{1, \ldots, N\}$, $i < j < l$. For $N \geq 4$, let consider a set of distinct points $\mathcal{X}_k = \{x_1, \ldots, x_N\}$. Relative angle inequality property is fulfilled if all points $x_k \in \mathcal{X}_k$ fulfill following inequalities:*

$$\begin{gathered} cos(\theta_{ikl} + \theta_{lkj}) \leq cos(\theta_{ikj}) \leq cos(\theta_{ikl} - \theta_{lkj}) \\ 0 \leq \theta_{ikl}, \theta_{lkj}, \theta_{ikj} \leq \pi \end{gathered} \tag{1.2}$$

where $\theta_{ikj} = \theta_{jki}$ encodes angle between vectors i et j at point x_k.

The need of a fifth property to define euclidean distances can be illustrated using a distance matrix completion problem. Let consider four points x_1, x_2, x_3 and x_4 forming a polyhedron (figure 1.3). Distance matrix $D \in \mathbb{R}^{4,4}$ encodes distances between the four points of this polyhedron. Let consider the following incomplete distance matrix D:

$$D = \begin{bmatrix} 0 & 1 & 5 & d_{14} \\ 1 & 0 & 4 & 1 \\ 5 & 4 & 0 & 1 \\ d_{41} & 1 & 1 & 0 \end{bmatrix} \tag{1.3}$$

The first property of a distance induces that $d_{14} \geq 0$ and $d_{41} \geq 0$. The third property induces that $d_{14} = d_{41}$ and finally, the fourth property of a distance

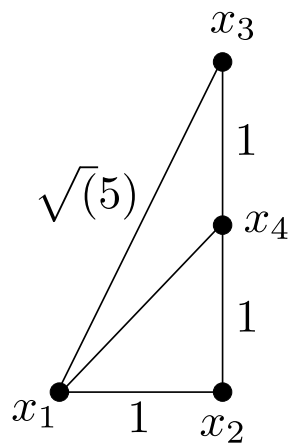


Figure 1.3: Four point polyhedron and point to point distances. Distance between x_1 and x_4 can not be retrieved from the four first properties of a distance (example taken from [8]).

allows us to bound d_{14} by $\sqrt{5} - 1 \leq d_{14} \leq 2$. However, only one value corresponds to an euclidean embedding since x_4 is at a distance of 1 from x_2 and x_3. Moreover, x_2 and x_3 are separated by a distance of 2 which induces that x_2, x_3 and x_4 must be collinear. All others distance values fulfill the first four distance properties but they do not correspond to an embedding regardless of the number of dimensions [8]. Therefore, the four properties of a distance are not sufficient to ensure that a distance corresponds to a distance in an euclidean space.

Proposition 1 (Euclidean distance) *A function $d : \mathcal{X}^2 \to \mathbb{R}^+$ corresponds to a distance in an euclidean space $\mathbb{R}^N$ if d is a distance (proposition 21) and d fulfills the relative angle inequality (definition 22).*

An alternative characterizatio of an euclidean distance function is provided by the following proposition [8]:

Proposition 2 (Euclidean distance matrix)

$$D \text{ is an euclidean distance matrix} \Leftrightarrow \begin{cases} c^t D c \leq 0 \\ e^t c = 0 \\ \|c\| = 1 \\ D \in \mathbb{S}_h^N \end{cases} \tag{1.4}$$

where $e = (1, \ldots, 1)^t$ and $\mathbb{S}_h^N$ corresponds to the set of $N \times N$ symmetric matrices having 0 main diagonal.

Since graph edit distance does not define a distance in an euclidean space it can not be directly used in machine learning methods where an implicit or

explicit vectorial embedding is mandatory. Despite this, graph edit distance encodes a dissimilarity measure between two molecular graphs which may be useful with some algorithms such as k-nearest-neighbors or k-median.

1.3 Graph embedding methods

Methods presented in section 1.1 are defined within graph space. Despite the fact that this space allows to encode complex structures as graphs, it suffers from a lack of mathematical properties. Therefore, the use of many well know machine learning methods is not possible within such a space. In order to use conjointly graphs and usual machine learning methods, one possibility consists to define an explicit vectorial space where each graph is embedded. Using such a scheme, each graph is encoded by a vector which can be used conjointly with in any machine learning methods defined on vectorial representations. Such vectorial representations of graphs are called graph embeddings.

1.3.1 Empirical kernel map

In order to combine graph edit distance and machine learning algorithms, some methods aim to define a graph embedding based on graph edit distance. Riesen has proposed to encode a graph by an explicit vector which encodes the distances between an input graph and a set of graph prototypes [39]. Given a set of prototype graphs denoted by $\mathcal{G}_{ref} = \{G_1, \ldots, G_N\}$, each graph is embedded into an explicit euclidean space by the following embedding function $\Phi^{\mathcal{G}}_{EKM} : \mathcal{G} \to \mathbb{R}^N$:

$$\Phi^{\mathcal{G}_{ref}}_{EKM}(G) = (d(G, G_1), \ldots, d(G, G_N)) \tag{1.5}$$

This method thus encodes each graph by a vector which may be used in any machine learning method defined on euclidean spaces. Although this method uses a widely used graph dissimilarity measure, the choice of prototype graphs strongly impacts the embedding quality. Riesen [39] addressed this last problem through different heuristics providing "good" sets of prototype graphs.

1.3.2 Isometric embedding

Another approach aims to define a graph embedding such that this embedding encodes a dissimilarity measure [22]. Let consider a set of graphs $\mathcal{G} = \{G_1, \ldots, G_n\}$ and a function encoding a graph dissimilarity $d : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$. We define a dissimilarity matrix $D = D_{i,j} = d(G_i, G_j)^2 \in \mathbb{R}^{n \times n}$. The method described in [22] consists in computing n vectors x_i having p-dimensions such that the distance between x_i and x_j is as close as possible from the dissimilarity measure between G_i and G_j encoded by $D_{i,j}$.

In order to define vectorial representations $(x_i)_{i \in \{1,\ldots,n\}}$, a method consists to define a matrix S encoding scalar products between the vectorial representation of graphs. Considering a distance matrix D, matrix S is defined by $S = (I - ee^t)D(I - ee^t)$ with $e = (1, \ldots, 1)^t$ and satisfies: $D_{i,j} = S_{i,i} + S_{j,j} - 2S_{i,j}$. This

last equation, corresponds to the well known relationship between euclidean distance and scalar products: $\|x_i - x_j\|^2 =< x_i, x_i > + < x_j, x_j > -2 < x_i, x_j >$. If S is a semi definite positive matrix, then its spectral decomposition is given by $S = V\Lambda V^t$ where Λ is a diagonal matrix containing the (positive) eigenvalues of S. Matrix S can then be written as XX^t with $X = V(\Lambda^{\frac{1}{2}})$. Each entry $S_{i,j}$ of S is then defined as a scalar product between two rows of matrix X. Taking x_i as an embedding of graph G_i we indeed obtain: $D_{i,j} = d^2(G_i, G_j) = \|x_i - x_j\|^2$.

However, if the dissimilarity matrix D does not correspond to a distance in an euclidean space (Definition 22), matrix S is not necessarily semi definite positive and we have to regularize it. One possibility consists to use the constant shift embedding method by removing to S its lowest (negative) eigen value: $S' = S - \lambda_n(S)I$ where $\lambda_n(S)$ denotes the lowest negative eigen value of S. This regularisation step alters the initial distances according to the following equation: $D' = D - 2\lambda_n(S)(ee^t - I)$. The magnitude of this alteration depends linearly on the lowest eigen value of S and such a regularization step should be performed only for small values of $|\lambda_n(S)|$.

1.3.3 Local approaches

Other graph embedding methods which are not based on graph edit distance can be defined using different approaches. A first family consists in encoding each graph by a set of information describing local adjacency relationships and labeling information. Attribute statistics based embedding [18] defines a graph embedding where each coordinate of the vectorial representation encodes either the number of occurrences of a node label, or the number of edges incident to two node labels. For $|L_V|$ node labels and $|L_E|$ edge labels, the size of a vector encoding a non oriented graph is thus equal to $\frac{1}{2}|L_E||L_V|(|L_V| + 1)$. This type of embedding limits the size of encoded paths and thus the amount of encoded information. Indeed, including paths of size 3 induces a huge increase of the size of vector in order to encode all possible labeled paths of size 3.

In order to encode more structural information, topological embedding [44] encodes the number of occurrences of a set of unlabeled structures corresponding to the set of isomorphic graphs having at most N nodes. Labeling information is included within a matrix representation by means of histograms: For each structure, an histogram encodes the distribution of node and edge labels among all labeled subgraphs corresponding to the unlabeled structure. This method allows to encode larger structures which encodes more structural information than paths of size 2. However, the multiple configurations of labels within all subgraphs isomorphic to a given unlabeled structure is reduced to a simple histogram of node and edge labels hence losing an important part of the label information. This trade off is induced by the limit on the size of vectors used to encode graphs.

1.3.4 Global approaches

Instead of using local approaches, some methods aim to define an embedding which globally encodes a graph rather than using a concatenation of local characteristics. Fuzzy multilevel graph embedding [25] define a vertorial representation encoding both local characteristics, such as node degrees or node and edge labels, and simple global information such as number of nodes and edges of graphs. Vectorial representation provided by a fuzzy multilevel graph embedding includes thus different levels of analysis of the graph.

Spectral embedding methods [24] base the vectorial representation of a graph from different characteristics of the spectrum of the graph laplacian matrix. The basic idea of this familly of methods is that the spectrum of the graph laplacian matrix is insensitive to any rotation of the adjacency matrix and may thus be considered as a characteristic of the graph rather than a characteristic of its adjacency matrix. The resulting graph embedding are based on global characteristics of the graphs which are however often difficult to interpret in terms of graph properties.

1.3.5 Conclusion

Graph embedding methods define graph vectorial representations. Conversely to methods defined within graph space, these vectorial representations can be used in any machine learning methods defined on vectors. However, the encoding of a graph by an explicit vector of limited size induce necessarily a loss of information. This loss may alter prediction models since some relevant information may not be encoded by a vector of limited size.

1.4 Kernel theory

Methods presented in previous sections are based on two different approaches. The first one consists in defining similarity measures between graphs not connected to any embedding. Such similarity measures may be combined with few machine learning algorithms such as the k-nearest neighbour or the k-median algorithms. The second one consists in defining vectorial representations of graphs. These methods may be combined with any machine learning algorithm but the tranformation from graphs to vectors may induce important loss of information due to the limited vector size.

In order to combine a graph similarity measure with many machine learning algorithms we have thus to transform our graphs into vectors. In the same time, the dimension of the embedding space must be sufficiently large (even infinite) in order to avoid to loss most of the graph information by this transformation. From this point of view, graph kernels provide a nice mathematical framework which allows to define a similarity measure between graphs which corresponds to a scalar product in some Hilbert space. The key point, known as the kernel trick, is that many machine learning algorithms may be rewritten solely in terms of scalar products between input data. The substitution of scalar product by graph

kernels within these methods allows then to avoid the explit transformation of graphs into vectors, hence allowing to work in Hilbert spaces of arbitrary dimension.

