Graph Kernels for Solving Chemical Problems in Machine Learning

Byan Sakura Kireyna Aji 13518066
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518066@std.stei.itb.ac.id

Machine learning has taken its role in increasing the availability of repositories that answers a lot of disputations that are presented by the study of chemistry such as predicting mutagenicity, toxicity, and anti-cancer activity on three publicly available data sets. In this area, machine learning must be capable to define graphs that link covalent bonds. This is where graph kernels takes its part to process such connections in various sizes and structures. Trials on graphs from chemical informatics show that these techniques are able to fasten computation by an order of magnitude or more. Kernels method is known not only for its accuracy and comparability throughout trials of datasets but also ability to speed up the process of computation.

***Keywords*—artificial intelligence, chemical informatics, graph kernels, machine learning.**

I. INTRODUCTION

Computing chemical data used to involve many different tasks like clustering, regression, classification, or ranking, most of them are related to Structure-Activity Relationship (SAR) analysis, that is, finding a relationship between the structures of molecules and their activity. The term activity in this area refers to a particular biological property the molecules exhibit, such as their ability to bind to a particular biological target, their toxicity properties, or their Absorption, Distribution, Metabolism, and Excretion properties.

Chemical problems that have been described as above learning often require the involvement of variable-sized structured data such as strings and orders, trees, and graphs that are supported by machine learning. These data, especially graphs, have their role on solving retrieval of documents, sequences of protein like DNA and RNA, and molecular structures. Machine learning helps in the presentation of such data in a structural manner to ease the extraction to find meaning, patterns, and regularities.

To see on the broader spectrum, machine learning methods have been utilized to process molecular data problems. Some of those methods are differentiated as inductive logic programming, genetic algorithm, graphical models, recursive neural networks, and kernel methods. This paper is going to pivot on the application of graph and the evolvement of kernel methods to define the role of informatics in the prediction of the toxicity and activity of chemical compounds.

The graphical model approach is a probabilistic approach where random variables are associated with the nodes of a graph and in places the intertwinement of the graph is related to Markovian independence assumptions between variables. The graph here typically consists input nodes to reflect the structure of input data, hidden nodes that are associated with masked dynamics and context propagation, and output nodes linking to classification or regression tasks. The parameter of graphical modeled are supported by local conditional distributions of a node variable given its neighbor variable. These come in the form of translation-invariance assumptions in regularly structured graph such as linear chains, bounded degree trees, and lattices.

Later, kernel methods came up as a formidable class of machine learning methods that are suitable for variable-size structured data. The fundamental idea of kernel methods is to construct a kernel based on input objects given to measure the similarity between them. This kernel can be seen as the inner product of the form $k(\mathbf{u}, \mathbf{v}) = \varphi(\mathbf{u}), \varphi(\mathbf{v})$ in an embedding feature space determined by the map.

Fig. 1. Kernel method for comparison [1]

Convex methods based on inner products that are computed via the kernel embedding space can tackle tasks such as regression and classification.

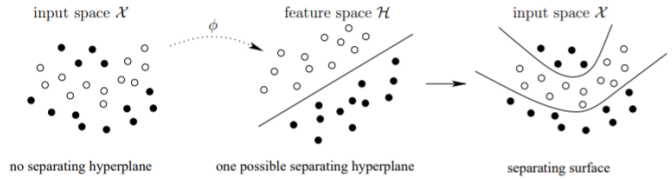


Fig. 2. The kernel approach for classification [1]

Thus, we can make use of kernel methods to answer problems like the prediction of toxicity, mutagenicity, and cancer rescue activity.

II. THEORY

2.1 Kernel Methods

Within machine learning, kernel methods are a class of algorithms in analyzing pattern to find and study general types of relations like clusters, rankings, principal components, correlations, and classifications in dataset. In contrast with another algorithm that also have means to solve such tasks, kernel methods only need a user-specified kernel over pairs of data points in raw representation.

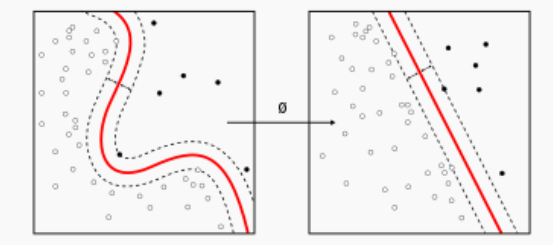


Fig. 3. Illustration of kernel methods. [2]

Importantly, kernel methods handle non-linear complex tasks with linear methods in a new space. To fix the ideas, consider a classification problem with training set S = $\{(\boldsymbol{u}_1, y_1)\ldots (\boldsymbol{u}_l, y_l)\}$, $(\boldsymbol{u}_i, y_i) \in$ X ×Y, i = 1..., where X is an inner-product space (e.g. Rd) with inner product denoted by ·, ·, and Y = {−1, +1}. In this setting, learning is the task of building a function $f \in$ YX from the training set S associating a class $y \in$ Y to a pattern $\boldsymbol{u} \in$ X such that the generalization error of f is as low as possible.

