Graph kernels for chemical informatics

Hosein Mohimani
GHC7717
hoseinm@andrew.cmu.edu

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Neural Networks 18 (2005) 1093–1110
Neural Networks
www.elsevier.com/locate/neunet

2005 Speical Issue

Graph kernels for chemical informatics

Liva Ralaivola[a,b], Sanjay J. Swamidass[a,b], Hiroto Saigo[a,b], Pierre Baldi[a,b,*]

[a]*School of Information and Computer Sciences, University of California, Irvine, CA 92697-3425, USA*
[b]*Institute for Genomics and Bioinformatics, University of California, Irvine, CA 92697-3425, USA*

Quantitative Structure-Activity relation-ships

- Question. How can we design perfect chemical compounds for a specific biological activity?

- Naïve Solution. Synthesize all the possible chemical compound. Then check the activity of all of them, and select the one with optimal activity

- Problem : There are more than 10^{18} possible chemical compounds

Quantitative Structure-Activity relations-ships

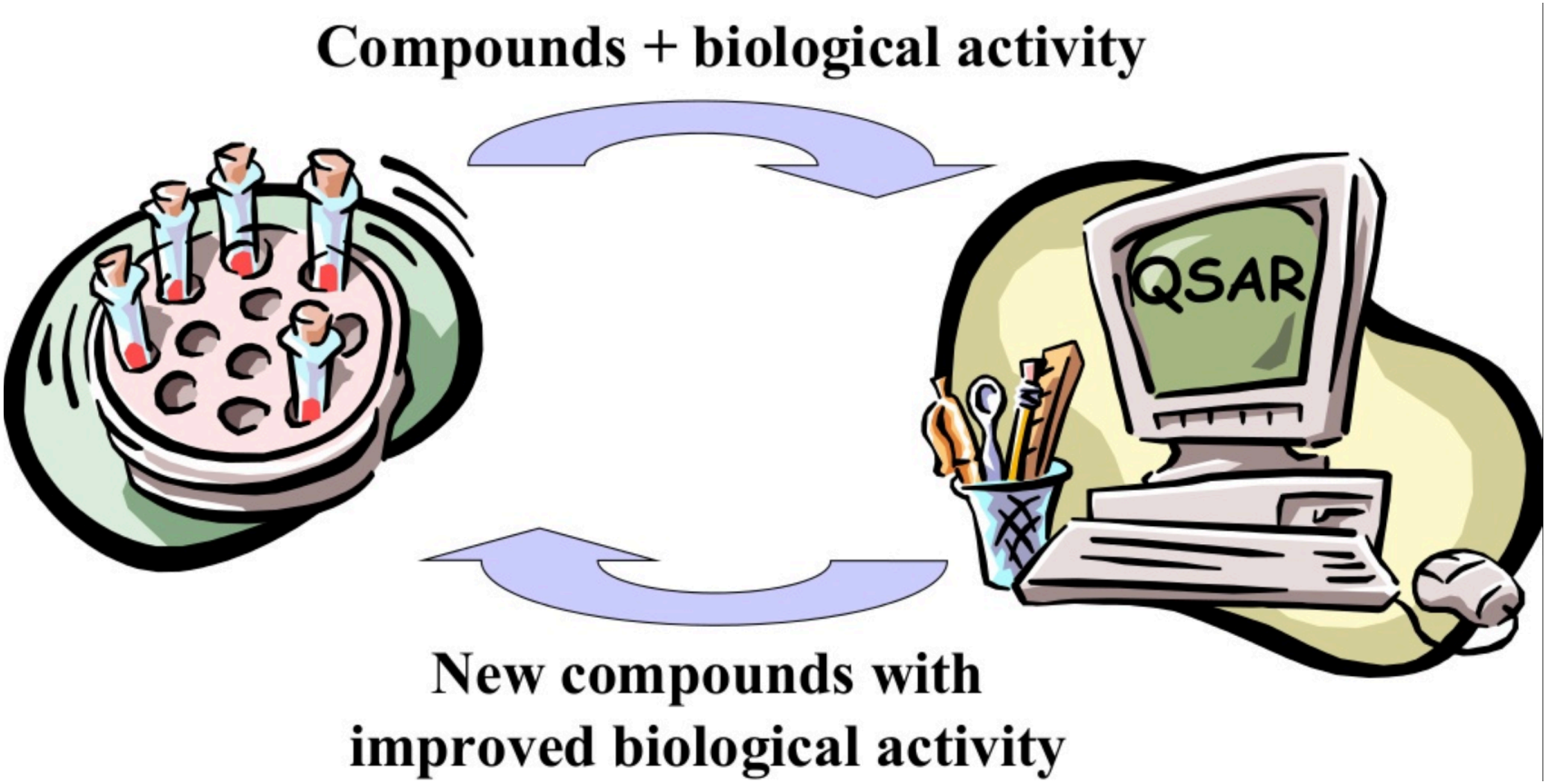
- QSAR : synthesize a small number of compounds (that make sense for target activity) and from their data, learn how to