1.4.1 Definitions

Intuitively, a kernel $k : \mathcal{X}^2 \to \mathbb{R}$ between two objects x et x' corresponds to a similarity measure defined as a scalar product between two projections $\Phi_{\mathcal{H}}(x)$ et $\Phi_{\mathcal{H}}(x')$ of x and x' into an Hilbert space $\mathcal{H}$:

$$\forall (x, x') \in \mathcal{X}^2,\ k(x, x') = \langle \Phi_{\mathcal{H}}(x), \Phi_{\mathcal{H}}(x') \rangle. \tag{1.6}$$

In order to define a valid kernel k, it is not mandatory to explicitly define the embedding function $\Phi_{\mathcal{H}} : \mathcal{X} \to \mathcal{H}$. However, to ensure that such a projection does exist, the kernel k has to fulfill some properties.

Definition 23 (Positive definite kernel) *A positive definite kernel on $\mathcal{X}^2$ is a function $k : \mathcal{X}^2 \to \mathbb{R}$ symmetric:*

$$\forall (x, x') \in \mathcal{X}^2, k(x, x') = k(x', x) \tag{1.7}$$

and semi definite positive (Mercer's condition [30]):

$$\forall\ \{x_1,\ \ldots,\ x_n\} \in \mathcal{X}^n, \forall\ c \in \mathbb{R}^n, \sum_i^n \sum_j^n c_i k(x_i, x_j) c_j \geq 0. \tag{1.8}$$

Some usual kernels defined on vectorial representations are listed in table 1.4.1.

Usual definite positive kernels between vectorial data.

Linear	$k(x, y) = x^T y$
Gaussian	$k(x, y) = exp(-\frac{\|x-y\|^2}{2\sigma^2})$
Polynomial	$k(x, y) = (x^T y)^d + c, c \in \mathbb{R}, d \in \mathbb{N}$

Definition 24 (Gram matrix) *A Gram matrix K associated to a kernel k on a finite set $X = \{x_1, \ldots, x_N\}$ is a $N \times N$ matrix defined by:*

$$\forall (i, j) \in \{1, \ldots, N\}^2,\ K_{i,j} = k(x_i, x_j). \tag{1.9}$$

For any finite set of objects $X = \{x_1, \ldots, x_N\}$, the Gram matrix associated to a positive definite kernel k is semi definite positive. Conversely, if for any set $X = \{x_1, \ldots, x_N\}$, the Gram matrix K associated to kernel k is semi definite positive, then k is a positive definite kernel.

A positive definite kernel k can be built from a combination of a set of positive definite kernels $k_1, \ldots, k_n$ (proposition 3).

Proposition 3 (Kernel combination [2]) *Let k_1 and k_2 denote two definite positive kernels defined on $\mathcal{X}^2$, $\mathcal{X}$ corresponding to a non empty space. We have:*

1. *The set of positive definite kernels is a closed convex cone. Therefore:*
   - *let $w_1, w_2 \geq 0$, kernel $k_3 := w_1 k_1 + w_2 k_2$ is positive definite;*
   - *let k_n a sequence of positive definite kernels and $k(x, x') := \lim_{n\to\infty} k_n(x, x')$, then k is a positive definite kernel.*
2. *The product of two definite positive kernels is a positive definite kernel.*
3. *Let consider that for $i = 1, 2$, k_i is a positive definite kernel on $\mathcal{X}_i^2$, $\mathcal{X}_i$ being defined as a non empty space. Then, the tensor product $k_1 \otimes k_2$ ($k_1 k_2$) and the direct sum $k_1 \oplus k_2$ ($k_1 + k_2$) correspond to positive definite kernels on $(\mathcal{X}_1 \times \mathcal{X}_2)^2$.*

Following [1], if k is a positive definite kernel on $\mathcal{X}$, then it exists an Hilbert space $\mathcal{H}$, having scalar product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, and an embedding $\Phi : \mathcal{X} \to \mathcal{H}$ such that:

$$\forall (x, x') \in \mathcal{X}^2,\ k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}} \tag{1.10}$$

The Hilbert space $\mathcal{H}$ is called the Reproducing Kernel Hilbert Space (RKHS) of k or more usually the feature space.

1.4.2 Kernel trick

Let us consider a polynomial kernel $k(x, y) = \langle x, y \rangle^2$ with $x = \{x_1, x_2\}, y = \{y_1, y_2\} \in \mathbb{R}^2$. Kernel value $k(x, y)$ is thus equals to:

$$k(x, y) = x_1^2 y_1^2 + x_2^2 y_2^2 + \sqrt{2}(x_1 x_2)\sqrt{2}(y_1 y_2) \tag{1.11}$$

Despite the fact that this last equation does not corresponds to the usual equation of a scalar product, it indeed corresponds to a scalar product between a mapping of x and y through the following function:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \xrightarrow{\Phi} \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \end{pmatrix} \tag{1.12}$$

Note that, using a polynomial kernel of degree 2, we implicitly work in a space of dimension 3, while our original data are of dimension 2. Let us now suppose that we with to combine our kernel with a k nearest neighbor algorithm. We have thus to compute distances between vectors in the feature space of dimension 3. Such distances may be computed as follows:

$$\begin{aligned} d_k^2(x, y) \quad &= \|\Phi(x) - \Phi(y)\|^2 \quad &= < \Phi(x), \Phi(x) > + < \Phi(y), \Phi(y) > \\ & & -2 < \Phi(x), \Phi(y) > \\ & & = k(x, x) + k(y, y) - 2k(x, y) \end{aligned} \tag{1.13}$$

Therefore the k-nearest neighbor algorithm, may be applied in the feature space of dimension 3 without computing the transformation Φ but by using solely our kernel between data of dimension 2. This trick, known as the kernel trick allows

to avoid the explicit transformation of our original input data in many machine learning algorithms. This last point allows to avoid to compute and to store vectors of very large or even infinite dimension, e.g. for the Gaussian kernel (Table 1.4.1).

Applied to graphs, the kernel trick allows us to avoid the explicit transformation of graphs into vectors and to focus our attention on the definition of a similarity measure. The fact that the implicit Hilbert space may be of very large dimension, allows to reduce the loss of information induced by the transformation from graphs to vectors.

1.4.3 Kernels and similarity measures

From a mathematical point of view, a kernel is defined as a scalar product between objects embedded into an Hilbert space. However, kernels are generally considered as similarity measures. This relationship between scalar product and similarity measure may be explained by the relationship between scalar product and euclidean distance (equation 1.13):

$$\begin{array}{lll} & \|\Phi(x) - \Phi(y)\|^2 & = k(x,x) + k(y,y) - 2k(x,y) \\ \Rightarrow & k(x,y) & = \frac{1}{2}(\|\Phi(x)\|^2 + \|\Phi(y)\|^2 - \|\Phi(x) - \Phi(y)\|^2) \end{array} \tag{1.14}$$

If we consider normalized vectorial representations, i.e. $\|\Phi(x)\| = 1 \; \forall x \in \mathcal{X}$, we obtain:

$$k(x,y) = 1 - \frac{1}{2} d_k(x,y)$$

where $d_k(x,y) = \|\Phi(x) - \Phi(y)\|^2$.

In this case, kernels are defined as the opposite of the euclidean distance between vectors in the feature space. Intuitively, euclidean distance encodes a dissimilarity between objects. Therefore, a kernel function defined as a decreasing function of the distance encodes a similarity measure between objects. An high value corresponds to an high similarity whereas a low value, close to 0, encodes a high dissimilarity between objects.

1.4.4 Kernel Methods

Support vector machines

Problem definition Support Vector Machines [5] (SVM) corresponds to a machine learning algorithm. The classification problem addressed by SVM is the following: Given a set of objects labeled by a class $\{x_i, y_i\}_{i=1}^n$, $x_i \in \mathbb{R}^d$ and $y_i \in \mathcal{Y} = \{-1, +1\}$, learn a function $f : \mathbb{R}^d \to \mathcal{Y}$ such that $f(x_i) = y_i$. SVM algorithm consists in finding an hyperplane having $d-1$ dimensions which separates data points according to their classes (figure 1.4). The optimal hyperplane is the one which maximises the distance between the hyperplane and the closest data points of each class. Such a distance is called the margin. Considering the hyperplane defined by $\langle w, x \rangle + b = 0$, this distance is inversely

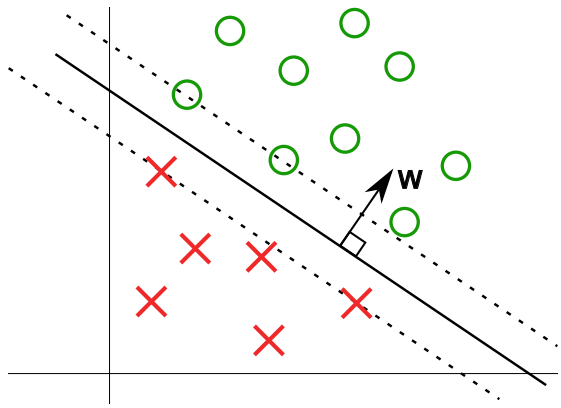


Figure 1.4: Hyperplane splitting data according to their classes (green circles and red crosses). Dashed lines encode margins.

proportional to the norm of vector w. Finding the best hyperplane thus relies to solve:

$$\begin{array}{l} \underset{w}{\text{minimize}}\ \frac{1}{2}\|w\|^2 \\ \text{subject to:} \\ y_i(\langle w, x\rangle + b) \geq 1, \forall i \in \{1, \ldots, n\} \end{array} \tag{1.15}$$

Considering a non linear separable case, i.e. where it does not exist an hyperplane having $d-1$ dimensions which separates data, Cortes and Vapnik proposed to include slack variables $\xi_i \in \mathbb{R}^+$ [7] . These variables allows to take into account classification errors during training. The addressed problem is then defined by:

$$\begin{array}{l} \underset{w}{\text{minimize}}\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to:} \\ y_i(\langle w, x\rangle + b) \geq 1 - \xi_i, \forall i \in \{1, \ldots, n\} \\ \xi_i \geq 0, \forall i \in \{1, \ldots, n\} \end{array} \tag{1.16}$$

where $C \in \mathbb{R}_+$ is a regularisation parameter which allows to weight the influence of errors made during training. An high C value favors a learning without errors, and thus potentially an over learning. Conversely, a low C value allows more errors during training and thus a larger generalization.

Using a kernel k together with SVM allows to find a linear hyperplane which solves equation 1.16 in the feature space associated to kernel k. Thanks to the kernel trick, this hyperplane may be a non linear separator in original data space. For example, let consider the kernel defined by equation 1.12. The hyperplane equation $\langle w, x\rangle + b = 0$ is computed within the kernel feature space and corresponds to a non linear equation in $\mathbb{R}^2$:

$$w_1x_1^2 + w_2x_2^2 + w_3\sqrt{2}x_1x_2 + b = 0 \tag{1.17}$$

Note that such an equation corresponds to a quadric.