The form f can be simplified by the hyper plane $f(\boldsymbol{u})$ = sign ($<\boldsymbol{w}, \boldsymbol{u}>$ + b), where sign (·) is the function returning the sign of its argument. The decision function f outputs a prediction depending on which side of the hyper plane $\boldsymbol{w}$, $\boldsymbol{u}$ + b = 0 the input pattern u lies. Under reasonable assumptions discussed in the references, solving for the "best" hyper plane leads to a convex quadratic optimization problem such that the solution vector $\boldsymbol{w}$ is a (usually sparse) linear combination of the training vectors: w = $\sum_{i=1}^{l} a_i y_i \boldsymbol{u_i}$ for some $a_i \in$ R+ i, = 1,...

$$f(\boldsymbol{u}) = \text{sign} \sum_{i=1}^{l} a_i y_i < \boldsymbol{u_i}, \boldsymbol{u} > + b \quad (1)$$

Nevertheless, in complex classification problems, the set of all possible linear decision surfaces may not be lavish enough to supply adequate classification, no matter what the values of the parameters $\boldsymbol{w} \in$ X and b $\in \mathbb{R}$ are. The purpose of the kernel trick is precisely to overcome such limitation by applying a linear approach to the transformed data $\varphi(\boldsymbol{u_i}),\ldots,\varphi(\boldsymbol{u_l})$ rather than the raw data. Here φ denotes an embedding function from the input space X to a feature space H, equipped with a dot product. Using the previous theorem, the separating function must now be of the form:

$$f(\boldsymbol{u}) = \text{sign} \sum_{i=1}^{l} a_i y_i < \varphi(\boldsymbol{u_i}), \varphi(\boldsymbol{u}) > + b \quad (2)$$

The main point of kernels approach is to change the dot product in feature space with a kernel $(\boldsymbol{u}, \boldsymbol{i}) = < \varphi(\boldsymbol{u}), \varphi(\boldsymbol{v}) >$ utilizing the definition of positive definite kernels, Gram matrices, and Mercer's theorem.

Positive definite kernel:

Let X be a nonempty space. Let $k \in \mathbb{R}$ X ×X be a continuous and symmetric function. k is a positive definite kernel if and only if for all ∈ N, for all $\boldsymbol{u_1},\ldots, \boldsymbol{u_l} \in \mathbb{R}$, the square × matrix $K = (k(\boldsymbol{u_i}, \boldsymbol{u_j}))_{i \le j \le l}$ is positive semi-definite. For a given set S_u = $\{\boldsymbol{u_1},\ldots, \boldsymbol{u_l}\}$, K is called the Gram matrix of k with respect to S_u. Positive definite kernels are also referred to as Mercer kernels.

Mercer's property:

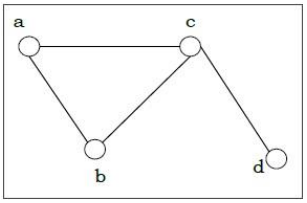
For any positive definite kernel function k $\mathbb{R} \in$ X ×X , there exists a mapping $\varphi \in \boldsymbol{H}^x$ into the feature space $\boldsymbol{H}$ equipped with the inner product ·, ·$\boldsymbol{H}$, such that:

$$\forall \boldsymbol{u}, \boldsymbol{v} \in \text{X } k(\boldsymbol{u}, \boldsymbol{v}) = <\varphi(\boldsymbol{u}), \varphi(\boldsymbol{v})>_H$$

To summarize, the application of kernel methods need two independent modules such as module to compute the kernel and a module to compute the optimal manifold in feature space. The construction of efficient kernels in chemistry is an essential step to address the arduous problem like classifying compounds according to their properties. An efficient kernels for molecular structures can be represented by labeled and undirected graph of bonds with labels assigned to both nodes and edges.

2.2. Graph

A graph G consists of an ordered set of n vertices V = $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$, and a set of directed edges $E \subset V \times V$.



Vertex	Degree	Even / Odd
a	2	even
b	2	even
c	3	odd
d	1	odd

Fig. 4. Example of graph. [4]

Each vertex has its own degree, which can be defined as the number of edges that meet with the vertex V. Degree of a vertex can be even or odd, depending on the amount of edges that it links. If the degree of a vertex is even, the vertex is called an even vertex. So goes the other way around.

The degree of a graph is the largest degree owned by the vertices in that graph. In figure 4, the degree of the graph is represented by 3.

In a graph, the sum of all the degrees of all vertices shall be equal to twice the number of edges. This is referring to the handshaking lemma theorem.

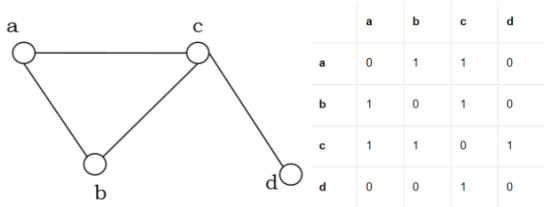
A vertex v_i, is said to be a neighbor of another vertex v_i, if they are connected by an edge, that is, if $(\boldsymbol{v_i}, \boldsymbol{v_j}) \in E$; this is also denoted $\boldsymbol{v_i} \sim \boldsymbol{v_j}$. This concept does not allow self-loops, that is, $(\boldsymbol{v_i}, \boldsymbol{v_i}) \notin E$ for any i. A walk of length k on G is a sequence of indices $i_0, i_1, \ldots, i_k$ such that $\boldsymbol{v}_{ir-1} \sim \boldsymbol{v}_{ir}$ for all 1 ≤ r ≤ k.

To show the connection between the nodes, an adjacency matrix is introduced to the game. It has a normalized cousin, defined $A{:=}\bar{A}D^{-1}$, with the property that each of its columns sums to one, and it can therefore serve as the transition matrix for a stochastic process. Here, D is a diagonal matrix of node

degrees, that is, D_{ij}= d_i =$\sum_j \bar{A}_{ij}$. A random walk on G is a process generating sequences of vertices $v_1, v_2, \dots, v_n$ according to $\mathbb{P}(i_{k+1}|i_1 \dots i_{k+1})$ =A_{ik+1jk}, that is, the probability at vik of picking $\boldsymbol{v}_{ik+1}$ next is proportional to the weight of the edge $(\boldsymbol{v}_{ik}, \boldsymbol{v}_{ik+1})$. The t^{th} power of *A* thus describes *t*-length walks, that is, $A^t{}_{ij}$ is the probability of a transition from vertex v_j to vertex v_i via a walk of length *t*. If p_0 is an initial probability distribution over vertices, then the probability distribution p_t describing the location of our random walker at time *t* is p_t = $A^t p_0$. The j^{th} component of p_t denotes the probability of finishing a *t*-length walk at vertex v_j .

The adjacency matrix *A* of *G* is defined as

$$[A]ij = \begin{cases} 1 \text{ if } (v_i, v_j) \in E \\ 0 \end{cases}$$



	a	b	c	d
a	0	1	1	0
b	1	0	1	0
c	1	1	0	1
d	0	0	1	0

Fig. 5. The Adjacency Matrix of a Graph [4]

Graphs have a lot of properties that can differentiate themselves into several types such as:

2.2.1. Null Graph

This kind of graph has no edges. It is denoted by $\boldsymbol{N}_n$.

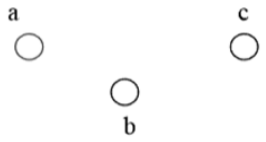


Fig. 6. Null Graph [4]

2.2.2. Simple Graph

A graph is considered as a simple graph if he graph is neither directed nor containing any loops or multiple edges.

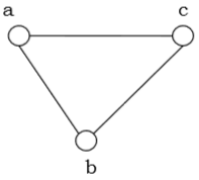


Fig. 7. Simple Graph [4]

2.2..3. Multi-Graph

When several edges between the same set of vertices are allowed, it is called Multi-graph.

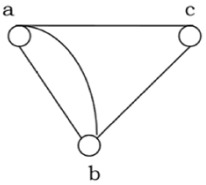


Fig. 8. Multi-Graph [4]

2.2.4. Connected and Disconnected Graph

A graph is connected if any two vertices of the graph are linked by a path; while a graph is disconnected if at least two vertices of the graph are not linked by a path. If a graph *G* is disconnected, then every maximal connected sub-graph of GG is called a connected component of the graph GG.