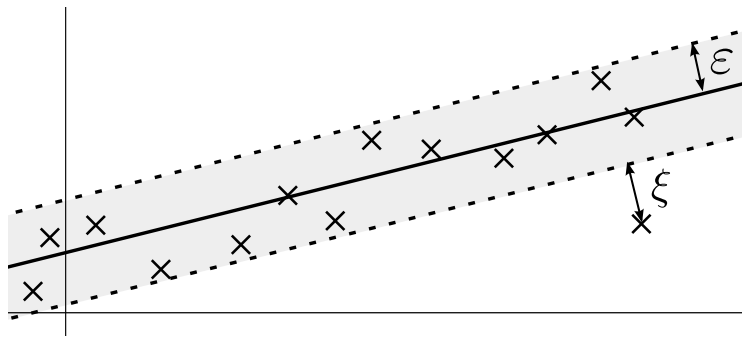


Figure 1.5: Hyperplane associated to an ε-tube (shaded).

Support Vectors Machines for regression

Regression problems consist in predicting a continuous value, conversely to classification problems which aim to predict a discrete value encoding a class. More formally, a regression problem is defined as: Given a learning set $\{x_i, y_i\}_{i=1}^n$, composed of a set of n objects $X = \{x_1, \ldots, x_n\}$ with $x_i \in \mathbb{R}^d$, each object being associated to a value $y_i \in \mathbb{R}$, learn a prediction function $f : \mathbb{R}^d \to \mathbb{R}$ such that $\hat{y}_i \simeq f(x_i)$.

Support vector machines are initially defined as a solver for classification problems. In order to handle regression problems, SVM have been adapted [10] by including an ε-insensitive cost function such as defined in [47]. Instead of computing an hyperplane splitting data according to their classes, SVM regression consists in computing an hyperplane w associated to an ε-tube which includes data points to predict.

Formally, the minimization problem addressed by SVM for regression is defined by:

$$\begin{aligned} &\underset{w}{\text{minimize}}\ \tfrac{1}{2}\|w\|^2 \\ &\text{subject to:} \\ &\begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon, \forall i \in \{1, \ldots, n\} \\ \langle w, x_i \rangle + b - y_i \geq \varepsilon, \forall i \in \{1, \ldots, n\} \end{cases} \end{aligned} \tag{1.18}$$

Minimizing equation 1.18 relies to compute a linear function which approximates y values with an accuracy ε. In order to allows errors during training, slack variables for regression have been introduced into minimization problem. Similarly to SVM classification, these slacks variables allows to weight, according to C, prediction errors made during the learning step. The minimization

problem is then defined as:

$$\begin{aligned}
&\underset{w}{\text{minimize}}\ \tfrac{1}{2}\|w\|^2 + C\textstyle\sum_{i=1}^{m}(\xi_i + \xi_i^*) \\
&\text{subject to:} \\
&\forall i \in \{1,\dots,n\} \begin{cases} y_i - \langle w, x_i\rangle - b \leq \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \geq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}
\end{aligned} \tag{1.19}$$

This regression algorithm allows to compute a prediction function up to an accuracy defined by ε. This parameter is particularly interesting in chemoinformatics since it may encode inaccuracies of physical properties induced by experiments which are not predictable using molecular graphs.

Kernel Ridge Regression

Considering ridge regression framework, this problem can be addressed by computing a linear function which encodes relationships between data x and responses y. This function may be computed by minimizing the following objective function:

$$\begin{aligned}
&\underset{w}{\text{minimize}}\ J(w) \\
&\text{with:} \\
&J(w) = \|y - Xw\|^2 + \lambda\|w\|^2
\end{aligned} \tag{1.20}$$

Minimizing the first term of equation 1.20 relies to solve a least squares problem corresponding to the minimization of prediction errors. The second term ($\lambda\|w\|^2$) corresponds to a regularisation term which aims to penalize vectors w having an high norm. The factor λ allows to weight the influence of the regularisation term into the minimization problem. Therefore, an high λ value may prevent prediction model from over learning which may induce a better generalization of the prediction model when applied on test sets.

The objective function to minimize, defined by equation 1.20, consists in a sum of L_2 norms which thus defines a convex function. Global minimum is reached for a w satisfying $\frac{\partial J}{\partial w} = 0$. This optimal w is obtained by the following analytical form:

$$w^* = X^t(XX^t + \lambda I)^{-1}y \tag{1.21}$$

Predicting property value of a new data x' relies thus on computing $\hat{y} = w^{*t}x'$.

Let consider that it exists a vector $\alpha \in \mathbb{R}^n$ such that $w = X^t\alpha$. We have thus $\alpha = (\lambda I + XX^t)^{-1}y$ and prediction of a new data is given by $\hat{y} = XX^t\alpha$. We can notice that the access to data X is only performed trough XX^t. Therefore, minimizing objective function can be performed without accessing directly to raw data $\{x_1,\dots,x_n\}$ but only trough the scalar product defined on this data since the matrix $XX^t \in \mathbb{R}^{d\times d}$ encodes scalar product $\langle x_i, x_j\rangle$ for each pair of objects $(x_i, x_j) \in \mathcal{X}^2$. Within the kernel framework, XX^t corresponds to the Gram matrix K associated to the kernel $k(x_i, x_j) = \langle x_i, x_j\rangle$. Given a non linear kernel $k_{\mathcal{H}}$, associated to a feature space $\mathcal{H}$ and an embedding function $\Phi_{\mathcal{H}} : \mathcal{X} \to \mathcal{H}$, ridge regression using $k_{\mathcal{H}}$ relies to compute a vector α lying on feature space $\mathcal{H}$, instead of original data space $\mathcal{X}$.

1.4.5 Graph kernels and chemoinformatics

Considering conclusions of sections 1.2 and 1.3, graph kernels provide a solution to both limitations involved by methods based on graph embeddings and methods defined in graph space. On one hand, graph kernels allow to define a similarity measure which is not limited by vector sizes since graph vectorial representations are not required to be explicitly and exhaustively defined. On the other hand, graph kernels guarantees that such a vectorial representation exists hence allowing the conjoint use of kernels with many machine learning methods. Therefore, graph kernels defines a natural connection between molecular graphs and machine learning algorithms. From this point of view, many graph kernels have been defined to resolve QSAR/QSPR problems in chemoinformatics.

1.5 Graph kernels based on bags of patterns

A widely used graph kernel family used in chemoinformatics consists in defining similarity between graphs from a set of sub structures extracted from graphs to be compared. Formally, given a set of sub structures $\mathcal{P}$, a convolution kernel $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ is defined as a sum of sub kernels $k_{\mathcal{P}} : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$. For each pair of sub structures $(p, p') \in \mathcal{P}^2$ extracted from graphs by a decomposition function $D : \mathcal{G} \to \mathcal{P}$, sub kernel $k_{\mathcal{P}}$ encodes the similarity between two molecular graphs according to p and p'. Kernel k is thus equal to:

$$k(G, G') = \sum_{p \subseteq D(G)} \sum_{p' \subseteq D(G')} k_{\mathcal{P}}(p, p') \tag{1.22}$$

Kernels presented in this section are defined as convolution kernels. Although these kernels use a same approach, they mainly differ on the set of sub structures used to define them.

1.5.1 Complete Graph Kernel

A first trivial approach consists in defining the set of sub structures by all possible sub graphs.

Definition 25 (Complete graph kernel) *Let G and G' two graphs. Complete graph kernel is defined by:*

$$k_{complete}(G, G') = \sum_{p \sqsubseteq G} \sum_{p' \sqsubseteq G'} k_{iso}(p, p') \tag{1.23}$$

with

$$k_{iso}(p, p') = \begin{cases} 1 \ \textit{iff}\ p \simeq p' \\ 0 \ \textit{otherwise}\ . \end{cases} \tag{1.24}$$

Complete graph kernel [13] relies on embedding graphs into an Hilbert space where each dimension encodes the number of occurrences of a subgraph of the

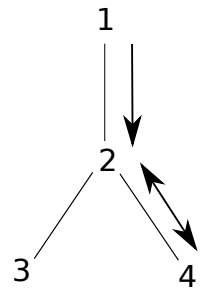


Figure 1.6: A path traversal altered by tottering corresponds to a label sequence equals to $1, 2, 4, 2, 4, 2, \ldots$.

considered graph. However, computing this kernel is NP-hard since it relies on determining if two graphs are isomorphic. Therefore, this kernel can not be efficiently computed which avoid using such an approach to chemoinformatics problems.

1.5.2 Linear patterns

Random walks

A less complex approach consists in deducing graph similarity from the number of random walks common to both graphs to be compared. Two methods [13, 23] have been proposed both based on the bag of random walks $\mathcal{W}(G)$ of a graph G. Kernels defined using random walks are based on the following formulation:

Definition 26 (Marginalized kernel) *Given two graphs G and G', $\mathcal{W}(G)$ and $\mathcal{W}(G')$ encode the sets of random walks in G and G' and $k_{\mathcal{W}} : \mathcal{W} \times \mathcal{W} \to \mathbb{R}$ is defined as a kernel between walks. Marginalized kernel is then defined as:*

$$k_{rw}(G, G') = \sum_{w \in \mathcal{W}(G)} \sum_{w' \in \mathcal{W}(G')} p_G(w) p_{G'}(w') k_{\mathcal{W}}(w, w') \tag{1.25}$$

where $p_G(w)$ corresponds to the probability of traversing w in G.

Kernel $k_{\mathcal{W}}$ corresponds to a kernel encoding a similarity between walks according to their labeling. Usually, this kernel is binary, i.e. $k_{\mathcal{W}}$ is equals to 1 if label sequences are similar and 0 otherwise. Method defined in [13] uses a different approach based on a direct product graph but is based on the same feature space and thus encodes the same information. Different implementations have been proposed[23, 49, 13, 28] and this kernel may be computed in polynomial time with the number of nodes of both graphs.

Tottering

A random walk corresponds to a node sequence which may oscillate between two connected neighbours (figure 1.6). This phenomenon, called tottering, induces

random walks which are not representative of the corresponding molecular graph structure since a same information may be repeated indefinitely. In order to limit the influence of this non representative structures, Mahé and al. [28] proposed to transform a molecular graph into an oriented graph in $\mathcal{O}(|V|^2)$. Then, kernel is computed on this new molecular representation which allows to avoid oscillations between two connected nodes and thus to reduce tottering phenomenon.

Paths

Instead of using random walks which may suffer from tottering, others kernels base their similarity measure of paths extracted from graphs to be compared. Although enumerating all paths of a graph remains a NP-Hard problem [4], a kernel can be defined using all paths composed of at most n nodes [36]. This path enumeration is performed using a depth first traversal from each node of the graph. Computing the kernel on two graphs $G = (V, E)$ and $G' = (V', E')$ relies on extracting paths from G and G' which induces $\mathcal{O}(n(|V||E| + |V'||E'|))$ operations. Enumerated paths are encoded by a vector which can be seen as a molecular descriptor set where each descriptor correspond to a path. These vectors may be computed using usual kernels such as Tanimoto kernel or MinMax kernel [36].