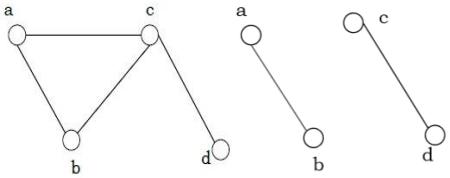


Fig. 9. Connected and Disconnected Graph [4]

2.2.5. Regular Graph

Regular graph is a graph which vertices have the same degree. In a regular graph *G* of degree *rr*, the degree of each vertex of *GG* is *r*.

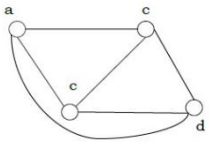


Fig. 10. Regular Graph [4]

2.2.6. Complete Graph

A complete graph is a graph with every two vertices pair are connected by exactly one edge. The complete graph with *n* vertices is denoted by *KnKn*.

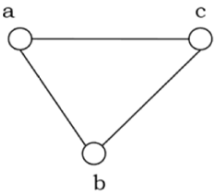


Fig. 11. Complete Graph [4]

2.2.7. Cycle Graph

Cycle graph happens when a graph consists of a single cycle, The cycle graph with *n* vertices is denoted by *CnCn*.

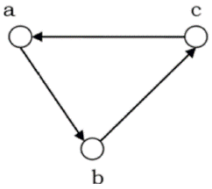


Fig. 12. Cycle Graph [4]

2.3. Tree

Tree is a connected graph with no cycle. There exist a different path for each pair of vertices in graph *G*. A tree with *N* number of vertices has (*N*−1) number of edges. The vertex which is of 0 degree is called root of the tree. The vertex which is of 1 degree is called leaf node of the tree and the degree of an internal node is at least 2.

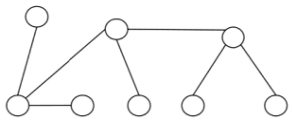


Fig. 13. Tree [4]

2.3.1. Labeled Trees

Labeled tree is basically a tree whose vertices are signed unique numbers from 1 to *n* It can counted as such trees for small values of *n* by hand so as to conjecture a general formula. The number of labeled trees of *n* number of vertices is $n-2$. Two labeled trees are isomorphic if their graphs are isomorphic and the corresponding points of the two trees have the same labels.

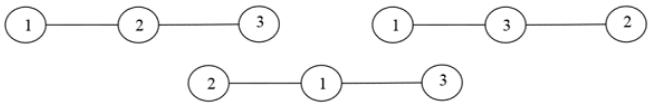


Fig. 14. .Labeled Trees [4]

2.3.2. Unlabeled Trees

An unlabeled tree is a tree the vertices of which are not assigned any numbers. The number of labeled trees of *n* number of vertices is $(2n)!(n+1)!n!$ with n^{th} as Catalan number.

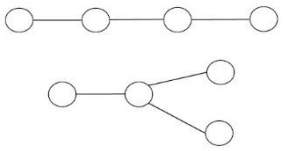


Fig. 15. Unlabeled. Trees [4]

2.3.2. Rooted Tree

Rooted tree *GG* is a connected acyclic graph with a special node that is called the root of the tree and every edge directly or indirectly originates from the root. An ordered rooted tree is a rooted tree where the children of each internal vertex are ordered. When every internal vertex of a rooted tree has not more than *m* children, it is called an *m*-ary tree. If every internal vertex of a rooted tree has exactly *m* children, it is called a full *m*-ary tree. If *m*=2, the rooted tree is called a binary tree.

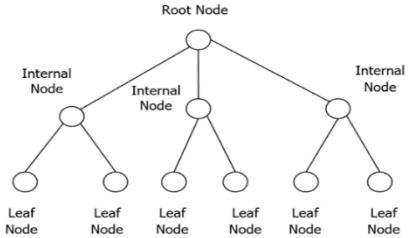


Fig. 15. Labeled Tree [4]

3.3. Graph Kernels

Graph kernel is a kernel function that can compute an inner product on graphs. It can be intuitively understood as functions measuring pairs of graph's similarity.

3.3.1. Marginalized Graph Kernels

Marginalized kernels define a global similarity measure by means of a simpler one expressed on auxiliary variables introduced in the problem. In our case, these latent variables consist of substructures of the graphs, and more precisely they are paths, which are easier to handle than sub-graphs. In labeled graphs, label sequences are associated with the paths of the graph, and their similarity is assessed by a string kernel. Moreover, paths are here considered as random walks, so that probability distributions are associated with the set of paths of the graphs. The kernel between two graphs is then defined as the expectation of the pairwise paths similarity, according to their probability distributions.