Another approach consists in computing similarity between graphs from the lengths of shortest paths between any pair of nodes. This method is based on Floyd's transformation [12] which consists in transforming a graph $G = (V, E, \mu, \nu)$ into a complete graph $G_F = (V, E_F, \mu, \nu_F)$ where each edge $e_f = (v_i, v_j) \in V^2$ is labeled by the length of a shortest path between v_i and v_j in G. Considering the two graph transformations $G_F = (V, E_F, \mu, \nu_F)$ and $G'_F = (V', E'_F, \mu', \nu'_F)$ of two graphs $G = (V, E, \mu, \nu)$ and $G' = (V, E, \mu, \nu)$, the shortest paths kernel is defined by:

$$k_{sp}(G, G') = \sum_{e \subseteq E_F} \sum_{e' \subseteq E'_F} k^1_{path}(e, e') \tag{1.26}$$

where $k^1_{path}(e, e')$ is a kernel defined between edges. This kernel may be defined as a Dirac function which considers two edges $e = (u, v)$ and $e' = (u, v)$ as similar if and only if $\mu(u) = \mu(u')$, $\mu(v) = \mu(v')$ and $\nu_F(e) = \nu'_F(e')$. Intuitively, two edges are similar if they connect two pairs of nodes having same labels and separated by the same distance in G and G'. This kernel can be computed in polynomial time ($\mathcal{O}(|V|^4)$), most of the time being dedicated to the computation of Floyd's transformation and to the comparison of the $|V|^2$ edges of transformed graphs. However, this approach only encodes the length of a shortest path connecting two nodes, but not the sequences of node and edge labels composing these shortest paths. This loss of information leads to a less discriminative similarity measure and thus to a less accurate kernel.

Although linear patterns allows to define a kernel which can be computed in polynomial time, most of the structural information included within molecular graphs can not be encoded using only linear patterns (figure 1.7).

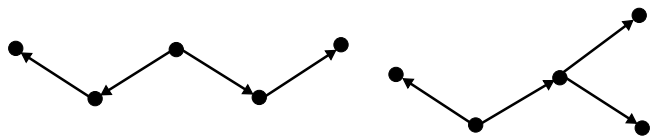


Figure 1.7: Two different oriented graphs having a same representation in a feature space based on linear patterns.

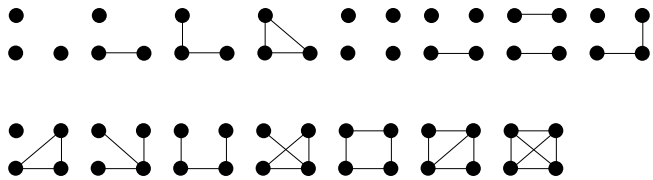


Figure 1.8: All graphlets of size 3 and 4 [42].

1.5.3 Non Linear Patterns

Graphlets

In order to increase the structural information encoded within bags of patterns, graphlet kernel [42] has been introduced. This kernel is based on bags of patterns defined as all unlabeled graphs having k nodes, $k \in \{3, 4, 5\}$ (figure 1.8). This kernel may be efficiently computed for graphs having a bounded maximum degree (say by d). Note that a molecular graph (Section 1.1.1) has a maximal degree bounded by 8.

Graphlet enumeration is divided in two steps. The first one corresponds to the enumeration of connected graphlets and the second one to the enumeration of disconnected graphlets. Connected graphlets are divided into two classes. The first class corresponds to graphlets of size k containing at least a path of size $k-1$ and the second class to others graphlets. The first class of graphlets is enumerated from paths of length $k-1$, themselves enumerated using a depth first traversal. The second class of graphlets is enumerated from *3-star* graphlets which correspond to graphlets having 4 nodes with a central node having a degree equals to 3. Others graphlets are then enumerated by looking at the neighborhood of *3-stars*. The complexity required to enumerate connected graphlets from a graph is in $\mathcal{O}(nd^4)$ where n encodes the number of nodes of the graph.

Concerning disconnected graphlets of size 3, their enumeration is based on pairs of nodes $(v_1, v_2) \in E$. If we consider a third node v_3 distinct from v_1 and v_2, three differents configurations are possible (figure 1.9). These different configurations may be distinguished using the neighborhood of (v_1, v_2). We have thus:

- $v_3 \notin \Gamma(v_1) \cup \Gamma(v_2) \rightarrow F_1$ configuration;
- $v_3 \in \Gamma(v_1) \setminus \Gamma(v_2) \vee v_3 \in \Gamma(v_2) \setminus \Gamma(v_1) \rightarrow F_2$ configuration;

- $v_3 \in \Gamma(v_1) \cap \Gamma(v_2) \rightarrow F_3$ configuration;

The method defined to enumerate graphlets of size 4 and 5 is based on the same scheme then the one used for graphlets of size $k = 3$: the set of 11 graphlets having 4 nodes is enumerated from the set of graphlets having 3 nodes. Similarly, the 34 graphlets having 5 nodes is enumerated from the ones composed of 4 nodes. This method allows to compute the distribution of graphlets of size k in $\mathcal{O}(nd^{k-1})$ operations which corresponds to a linear complexity with the number of nodes of graphs if the maximum degree is bounded. Graphlet kernel is then defined as a scalar product between normalized vectors encoding the number of occurrences of each graphlet:

Definition 27 (Graphlets kernel) *Let $G = (V, E) \in \mathcal{G}$ and $G' = (V', E') \in \mathcal{G}$ two graphs and D_G a vector encoding the number of occurrences of graphlets in G:*

$$D_G(i) = \frac{\#graphlet(i) \subseteq G}{\sum_j \#graphlet(j) \subseteq G} \tag{1.27}$$

Graphlet kernel is then defined by:

$$k_{graphlets} = D_G^\top D_{G'} \tag{1.28}$$

Although this kernel allows to encode more structural information than kernels based on linear patterns, it does not encode the similarity involved by labeling. Therefore, this kernel is limited to unlabeled graphs which corresponds to a limited application domain in chemoinformatics. Indeed, atom's chemical elements are an important information and must be taken into account in order to define an accurate similarity measure.

Tree patterns

In order to include molecular graphs labeling into kernel computation, tree pattern kernel [37, 26] is defined as a kernel based on non linear and labeled patterns. This method deduces the similarity between two graphs from the number of their common tree patterns.

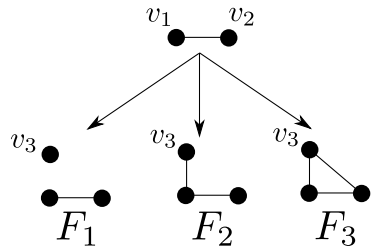


Figure 1.9: Differents possible topological configurations involving a pair of nodes and a third node.

Definition 28 (Tree pattern) *Let $G(V,E) \in \mathcal{G}$. If it exists a node $r \in V$, then r is a tree pattern of G rooted in r and having an height equals to 1. let $t_1, t_2, \ldots, t_n$ n tree patterns respectively rooted in $r_1, r_2, \ldots, r_n$, with $r_i \neq r_j, \forall i \neq j$. If $(r, r_1), (r, r_2), \ldots, (r, r_n) \in E$, then $r(t_1, t_2, \ldots, t_n)$ is a tree pattern rooted in r. r is defined as the parent of each r_i.*

Each tree pattern is associated to a dimension in the feature space. Each dimension encodes the number of tree patterns included within the two graphs to be compared.

Definition 29 (Neighborhood matching set) *Let $G = (V, E, \mu, \nu)$ and $G' = (V', E', \mu', \nu')$ denote two graphs and let us consider two nodes $r \in V$ and $s \in V'$. Neighborhood matching set $M_{r,s} \in (\Gamma(r), \Gamma(s))$ is defined by:*

$$\begin{aligned} M_{r,s} = \{R \subseteq \Gamma(r) \times \Gamma(s) \mid \quad & (\forall (a,b),(c,d) \in R : a \neq c \wedge b \neq d) \wedge \\ & (\forall (a,b) \in R : \mu(a) = \mu'(b) \wedge \nu(r,a) = \nu'(s,b))\} \end{aligned} \tag{1.29}$$

Definition 30 (Kernel between tree patterns) *Let $G = (V, E, \mu, \nu)$ and $G' = (V', E', \mu', \nu')$ two graphs, kernel between two tree patterns having a height equals to h and rooted in r and s is defined by:*

$$k(r, s, h) = \lambda_r \lambda_s \sum_{R \in M_{r,s}} \prod_{(r',s') \in R} k(r', s', h-1). \tag{1.30}$$

For $h = 1$, $k(r, s, 1)$ is equals to 1 if $\mu(r) = \mu'(s)$, 0 otherwise.

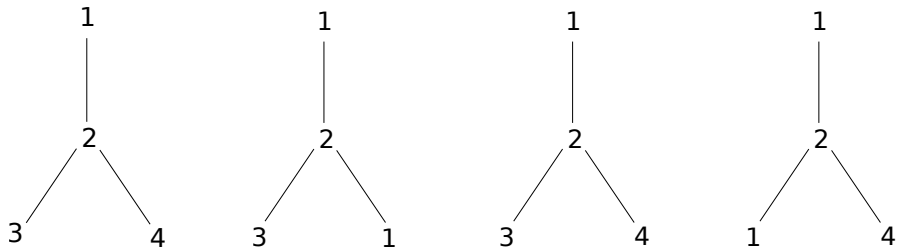
Kernel value is strictly greater than 0 if the two tree patterns are isomorphic. Parameters λ_r and λ_v are defined as positive real numbers lower than 1 such that large tree patterns have a low contribution. Tree pattern kernel is then defined as:

Definition 31 (Tree pattern kernel) *Let $G = (V, E, \mu, \nu) \in \mathcal{G}$ and $G' = (V', E', \mu', \nu') \in \mathcal{G}$. Tree pattern kernel for an height h is defined by:*

$$k(G, G', h) = \sum_{r \in V} \sum_{s \in V'} k(r, s, h) \tag{1.31}$$

Weisfeiler-Lehman kernels

Weisfeiler-Lehman kernels [43, 41] correspond to graph kernels based on tree patterns. The main difference with tree pattern kernel is that this method is based on Weisfeiler-Lehman isomorphism test. This algorithm allows to compute the kernel in linear time with the number of nodes of graphs which provide a faster computational time than tree pattern kernel (section 1.5.3). Moreover, Weisfeiler-Lehman kernels computes an explicit representation of the feature space associated to kernels. This explicit enumeration allows to performed tree pattern enumeration only N times. Using an implicit enumeration, pattern enumeration has to be performed for each pair of graph, i.e. N^2 times.



(a) Tree of size 4

(b) Tree patterns of size 4 included in (a)

Figure 1.10: Difference between tree patterns and trees : graph (a) includes only one sub tree of size 4 rooted in 1 (itself) but 3 different tree patterns (b), all rooted in 1.

One has to notice that tree patterns differ from trees (figure 1.10). Similarly to the difference between paths and walks, a sub tree pattern may include a same node twice. Therefore, information encoded by tree patterns may also be altered by tottering (section 1.5.2). However, graph transformation proposed by Mahé and al. to prevent tottering in random walks kernel can be applied to kernels based on tree patterns.

Graph fragments

Graph fragment kernel [51] corresponds to a kernel based on a set of patterns defined as all connected sub graphs having at most l edges. This set of subgraphs, denoted GF, can be divided into different subsets:

- *PF (Path Fragments):* encodes all linear patterns;
- *TF (Tree Fragments):* encodes all sub trees having at least one node v such that $d(v) \geq 2$.
- *AF (Acyclic Fragments)* is defined as $AF = TF \cup PF$ which thus corresponds to all sub trees.
- The fourth sub set corresponds to the set difference $GF \setminus AF$. This subset encodes all sub graphs containing at least one cycle.

This set of patterns defines the feature space associated to kernel where each coordinate encodes either the absence/presence or the number of occurrences of a given sub graph. A molecular graph G is encoded by a vector $\Phi(G) = \{\Phi_1(G), \ldots, \Phi_d(G)\}$, where d encodes the number of different patterns enumerated for a given length l. The kernel between two graphs G and G' is then defined as a Min-Max kernel [36] between the two vectors $\Phi(G)$ and $\Phi(G')$.

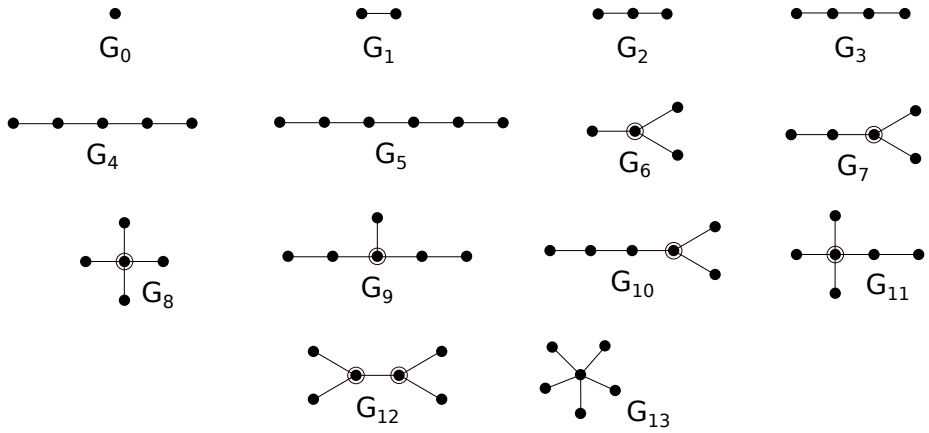


Figure 1.11: Set of treelet's structures

This method, based on an exhaustive enumeration of all sub graphs, relies to compute complete graph kernel for a l big enough (section 1.5.1). Since this problem is a NP-hard problem and thus not feasible for large graphs, graph fragment is generally defined for $l = 7$ in order to obtain reasonable computational times. Moreover, no significant accuracy gain is observed for length $l \geq 5$.

Treelet Kernel

Conversely to methods based on tree patterns, treelet kernel [15] uses a bags of patterns defined as a sub set of strict sub trees, called treelets. Treelets are defined as the set of all labeled trees having at most 6 nodes (figure 1.11). Conversely to methods based on a non limited set of substructures, such as the tree pattern kernel, considering a set of pre defined structures allows to define an efficient *ad hoc* linear enumeration. This enumeration is performed in two steps:

The first step consists to identify treelet's structure. This step is based on the method used to enumerate connected graphlets [42] (section 1.5.3). Graphlet enumeration method is extended to enumerate tree structures up to 6 nodes while keeping a linear complexity when considering degree bounded graphs. This first step allows to associate a structure index $(G_0, \ldots, G_{13})$ (Figure 1.11) to each treelet.

The second step consists to encode the labels of each instance of treelet found within a graph. This step is based on a tree traversal of each instance of treelet based on Morgan algorithm [31]. This traversal provides a string which as shown in [15] identifies the label of each node and each edge of a treelet given the index of its structure. Note that, since the set of structures is finite, the computation of the string encoding treelet's labels is performed in constant time. This last point is an important advantage over alternative encodings used for example by frequent subgraphs or graph fragments methods [9].

The concatenation of treelet's index and treelet's key defines a unique code for each treelet which allows to perform an explicit enumeration of all treelets composing a graph. Based on this enumeration, we define a function f which associates to each graph G a vector $f(G)$ whose components encode the number of occurences of each treelet t found in G:

$$f(G) = (f_t(G))_{t \in \mathcal{T}(G)} \text{ with } f_t(G) = |(t \subseteqq G)| \tag{1.32}$$

where $\mathcal{T}(G)$ denotes the set of treelets extracted from G and $\subseteqq$ the subgraph isomorphism relationship. Then, similarity between treelet distributions is computed using a sum of sub kernels between treelet's frequencies:

$$k_{\mathcal{T}}(G, G') = \sum_{t \in \mathcal{T}(G) \cap \mathcal{T}(G')} k(f_t(G), f_t(G')) \tag{1.33}$$

where $k(.,.)$ defines any positive definite kernel between real numbers such as linear kernel, gaussian kernel or intersection kernel. Note that, conversely to tree pattern kernel (section 1.5.3), this kernel explicitly enumerates sub trees by computing the number of occurrences of each pattern.

1.5.4 3D pattern

Three-dimensional molecular information may be an important and useful information for some chemoinformatics problems such as docking or optical angle rotation. Among the few existing kernels including such information, 3D pharmacophore kernel [27] encodes a set of patterns corresponding to triplets of distinct atoms. Each pattern can thus be understood as a potential pharmacophore and encodes euclidean distances between each pair of nodes included within a corresponding triplet. These distances are measured on the most stable 3D conformation of a molecule. Pharmacophore kernel is defined as a convolution kernel based on a kernel $k_{\mathcal{P}}$ defined on atom triplets. This last kernel is defined as a combination of two kernels:

- A first kernel which compares nodes labels of the two triplets to be compared;
- a second kernel which encodes spatial information by comparing distances between atoms by means of gaussian kernels. More formally, if $p = (x_i)_{i=\{1,\ldots,3\}}$ and $p' = (x'_i)_{i=\{1,\ldots,3\}}$ are two atom triplets. This kernel is defined as:

$$k_{spatial}(p, p') = \prod_{i=1}^{3} k_{dist}(\|x_i - x_{i+1}\|, \|x'_i - x'_{i+1}\|) \tag{1.34}$$

  Kernel k_{dist} may be defined as a gaussian kernel and aims to compare two distances.

This kernel encodes spatial information of molecular compounds and allows to define a relevant similarity measure for chemoinformatics problems including 3D information. However, distances used in this kernel are relative to a priori 3D conformations of both molecules while some alternative conformations may better explain the property to predict.

1.5.5 Cyclic patterns

Similarly to 3D information, cyclic information may have a particular influence on molecular properties since cycles reduce the atom's degrees of freedom.

Cyclic pattern kernel

A first approach consists in defining a kernel based on the set of simple cycles of a molecular graph (figure 1.12). Considering a graph G, cyclic pattern kernel [21] is based on a decomposition of G into two sub sets:

- A first subset $C(G)$ as the set of simple cycles included within G (figure1.12);
- and a second subset $\mathcal{B}(G)$ defined as the set of bridges of G and corresponding to atoms and edges not included in $C(G)$. $\mathcal{B}(G)$ corresponds thus to a forest.

Considering these two sub sets, cyclic pattern kernel is defined as the sum of a tree pattern kernel applied on $\mathcal{B}(G)$ together with a kernel applied on $C(G)$. Considering two graphs G and G', cyclic pattern kernel is defined as:

$$k_C(G, G') = k_{TP}(\mathcal{B}(G), \mathcal{B}(G')) + k_{inter}(C(G), C(G')) \tag{1.35}$$

where $k_{TP}(\mathcal{B}(G), \mathcal{B}(G'))$ corresponds to tree pattern kernel (section 1.5.3) and $k_{inter}(C(G), C(G'))$ is defined as an intersection kernel which computes the number of common cycles in $C(G)$ and $C(G')$. Similarity between molecular graphs is thus deduced from a kernel encoding acyclic similarity and a kernel encoding cyclic similarity.

However, computing the set of simple cycles of a graph is a NP-hard problem which may however be addressed in chemoinformatics problem since molecular graphs generally include a low number of simple cycles. In order to tackle this complexity, cyclic pattern kernel has been revisited [20] and has been defined using the set of relevant cycles (denoted $\mathcal{C}_{\mathcal{R}}$) [50] instead of the set of simple cycles. The enumeration of relevant cycles can be performed in polynomial time which reduces kernel computational complexity. Relevant cycles can be understood as a set of cycles which defines a basis for encoding all cycles included in molecular graph (figure 1.13).

Cyclic pattern kernel allows to explicitly encode cyclic similarity into kernel computation which may provide an accurate similarity measure for chemoinformatics problems. However, although this approach allows to catch some cyclic information, it does not encode relationships between molecular cycles.

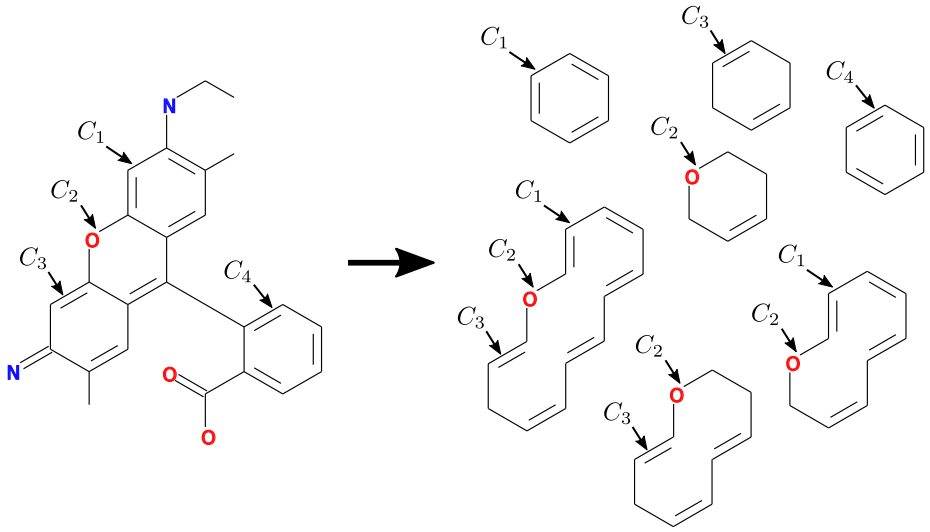


Figure 1.12: Decomposition of a molecular graph into its set of simple cycles.