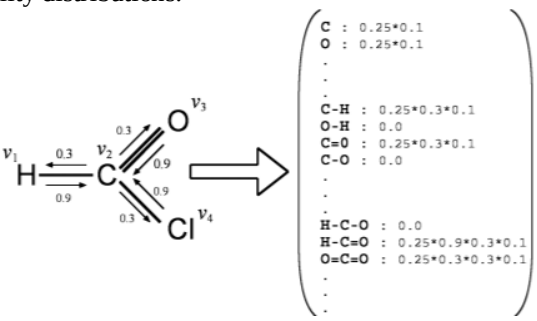


Fig.15. Marginalized Graph Kernels Application in Chemical Compound [5]

3.3.2 Type of Kernels

.Graph Kernels later is differentiated into classes according to the existence of labeled pairs or labels sequences.

3.3.2.1. Labeled Pairs Kernels

For a graph *G*, let the vertex-label matrix *L* be the $p \times n$ matrix such that its (r, i)-coefficient L_{ri} is equal to 1 if and only if the label of vertex i is equal to l_r^a.

$$k(G_1, G_2) = L_1 \sum_{i=0}^{\infty} \lambda_i E_1^i L_1', L_2 \sum_{i=0}^{\infty} \lambda_i E_2^i L_2' \quad (3)$$

where *M'* denotes the transpose of *M*. The inner product $\cdot, \cdot$ is the Frobenius matrix product. The Frobenius product of two $m \times n$ matrices *A* and *B* is defined by the trace tr $(A*B)$, where $A *= \bar{A}'$ is the conjugate transpose of *A*. These kernels are called labeled-pair kernels because, given two graphs G_1 and G_2, they count the number of walks in G_1 and G_2, of the same length and with the same labels on their first and last nodes.

(1) exponential kernel:

$$\sum_{i=0}^{\infty} \lambda_i E^i = \sum_{i=0}^{\infty} \frac{(\Upsilon E)^i}{i!} \quad (4)$$

(2) truncated power series kernel:

$$\sum_{i=0}^{\infty} \lambda_i E^i = \sum_{i=0}^{\infty} (\Upsilon E)^i \quad (5)$$

(3) convergent geometric kernel:

$$\sum_{i=0}^{\infty} \lambda_i E^i = \sum_{i=0}^{\infty} (1 - \Upsilon E)^{-1} \quad (6)$$

3.3.2.2. Sequences of Labels Kernels

In counting shared label sequences, product graph kernels use a procedure that is based on the direct graph product of two graphs.

Let $G_1 = (V_1, \varepsilon_1)$ and $G_2 = (V_2, \varepsilon_2)$ be two vertex- and edge-labeled graphs. Given those two graph G_1 and G_2, the graph product kernel k_x is defined as:

$$k_x(G_1, G_2) = \sum_{i\ j-1}^{|V_x|} \left[\sum_{n=0}^{\infty} \lambda_n E_x^n \right]_{ij} \tag{7}$$

III. CHEMICAL KERNELS

There is some limitations on how far kernels can go solving chemical problems like address problems of molecular classification. In order to evade such complication, there found a technique of *molecular fingerprinting* in 1998 by Flower and 2001 by Raymond and Willett.

This kind of kernel approach can be computed efficiently and leverage the peculiar properties of tiny-molecules graphs in organic chemistry. Particularly, these graphs are pretty small in either number of vertices or the number of edges and they are very constrained by the laws of chemistry.

3.1. Molecular Fingerprinting

In the past, fingerprints are bit-vectors of a given size l, typically taken in the range 100-1000. With a molecule *M* having *n* atoms and *m* bonds, constructing a corresponding fingerprint requires starting *depth-first-search* explorations in each atom in the molecule. Thus such substructures being considered are labeled paths, which may include labeled cycles.

In this bit vectors, the maximal path length *d* is set to $+\infty$ which means that if we want to extract all the paths starting from all the atoms of a molecule, the complexity of the procedure in only 0(*nm*) if we do not allow path emanating from a vertex to share edges once they have separated.

However, in its actual application, the value of *d* is often set low. Because fingerprinting is commonly used by chemists, the values for *l*, *b*, and *d* are often already set along, if needed, with extra information like irrelevant paths that should be discarded. To add in the last, an expanding number of chemical databases already include some kind of fingerprint field in their tables, although standards have not yet been set.