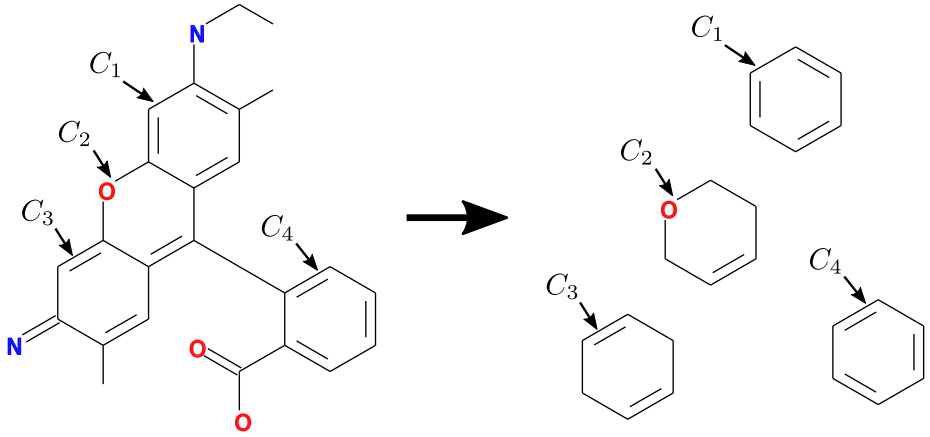


Figure 1.13: Decomposition of a molecular graph into its set of relevant cycles.

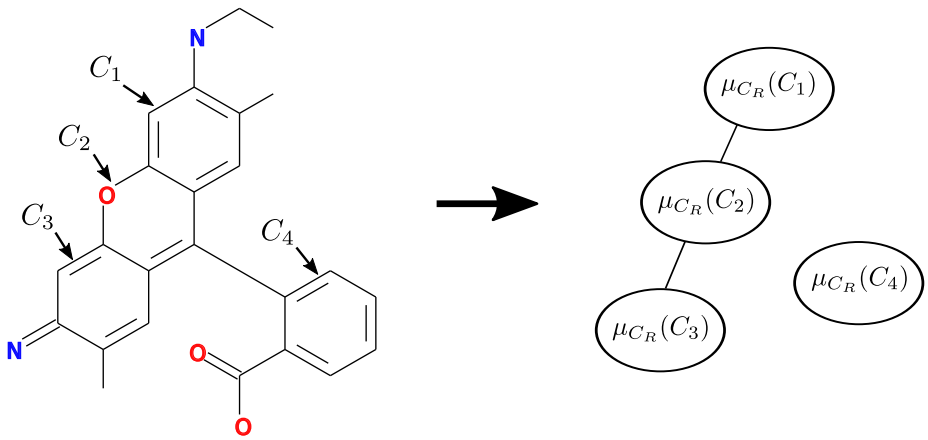


Figure 1.14: Relevant cycle graph computed from a molecular graph.

Relevant cycle graph

In order to encode more cyclic information, relevant cycle relationships may be encoded using relevant cycle graph [16]. This graph representation aims to encode adjacency relationships between relevant cycles and thus to encode the cyclic system of a molecule. This graph is defined as $G_{\mathcal{C}} = (\mathcal{C}_{\mathcal{R}}, E_{\mathcal{C}_{\mathcal{R}}}, \mu_{\mathcal{C}_{\mathcal{R}}}, \nu_{\mathcal{C}_{\mathcal{R}}})$ where each vertex encodes a relevant cycle and two vertices are connected by an edge if corresponding cycles share at least one vertex of the initial graph (figure 1.14). Relevant cycle graph labeling functions are defined as follows:

- $\mu_{\mathcal{C}_{\mathcal{R}}}(C)$: Each cycle C is defined by a sequence of edge and vertex labels encountered during the traversal of C. In order to obtain a sequence invariant to cyclic permutations, $\mu_{\mathcal{C}_{\mathcal{R}}}(C)$ is defined as the sequence having the lowest lexicographic order.

- $\nu_{\mathcal{C}_{\mathcal{R}}}(e)$: An edge e in G_C encodes a path shared by two cycles and corresponds to a sequence of edge and node labels. Since such a path may be traversed from its two extremities, we define $\nu_{\mathcal{C}_{\mathcal{R}}}(e)$ as the sequence of lowest lexicographic order.

In order to compute a cyclic similarity using relevant cycle graphs, treelet kernel (section 1.5.3) may be applied on the set of treelets extracted from relevant cycle graphs where nodes encode cycles. This kernel is thus defined as follows:

$$k_{\mathcal{C}}(G, G') = \sum_{t_{\mathcal{C}} \in \mathcal{T}(G_{\mathcal{C}}) \cap \mathcal{T}(G'_{\mathcal{C}})} k(f_{G_{\mathcal{C}}}(t_{\mathcal{C}}), f_{G'_{\mathcal{C}}}(t_{\mathcal{C}})) \tag{1.36}$$

Then, using cyclic pattern kernel, a kernel based on relevant cycle graph may be defined as a sum of a kernel encoding acyclic similarity, such as treelet kernel $k_{\mathcal{T}}$ applied on original molecular graph, and a kernel $k_{\mathcal{C}}$ applied on relevant

cycle graph:

$$k(G, G') = k_{\mathcal{T}}(G, G') + \lambda k_{\mathcal{C}}(G, G') \tag{1.37}$$

where $\lambda \in \mathbb{R}_+$ allows to weight the contribution of cyclic kernel.

This kernel allows to take into account more cyclic information since, conversely to cyclic pattern kernel, relationships between relevant cycles are encoded in the relevant cycle graph. However, such as cyclic pattern kernel, global similarity measure is based on a sum which splits cyclic information and acyclic information. This last point constitutes a drawback since within the chemoinformatics framework, relationships between a cycle and its substituents may have an influence on molecular properties and must be encoding in kernel definition.

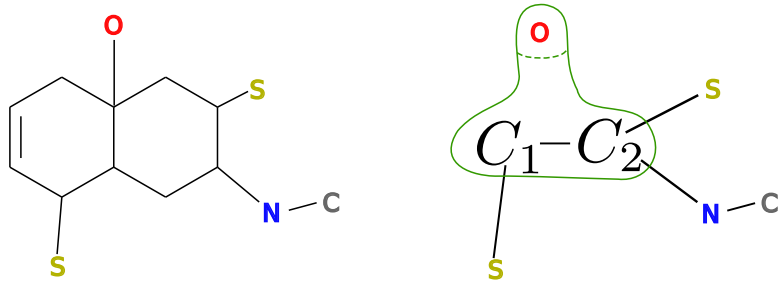
Relevant cycle hypergraph

Relevant cycle hypergraph [17] (figure 1.15) corresponds to an hypergraph representation [3] which allows to encode both cyclic and acyclic information into a single representation. This molecular representation is based on relevant cycle graph which encodes molecular cyclic systems. Relevant cycle graph may be augmented by adding acyclic parts in order to obtain the relevant cycle hypergraph. Given a graph $G = (V, E, \mu, \nu)$, its corresponding relevant cycle hypergraph $H_{RC}(G) = (V_{RC}, E_{RC})$ is defined by as follows:

- The set of nodes V_{RC} is defined as an union of two subsets:
  1. A first subset corresponding to the set of relevant cycles,
  2. and a second subset corresponding to the set of atoms not included within a cycle of G.

  All atoms of the original molecular graph are thus encoded in the relevant cycle hypergraph either by a vertex encoding a cycle or by the atom itself. In the same way as for vertices,

- the set of hyperedges E_{RC} is composed of two subsets:
  1. A set of edges composed of:
     - edges between relevant cycle vertices, corresponding to the set of edges of the relevant cycle graph,
     - edges of G which connect two acyclic atoms or connect a single relevant cycle to another single relevant cycle (C_1 and C_2 in figure 1.15) or an acyclic part of G (C_1 and S in figure 1.15),
  2. and a set of hyperedges which allows to encode special cases where an edge connects at least two distinct relevant cycles to another part of the molecule. For example, an hyperedge is required to encode adjacency relationship between C_1 and C_2 in one hand and an oxygen atom on the other hand in figure 1.15.



(a) Cyclic molecular graph.

(b) Relevant cycle hypergraph representation of (a).

Figure 1.15: Relevant cycle hypergraph representation of a molecule. We can note that hyperedge (in green) encodes an adjacency between oxygen atom in one hand and two cycles in the other hand.

Similarly to vertices, each atomic bond included in a molecular graph is encoded in its relevant cycle hypergraph. Therefore, relevant cycle hypergraph encodes all information included in molecular graph representation.

Considering such a new hypergraph representation, similarity must be computed using an hypergraph kernel. In order to encode local relationships between cycles and substituents, treelet kernel has been adapted [17] in order to enumate treelets within relevant cycle hypergraphs. This kernel defines a similarity based both on an enumeration of relevant cycles of both graphs (pattern G_0 in figure 1.11) and on an enumeration of more complex treelets encoding relationships between cycles and relationships between cyclic and acyclic parts of a molecule.

1.5.6 Pattern Weighting

Considering graph kernels based on bags on patterns, it may be interesting to weight each pattern according to its influence for a given property. Similarly to the notion of pharmacophore, some patterns may have a particular influence whereas some do not encode any information about the property to predict. Considering a kernel defined as a sum of sub kernels, where each sub kernel encodes a similarity according to a given pattern $p \in \mathcal{P}$, pattern weighting relies on computing a weight for each term of the sum. The weighted kernel is defined as:

$$k_{\text{weigh}}(x, x') = \sum_{p \in \mathcal{P}} w(p) * k_p(x, x') \tag{1.38}$$

where $w : \mathcal{P} \to \mathbb{R}_+$ encodes the influence of each pattern: an high value $w(p)$ encodes an high influence of pattern p and thus corresponds to an high con-

tribution to the similarity measure. Conversely, a $w(p)$ close to or equals to 0 consists in removing the contribution of pattern p to the similarity measure.

A priori methods

Tree pattern kernel as defined by Mahé [29] (Section 1.5.3) includes a tree pattern weighting. This kernel is defined on the same feature space than the original kernel but includes a weighting of each pattern:

Definition 32 (Weighted tree pattern kernel) *Let us consider two graphs $G = (V, E) \in \mathcal{G}$ and $G' = (V', E') \in \mathcal{G}$ together with the set of tree patterns of size h : $\mathcal{T}_h$ a counting function of tree patterns: $\phi_t : \mathcal{G} \to \mathbb{N}$ and a weighing function: $w : \mathcal{T}_h \to \mathbb{R}^+$. The weighted tree pattern kernel is defined by:*

$$k^h(G, G') = \sum_{t \in \mathcal{T}_h} w(t)\phi_t(G)\phi_t(G'). \tag{1.39}$$

This tree pattern kernel definition corresponds to an adaptation of equation 1.38 to tree pattern kernel. Two weighting functions based on structural information included in tree patterns have been proposed [29].