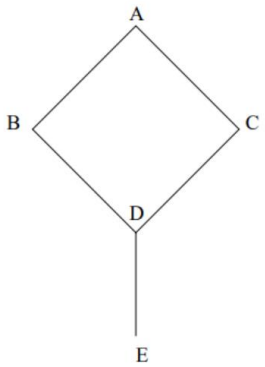


Fig. 14. Simple graph to define various options in dfs extraction [1]

3.2. Depth First Search

In a tree with the root *A*, depth first search exploration of depth *d* yields a list of all the paths of length *d* emanating from that node. Since molecular graph has no cycle, there may be different implementations of the depth- first search leading to different set of paths. Variations emerge depending on whether cycles are allowed in a given oath and whether two different paths from the some node are allowed to share any edges after the first point of divergence from each other. Here goes the example of DFS

(1) DFS with no cycles:
- A
- A-B
- A-B-D
- A-B-D-C
- A-B-D-E
- A-

(2) DFS with cycles:
- A
- A-B
- A-B-D
- A-B-D-C
- A-B-D-C-A
- A-B-D-E

(3) DFS with all paths and no cycles:
- A
- A-B
- A-B-D
- A-B-D-C
- A-B-D-E
- A-C
- A-C-D
- A-C-D-B
- A-C-D-E

(4) DFS with all paths and with cycles:
- A
- A-B
- A-B-D
- A-B-D-C
- A-B-D-C-A
- A-B-D-E
- A-C
- A-C-D
- A-C-D-B
- A-C-D-B-A
- A-C-D-E

For molecules with *n* atoms and *m* edges, the complexity in extraction of all path is *O*(*nm*) for the first and the second cases. In third and fourth cases with the length up to *d*, the complexity is at most $O(n\alpha^d)$, with α as branching factor. Since typical degree of graphs in organic chemistry is often only slightly above two, the branching factor is probably just slightly above one.

3.3. Generalized Fingerprints

Long binary feature vectors with a unique bit position

reserved for each path can be utilized to evade the loss of information associated with clashes. Seen like this, a molecule can be viewed as a text document with all the labeled paths of length up to *d* that can be retrieved by depth-first searches. However, some trials using the TF-IDF approach, did not produce any significant improvements and therefore it is not used here. An alternative to the TF-IDF scheme, originated also in the field of information retrieval, is to consider a reduced set of paths selected according to the mutual information criteria to keep the paths that carry the most important information for a given classification.

3.4. Fingerprint Similarity

Another kernels can be used to define the similarity of fingerprint, such as:

$$k_d(\boldsymbol{u},\boldsymbol{v}) = <\boldsymbol{u},\boldsymbol{v}>_{\boldsymbol{d}} = \sum_{\boldsymbol{path}\in\mathcal{P}(\boldsymbol{d})} \varnothing_{\boldsymbol{path}}(\boldsymbol{u})\,\varnothing_{\boldsymbol{path}}(\boldsymbol{v}) \tag{7}$$

This form of kernel can later be developed as another forms such as:

Tanimoto kernel:

Let $\boldsymbol{u}$, $\boldsymbol{v}$ denote two molecules and *d* be an integer. Consider the feature map φ_d and the corresponding kernel k_d.

$$k_d^t(\boldsymbol{u},\boldsymbol{v}) = \frac{k_d(\boldsymbol{u},\boldsymbol{v})}{k_d(\boldsymbol{u},\boldsymbol{u}) + k_d(v,\boldsymbol{v}) - k_d(\boldsymbol{u},\boldsymbol{v})} \tag{8}$$

MinMax kernel:

Let $\boldsymbol{u}$, $\boldsymbol{v}$ denote two molecules and *d* be an integer. Consider the feature map $\varphi_d(\cdot)$, and the corresponding $\varphi_{path}(\cdot)$.

$$k_d^m(\boldsymbol{u},\boldsymbol{v}) = \frac{\sum_{\boldsymbol{path}\in\mathcal{P}(\boldsymbol{d})} \min(\varphi_{path}(\boldsymbol{u}), \varphi_{path}(\boldsymbol{v}))}{\sum_{\boldsymbol{path}\in\mathcal{P}(\boldsymbol{d})} \max(\varphi_{path}(\boldsymbol{u}), \varphi_{path}(\boldsymbol{v}))} \tag{9}$$

Hybrid kernel:

Hybrid kernel is a convex combination of two kernels, respectively measuring the number of common paths and common missing-paths between two molecules.

$$k_d^t(\boldsymbol{u},\boldsymbol{v}) = \frac{2\,k_d(\boldsymbol{u},\boldsymbol{v})}{k_d(\boldsymbol{u},\boldsymbol{u}) + k_d(v,\boldsymbol{v})} \tag{10}$$

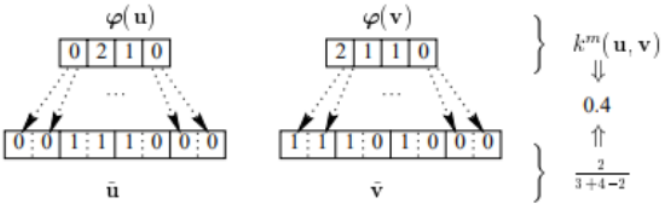


Fig. 15. Connection between Tanimoto and MinMax kernels [1]

3.5. Algorithm

In the combination of those kernels and the Voted Perceptron learning algorithm, Freund and Schapire proposed an algorithm in 1999 as an efficient learning method with the main idea of perceptron.

$p \leftarrow 0,\ \boldsymbol{\alpha}_1 \leftarrow \mathbf{0},\ c_1 \leftarrow 0$

repeat *until convergence of the training error*

- **for** $i = 1, \ldots, A$
  - *compute prediction* $\hat{y} \leftarrow sign\left(\sum_{j=1}^{A}\right)$
  - **if** $\hat{y} = y_i$
    - $c_p \leftarrow c_p + 1$
  - **else**
    - $\boldsymbol{\alpha}_{p+1} \leftarrow \boldsymbol{\alpha}_p$
    - $\alpha_{p+1,i} \leftarrow \alpha_{pi} + y_i$
    - $c_{p+1} \leftarrow 1$
    - $p \leftarrow p + 1$
  - **end if**
- **end for**

end repeat

IV. APPLICATION

After all those kernels discussed above are being put together with the algorithm, the answers for predicting mutagenicity, toxicity, and anti-cancer activity can be found in three different datasets.

4.1. Mutagenicity

The Mutagenicity dataset originally consists 230 chemical compounds assayed in *Salmonella typhimurium.* Among all those compounds, only 188 are considered to be learnable. The result from other groups that are reported for comparison purposes were obtained also on the same subset of 188 molecules.

4.2. Toxicity

The Predictive Toxicology Challenge (PTC) dataset [Helma et al., 2001] reports the carcinogenicity of several hundred chemical compounds for Male Mice (MM), Female Mice (FM), Male Rats (MR) and Female Rats (FR)

	Mutag	MM	FM	MR	FR
#pos.	125 (66.5%)	129 (38.4%)	143 (41.0%)	152 (44.2%)	121 (34.5%)
#neg.	63 (33.5%)	207 (61.6%)	206 (59.0%)	192 (55.8%)	230 (65.5%)
total ex.	188	336	349	344	351
Avg. #atoms/mol.	17.93	25.05	25.25	25.56	26.08
Avg. #bonds/mol.	19.79	25.39	25.62	25.96	26.53
Avg. degree	2.21	2.03	2.03	2.03	2.03

Table I: Mutagenicity and Toxicity Dataset [5]

4.3. Cancer

The dataset on cancer screening result was put on public by the National Cancer Institute. It provides screening result for the ability roughly 70,000 compounds to kill the growth of a

panel of 60 human tumor cell lines. For each cell line, roughly 3,500 compounds, described by their 2D structures, are supplied with information on their anti-tumor activity.

Not only is the NCI dataset considerably larger than the Mutagenicity and Toxicity datasets, but overall it is also more balanced. Thus the trivial background statistical predictor always predicting the class encountered more frequently has poorer performance on the NCI dataset.