Branching cardinality

The first one is based on branching cardinality. Branching cardinality is defined as the number of leaves -1. Given a tree pattern $t = (V, E)$, branching cardinality is defined by:

$$branch(t) = \sum_{v \in V} \mathbf{1}(\Gamma(v) = 1) - 1 \tag{1.40}$$

where $\mathbf{1}(x)$ is equals to 1 if x is true. Weighting function is then defined by:

$$w_{branch}(t) = \lambda^{branch(t)} \tag{1.41}$$

where $\lambda \in \mathbb{R}^*_+$. This weighting function aims to favor linear patterns if $\lambda < 1$. Conversely, structurally complex patterns will be favored if $\lambda > 1$.

Ratio Depth/Size

The second weighting function is based on ratio between height and size of a tree pattern. For a given tree pattern $t = (V, E)$ rooted in $r \in V$, the size-based weighting function is defined by:

$$w_{size}(t) = \lambda^{|V|-h} \tag{1.42}$$

Similarly to section 1.5.6, $\lambda < 1$ favors tree patterns having similar sizes and heights, which correspond to linear patterns. Conversely, $\lambda > 1$ favor more structurally complex tree patterns.

These two weighting functions allow to weight tree pattern influence according to their structural information. Parameter λ must be tuned using *a priori* chemical knowledge or cross validation. However, this weighting framework does not allow to weight specifically each tree pattern according to a property to predict.

Multiple Kernel Learning

Considering a pattern weighting problem as defined in equation 1.38, Multiple Kernel Learning (MKL) aims to compute an optimal weighting function w according to a given dataset. Considering a finite set of m patterns, weighting function w is encoded by a vector $d \in \mathbb{R}_+^m$ where each coordinate d_i encodes a weight associated to a pattern. Kernel is then defined as:

$$k_{\text{MKL}}(x, x') = \sum_{i=1}^{m} d_i * k_i(x, x') \tag{1.43}$$

Multiple kernel learning methods consist in computing an optimal vector d according to a prediction task.

Simple MKL

Simple MKL [35] consists in optimizing problems addressed by SVM (section 1.4.4) by computing an optimal weighting. Considering a kernel as defined in equation 1.43, simple MKL consists in solving:

$$\underset{d}{\text{minimize}}\ J(d) \text{ such that } \begin{cases} \sum_{i=1}^{m} d_i = 1 \\ d_i > 0,\ \forall i \in \{1, \ldots, m\} \end{cases} \tag{1.44}$$

with

$$J(d) = \begin{cases} \min\limits_{w,b,\xi} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \\ \text{subject to:} & y_i(\langle w, x\rangle + b) \geq 1 - \xi_i, \forall i \in \{1, \ldots, n\} \\ & \xi_i \geq 0,\ \forall i \in \{1, \ldots, n\} \end{cases} \tag{1.45}$$

This minimization problem includes a constraint on the L_1 norm of vector d which induces sparsity on vector d. This sparsity allows to keep only most relevant patterns into kernel computation removing irrelevant ones. Simple MKL problem is resolved by alternating a classical SVM resolution together with a projected gradient descent according to vector d. This projected gradient step allows to minimize objective function while ensuring that constraints on vector d are fulfilled. Since problem defined in equation 1.44 is convex, iterating these two steps leads to the optimal vector d.

Generalized MKL

In order to allow more flexibility on sparsity constraint, generalized MKL [48] defines an objective function which includes a Tikhonov regularization on vector d instead of an equality constraint in simple MKL. Corresponding minimization problem is defined as:

$$\underset{d}{\text{minimize}}\ J(d) \text{ such that } d_i > 0,\ \forall\ i \in \{1, \ldots, m\} \tag{1.46}$$

with

$$J(d) = \begin{cases} \min\limits_{w,b,\xi} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sigma\|d\|_1 \\ \text{subject to :} & y_i(\langle w, x\rangle + b) \geq 1 - \xi_i, \forall i \in \{1, \ldots, n\} \\ & \xi_i \geq 0,\ \forall i \in \{1, \ldots, n\} \end{cases} \tag{1.47}$$

Parameter σ allows to weight the influence of L_1 norm regularization into the minimization problem. An high λ favors vector d having a low L_1 norm and thus sparse vectors. Conversely, a low σ relaxes sparsity constraint. The method used to resolve this problem is closely similar to the one proposed to resolve MKL method. The main difference is that the gradient is no longer projected on simplex since equality constraint does no longer exists.

Infinite MKL

In order to be able to deal with thousands patterns, one have to either compute each gram matrix for each sub kernel at each iteration or store each gram matrix. When considering thousands of patterns and graphs, this two options induce too much computational time or memory space to be applicable. In order to be able to handle such datasets, infinite MKL [53] defines a MKL method based on simple MKL which only considers a sub set of sub kernels at each iteration.

This method only performs optimization using a sub set of active kernels, that is to say kernels having a weight d_i strictly greater than 0. At the end of each simple MKL optimization, active kernel set is updated by removing kernels having a null weight and potentially active kernels are added to the set of active kernels. Potentially active kernels may be determined using an oracle based on KKT conditions and gradients associated to each sub kernel [53].

This approach allows to consider an high, possibly infinite, number of sub kernels and allows to select most relevant ones according to a particular property to predict.

1.6 Experiments

Differents kernels presented in section 1.5 have been tested on several chemoinformatics datasets. These datasets are divided in regression and classification problems[1]

[1] All these dataset are available on the IAPR TC15 Web page: https://iapr-tc15.greyc.fr/links.html

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

Method	**RMSE**($^\circ C$)
(1) Edit Distance	10.27
(2) Graph embedding	10.19
(3) Path kernel	12.24
(4) Random walks kernel	18.72
(5) Tree pattern kernel	11.02
(6) Treelet Kernel	8.10
(7) Treelet kernel with MKL	5.24

1.6.1 Boiling point prediction

The first prediction problem used to test different methods is based on a dataset composed of 185 acyclic molecules [6]. This prediction problem consists in predicting boiling point of molecules. The first line of table 1.6.1 shows results obtained by a gaussian kernel applied on approximate edit distance (section 1.2.3). Note that since graph edit distance does not define an euclidean distance, Gram matrix is regularized in order to be semi-definite positive. The second line corresponds to the embedding method described in section 1.3.1. We can first note that methods based on edit distance obtains intermediate results on this dataset. These results may be due either on the approximation of edit distance or the loss of information induced by regularization or embedding. Next lines corresponds to graph kernels defined in section 1.5.3). Line 3 corresponds to a kernel based on paths (section 1.5.2 and line 4 corresponds to a random walks kernel (section 1.5.2). These two kernels suffer from the low expressiveness of linear patterns and we can note that tottering phenomenon degrades prediction accuracy. Finally the third last lines corresponds to kernels based on non linear patterns. As we can see line 7, best results are obtained thanks to the combination of multiple kernel learning (section 1.5.6) which allows to only considers relevant patterns.

This first experiment highlights the gain obtained by using non linear patterns instead of linear patterns. Moreover, pattern weighting step allows to greatly increase prediction accuracy by removing irrelevant patterns from similarity measure computation. This weighting step allows to get best results among tested methods

1.6.2 Classification

The two next experiments correspond to binary classification problems which consist in predicting if a molecule has a particular activity or not.

AIDS dataset

First classification experiment has been performed on a graph database provided by [40]. This dataset, defined from the AIDS Antiviral Screen Database of

Results on AIDS dataset.

Method	**Classification Accuracy**
(1) KNN on graph edit distance	97.3%
(2) Graph embedding based on graph edit distance	98.2%
(3) Gaussian kernel on graph edit distance	99.7%
(4) Path kernel	98.5%
(5) Random walks kernel	98.5%
(6) Treelet Kernel	99.1%
(7) Treelet Kernel with MKL	99.7%

Active Compounds, is composed of 2000 chemical compounds some of them being disconnected. These chemical compounds have been screened as active or inactive against HIV and they are split into three sub sets:

- A train set composed of 250 compounds used to train SVM;
- a validation set composed of 250 compounds used to find parameters set giving the best accuracy result;
- A test set composed of the remaining 1500 compounds used to test the classification model.

This dataset is composed of a large set of different chemical compounds including both cyclic and acyclic molecules and composed of several heteroatoms.

Table 1.6.2 shows results obtained by different methods on this dataset. First line of table 1.6.2 corresponds to a classifier defined as a k-nearest neighbours algoritm using approximate graph edit distance. Line 2 corresponds to a graph embedding as described section 1.3.1 and line 3 corresponds to a gaussian kernel applied on graph edit distance and regularized. We can first note that the use of k-nearest neighbour or graph embedding approach does not lead to good results on this dataset. Conversely, gaussian kernel on edit distance (table 1.6.2, line 4) combined with SVM obtains the best results on this dataset. Note that the regularization added to Gram matrix does not alter prediction accuracy. This may be explained by the fact that classification results may not be altered by a reasonable distortion of the graph edit distance induced by the regularization step. Conversely regression problems which consists to predict a real value instead of a binary class may be altered by any modification of the initial distance. Next lines correspond to kernels using differents bags of patterns. Kernels corresponding to line 4 and line 5 are based on linear patterns (section 1.5.2). As seen in regression experiment, the low expressiveness of kernels based on linear patterns leads to poor results on this dataset. Conversely, methods based on non linear patterns (lines 6 and 7) obtain better results since they encode more structural information than kernels based on linear patterns. Moreover, as observed in regression experiment, multiple kernel learning step allows to reach the best results on this classification problem. Note that structures having biggest weights can be assimilated to pharmacophores. These relevant patterns according to MKL may be analyzed by chemical experts.

This experiment confirms the conclusions drawn in section 1.6.1 on a regression problem with only acyclic molecules.

PTC dataset

The second classification experiment is a classification problem taken from the Predictive Toxicity Challenge [46] which aims to predict carcinogenicity of chemical compounds applied to female (F) and male (M) rats (R) and mice (M). This experiment is based on ten different datasets, each of them being composed of one trainset and one testset. Table 1.6.2 shows the number of correctly classified molecules over the ten testsets for each method and for each class of animal.

Table 1.6.2 shows results obtained by different kernels encoding cyclic information in different ways. The first line of this table corresponds to treelet kernel which does not encode any cyclic information since it is only based on acyclic patterns. Lines 2 to 4 correspond to methods encoding different levels of cyclic information. Line 2 corresponds to cyclic pattern kernel which simply compares common cycles (section 1.5.5). Line 3 corresponds to a treelet kernel applied on relevant cycle graph (section 1.5.5) which encodes cycles relationships. Line 4 corresponds to results obtained by treelet kernel adapted to relevant cycle hypergraph (section 1.5.5) comparison [17]. First, we can note that best results are obtained by kernel which encodes both cyclic and acyclic relationships which validates the relevance of including cyclic information, and more particularly adjacency relationships between cyclic and acyclic parts of a molecule.

Then, lines 5 to 7 correspond to different kernels combined with multiple kernel learning. This weighting step shows that kernel based on cyclic information obtains best results on two datasets over four (line 7, datasets MM and FM) and kernel only based on acyclic patterns obtains best results over the two others datasets (line 5, datasets MR and FR). Note that pattern weighting step allows us to reduce the number of patterns included within kernel computation from about 3500 to 150, depending on dataset. Finally, since differents kernels obtain best results over the four datasets, weighted combination of kernel encoding cyclic information and a kernel encoding acyclic information leads to the best results on this dataset (line 8). This combination shows the flexibility of kernel approaches by means of multiple kernel learning and linear combinations of kernels.