Screen	#pos.	#neg.	Screen	#pos.	#neg.
786-0	1832 (52.3%)	1674 (47.7%)	NCI-H226	1781 (51.4%)	1683 (48.6%)
A498	1782 (51.2%)	1698 (48.8%)	NCI-H23	1968 (52.9%)	1751 (47.1%)
A549	1901 (50.9%)	1833 (49.1%)	NCI-H322M	1765 (47.8%)	1925 (52.2%)
ACHN	1795 (50.8%)	1736 (49.2%)	NCI-H460	2049 (56.9%)	1550 (43.1%)
BT-549	1399 (50.4%)	1379 (49.6%)	NCI-H522	2138 (59.8%)	1435 (40.2%)
CAKI-1	1865 (52.1%)	1715 (47.9%)	OVCAR-3	2001 (54.2%)	1690 (45.8%)
CCRF-CEM	2217 (63.7%)	1263 (36.3%)	OVCAR-4	1840 (51.4%)	1742 (48.6%)
COLO-205	1943 (53.3%)	1702 (46.7%)	OVCAR-5	1651 (45.0%)	2019 (55.0%)
DU-145	1416 (48.1%)	1529 (51.9%)	OVCAR-8	1979 (53.3%)	1735 (46.7%)
EKVX	1968 (53.5%)	1713 (46.5%)	PC-3	1522 (51.0%)	1460 (49.0%)
HCC-2998	1804 (56.8%)	1373 (43.2%)	RPMI-8226	2116 (59.4%)	1448 (40.6%)
HCT-116	2049 (55.0%)	1674 (45.0%)	RXF-393	1850 (54.4%)	1551 (45.6%)
HCT-15	1993 (53.4%)	1738 (46.6%)	SF-268	2020 (54.3%)	1701 (45.7%)
HL-60-TB	2188 (64.6%)	1198 (35.4%)	SF-295	2027 (54.1%)	1718 (45.9%)
HOP-62	1888 (52.0%)	1740 (48.0%)	SF-539	1920 (56.7%)	1464 (43.3%)
HOP-92	1982 (56.6%)	1521 (43.4%)	SK-MEL-28	1774 (47.6%)	1950 (52.4%)
HS-578T	1550 (54.0%)	1320 (46.0%)	SK-MEL-2	1783 (49.5%)	1817 (50.5%)
HT29	2004 (54.0%)	1708 (46.0%)	SK-MEL-5	2034 (55.2%)	1651 (44.8%)
IGROV1	1956 (53.0%)	1734 (47.0%)	SK-OV-3	1711 (48.8%)	1792 (51.2%)
K-562	2139 (59.1%)	1479 (40.9%)	SN12C	1918 (52.1%)	1764 (47.9%)
KM12	1941 (52.4%)	1764 (47.6%)	SNB-19	1840 (49.4%)	1885 (50.6%)
LOX-IMVI	2053 (57.0%)	1550 (43.0%)	SNB-75	2131 (61.1%)	1359 (38.9%)
M14	1815 (51.1%)	1736 (48.9%)	SR	1869 (62.2%)	1137 (37.8%)
MALME-3M	1886 (53.8%)	1621 (46.2%)	SW-620	1940 (51.7%)	1813 (48.3%)
MCF7	1733 (57.0%)	1306 (43.0%)	T-47D	1550 (53.3%)	1359 (46.7%)
MDA-MB-231	1475 (50.0%)	1473 (50.0%)	TK-10	1650 (47.3%)	1840 (52.7%)
MDA-MB-435	1519 (51.0%)	1462 (49.0%)	U251	2044 (54.4%)	1711 (45.6%)
MDA-N	1503 (50.7%)	1459 (49.3%)	UACC-257	1873 (50.9%)	1808 (49.1%)
MOLT-4	2175 (61.5%)	1359 (38.5%)	UACC-62	2046 (55.5%)	1638 (44.5%)
NCI-ADR-RES	1586 (51.0%)	1525 (49.0%)	UO-31	1994 (55.2%)	1621 (44.8%)

Table II. NCI Dataset [5]

V. CONCLUSION

Throughout a lot of branch of studies, it is shown that graph has a big role in the development of technology for better future. In this paper, we have already discussed about how the application of graph and kernel methods in machine learning can lead to newfound discoveries of organizing datasets so they can be easily integrated to put into use.

VII. ACKNOWLEDGMENT

To open the acknowledgement I would like to express all of my praises to The Almighty for it is only because of His blessings I could put this task into some work.

I would like to deliver my gratitude toward Dr. Ir. Rinaldi Munir, MT. since only because of him I had such an opportunity to expand the spectrum of my understanding in the matter of graph's application by composing this paper. I would also like to appreciate Mrs. Fariska as the lecturer in my class, one of the main reasons I could understand the concept of graph enough to the point of being able to write this paper.

REFERENCES

[1] Jean-Philippe Vert. The optimal assignment kernel is not positive definite. Technical Report 0801.4061, arXiv, May 2008. http://aps.arxiv.org/abs/0801.4061.
Accessed at 5.54 AM on 11/30/2019

[2] Kernel-Machines Org—community website
Accessed at 9.03 AM on 11/30/2019

[3] Rinaldi Munir, Matematika Diskrit. Bandung: Penerbit INFORMATIKA Bandung, 2010, ch 7

[4] https://www.tutorialspoint.com/discrete_mathematics.htm
Accessed at 8.35 PM 12/1/2019

[5] Helma, R. D. King, S. Kramer, and A. Srinivasan. The predictive toxicology challenge 2000-2001. Bioinformatics, 17(1):107–108, 2001.C.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 30 November 2019

Byan Sakura Kireyna Aji
13518066