1.7 Conclusion

Graph kernel framework allows to define scalar products between implicit or explicit vectorial representations of graphs in a given feature space. On one hand, conversely to methods based on graph theory, graph kernels can be used in well known and widely used machine learning methods such as SVM. On the other hand, exemption of an explicit vectorial representation allows to encode more information than methods based on an explicit and fixed size vectorial representation. This characteristic allows to define accurate graph similarity

Classification accuracy on PTC dataset.

Method	**MM**	**FM**	**MR**	**FR**
(1) Treelet kernel (TK)	208	205	209	212
(2) Cyclic pattern kernel	209	207	202	228
(3) TK on Relevant cycle graph	211	210	203	232
(4) TK on relevant cycle hypergraph (TCH)	217	224	207	233
(5) TK + MKL	217	224	223	250
(6) TC + MKL	216	213	212	237
(7) TCH + MKL	225	229	215	239
(8) TK + λTCH	225	230	224	252

measures which encode most of structural and labeling information encoded by molecular graphs. Therefore, graph kernels allows to combine efficient machine learning methods with accurate and expressive similarity measures.

Defining a graph kernel consists in defining a graph similarity measure which encodes a maximum of useful information and which fulfills all required properties to define a positive definite kernel. Kernels based on bags of patterns deduce molecular graph similarities from similarities of bags of patterns extracted from these graphs. Kernels based on bags of patterns aim to encode a maximum of information while keeping an efficient computational time in order to be applicable to datasets encountered in chemoinformatics. Kernels based on non linear patterns encode more structural information than kernels based on linear patterns and can be computed in linear time when applied on molecular graphs. Moreover, some bags of patterns are defined such as they explicitly encode cyclic information into similarity measure. This information is particularly useful in chemoinformatics since molecular cycles have a great influence on molecular properties. Among kernels based on bags of patterns, relevant cycle hypergraph encodes both acyclic and cyclic parts and their relationships into a single representation which allows to explicitly encode adjacency relationships between a cycle and its substituents.

Kernel theory allows to define a kernel from a linear combination of sub kernels. Considering kernels based on bags of patterns, each sub kernel may be defined as a kernel encoding a molecular similarity according to a particular pattern. From this point of view, multiple kernel learning methods allow to compute an optimal weight for each sub kernel according to a property to predict. This weight corresponds to the contribution of each pattern to kernel computation and may be understood as a measure of the influence of each pattern. On one hand, this weighting step allows to increase accuracy of prediction models by removing irrelevant patterns from the kernel computation. On the other hand, patterns corresponding to high weights may be seen as pharmacophores. These pharmacophores obtained without *a priori* chemical knowledge may be analyzed by chemical experts to understand some molecular behaviors.

In conclusion, graph kernels provide an useful framework which may obtain accurate prediction results by combining expressive similarity measures and

powerful machine learning methods.

Bibliography

[1] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[2] C. Berg, J.P.R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*. Applied Mathematical Sciences. Springer-Verlag, 1984.

[3] Claude Berge. *Graphs and hypergraphs*, volume 6. Elsevier, 1976.

[4] K.M. Borgwardt and H. Kriegel. Shortest-Path kernels on graphs. *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 74–81, 2005.

[5] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[6] D. Cherqaoui, D. Villemin, A. Mesbah, J. M. Cense, and V. Kvasnicka. Use of a Neural Network to Determine the Normal Boiling Points of Acyclic Ethers, Peroxides, Acetals and their Sulfur Analogues. *J. Chem. Soc. Faraday Trans.*, 90:2015–2019, 1994a.

[7] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[8] Jon Dattorro. *Convex optimization & Euclidean distance geometry*. Meboo Publishing USA, 2005.

[9] Mukund Deshpande, Michihiro Kuramochi, and George Karypis. Frequent sub-structure-based approaches for classifying chemical compounds. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 35–, Washington, DC, USA, 2003. IEEE Computer Society.

[10] Harris Drucker, Chris JC Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in neural information processing systems*, pages 155–161, 1997.

[11] Aurélien Ducournau. *Hypergraphes: clustering, réduction et marches aléatoires orientées pour la segmentation d'images et de vidéo.* PhD thesis, École Nationale d'Ingénieurs de Saint-Étienne., 2012.

[12] Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345–, June 1962.

[13] Thomas Gärtner, Peter A. Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and the 7th Kernel Workshop*, 2003.

[14] Johann Gasteiger, editor. *Handbook of Chemoinformatics.* Wiley-VCH, 1. edition, 2003.

[15] Benoit Gaüzère, Luc Brun, and Didier Villemin. Two New Graphs Kernels in Chemoinformatics. *Pattern Recognition Letters*, 33(15):2038–2047, 2012.

[16] Benoit Gaüzére, Luc Brun, Didier Villemin, and Myriam Brun. Graph kernels based on relevant patterns and cycle information for chemoinformatics. In *Proceedings of ICPR 2012*, pages 1775–1778. IAPR, IEEE, November 2012.

[17] Benot Gazre, Luc Brun, and Didier Villemin. Relevant cycle hypergraph representation for molecules. In WalterG. Kropatsch, NicoleM. Artner, Yll Haxhimusa, and Xiaoyi Jiang, editors, *Graph-Based Representations in Pattern Recognition*, volume 7877 of *Lecture Notes in Computer Science*, pages 111–120. Springer Berlin Heidelberg, 2013.

[18] Jaume Gibert, Ernest Valveny, and Horst Bunke. Graph embedding in vector spaces by node attribute statistics. *Pattern Recognition*, 45(9):3072–3083, 2012.

[19] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[20] Tamás Horváth. Cyclic pattern kernels revisited. In Springer-Verlag, editor, *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, volume 3518, pages 791 – 801, 2005.

[21] Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*, page 158. ACM Press, 2004.

[22] Salim Jouili and Salvatore Tabbone. Graph embedding using constant shift embedding. In *Proceedings of the 20th International conference on Recognizing patterns in signals, speech, images, and videos*, ICPR'10, pages 83–92, Berlin, Heidelberg, 2010. Springer-Verlag.

[23] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized Kernels Between Labeled Graphs. *Machine Learning*, 2003.

[24] Bin Luo, Richard C. Wilson, and Edwin R. Hancock. Spectral embedding of graphs. *Pattern Recognition*, 36(10):2213 – 2230, 2003.

[25] Muhammad Muzzamil Luqman, Jean-Yves Ramel, Josep Lladós, and Thierry Brouard. Fuzzy multilevel graph embedding. *Pattern Recognition*, 46(2):551–565, 2013.

[26] P. Mahé and J.-P. Vert. Graph kernels based on tree patterns for molecules. *Machine Learning*, 75(1):3–35, October 2008.

[27] Pierre Mahé, Liva Ralaivola, Véronique Stoven, and Jean-Philippe Vert. The pharmacophore kernel for virtual screening with support vector machines. *Journal of chemical information and modeling*, 46(5):2003–14, 2006.

[28] Pierre Mahé, Nobuhisa Ueda, Tatsuya Akutsu, Jean-Luc Perret, and Jean-Philippe Vert. Extensions of marginalized graph kernels. In *Twenty-first international conference on Machine learning - ICML '04*, page 70. ACM Press, 2004.

[29] Pierre Mahé and Jean-philippe Vert. Graph kernels based on tree patterns for molecules. *Machine Learning*, 75(1)(September 2008):3–35, 2009.

[30] J. Mercer. Functions of Positive and Negative Type, and Their Connection with the Theory of Integral Equations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 83(559):69–70, November 1909.

[31] H. L. Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *J. of Chem. Doc.*, 5(2):107–113, 1965.

[32] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[33] Michel Neuhaus and Horst Bunke. *Bridging the Gap Between Graph Edit Distance and Kernel Machines*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2007.

[34] G. Poezevara, B. Cuissart, and B. Crémilleux. Discovering emerging graph patterns from chemicals. In *Proceedings of the 18th International Symposium on Methodologies for Intelligent Systems (ISMIS 2009)*, pages 45–55, Prague, 2009. LNCS.

[35] Alain Rakotomamonjy, Francis Bach, Stephane Canu, and Yves Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

[36] Liva Ralaivola, Sanjay J Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural networks : the official journal of the International Neural Network Society*, 18(8):1093–110, October 2005.

[37] Jan Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In *First International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74. Citeseer, 2003.

[38] K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(7):950–959, 2009.

[39] Kaspar Riesen. *Classification and Clustering of Vector Space Embedded Graphs*. PhD thesis, Institut fr Informatik und angewandte Mathematik Universitt Bern, 2009.

[40] Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, SSPR & SPR '08, pages 287–297, Berlin, Heidelberg, 2008. Springer-Verlag.

[41] Nino Shervashidze and Karsten M Borgwardt. Fast subtree kernels on graphs. In *Advances in Neural Information Processing Systems*, pages 1660–1668, 2009.

[42] Nino Shervashidze, Kurt Mehlhorn, S.V.N. Vishwanathan, Tobias Petri, and Karsten M. Borgwardt. Efficient Graphlet Kernels for Large Graph Comparison. In *ReCALL*, 2009.

[43] Nino Shervaszide. *Scalable Graph Kernels*. PhD thesis, Universitt Tbingen, 2012.

[44] Nicolas Sidere, Pierre Héroux, and Jean-Yves Ramel. Vector representation of graphs: Application to the classification of symbols and letters. In *ICDAR*, pages 681–685. IEEE Computer Society, 2009.

[45] R. Todeschini and V. Consonni. *Molecular Descriptors for Chemoinformatics*, volume 41. WILEY-VCH, Weinheim (Allemagne), 2009.

[46] Hannu Toivonen, Ashwin Srinivasan, Ross D. King, Stefan Kramer, and Christoph Helma. Statistical evaluation of the predictive toxicology challenge 20002001. *Bioinformatics*, 19(10):1183–1193, 2003.

[47] V Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[48] Manik Varma and Debajyoti Ray. Learning the discriminative power-invariance trade-off. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[49] S.V.N. Vishwanathan, K.M. Borgwardt, and N.N. Schraudolph. Fast computation of graph kernels. *Advances in Neural Information Processing Systems*, 19:1449, 2007.

[50] P. Vismara. Union of all the minimum cycle bases of a graph. *The Electronic Journal of Combinatorics*, 4(1):73–87, 1997.

[51] Nikil Wale, IanA. Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

[52] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, ICDM '02, pages 721–, Washington, DC, USA, 2002. IEEE Computer Society.

[53] Florian Yger and Alain Rakotomamonjy. Wavelet kernel learning. *Pattern Recognition*, 44(10):2614–2629, 2011.

Index

connectivity, 5
cycle, 4

degr, 4

graph, 4
graphe non orient, 4

hypergraphe, 6
hypergraphe orient, 6

isomorphism, 5

labeled graph, 5

Matrice de Gram, 16

path, 4

sub graph, 4