  - Predict the biological activity of other compounds

  - Predict the structure of optimal compound

Interpolation (predicting results for missing data point from the ones available)

QSAR Feedback loop



QSAR

- QSAR is a mathematical relationship between biological activity of a molecule, and its chemical/geometrical properties

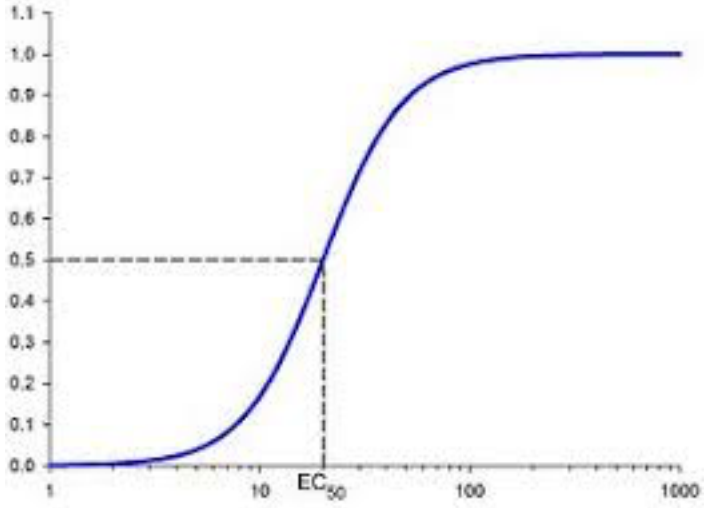
- QSAR attempt to learn consistent relationships between biological activity and molecular properties, so that these rules can be used to evaluate the activity of new compounds

Biological activity

- Example Half Maximal Effective Concentration (EC50)

- EC50 refer to the concentration of a drug which induces a response halfway between baseline (no drug) and maximum (drug so abundant that activity saturates)

- a measure for drug potency



Chemical / Geometrical Properties

- Portion of the molecular structure responsible for specific biological/pharmacological activity
- shape of the molecule
- electrostatic fields

QSAR problem formulation

- Given a set of n properties $f_1, \ldots, f_n$, and a biological activity A,

	A	f_1	f_2	...	f_n
Cmp1	*3.4*	*2.7*	*1.3*	...	*2.2*
Cmp2	*1.3*	*0.5*	*2.8*	...	*1.5*
...					
Cmp'	*?*	*2.4*	*4.1*	...	*3.8*

How can we predict activity for a new compound ?

Its crucial to select relevant properties

QSAR problem formulation

- **Goal :** By learning from a set of

- **Input :** m compounds $Cmp_1, \ldots, Cmp_m$, along with their activities $A_1, \ldots, A_m$ and their properties f_{ij} for $1 \leq i \leq m$ and $1 \leq j \leq n$

- **Output :** for a new compound Cmp' with properties $f'_1, \ldots, f'_n$ predict its activity A'

QSAR techniques : Partial Least Square

- Model activity as a linear combination of features

$$A = C_0 + C_1 f_1 + \ldots + C_n f_n$$

Coefficients are learned by minimizing the prediction error for the training data

Bottleneck of feature-based QSAR

- What are good features ?
- Good Features are difficult to compute
- There is no straightforward approach to compute features from the chemical structure
- Its difficult to find a set of features that cover all activities
- A more natural approach : using atom & bond connectivity

Learning variable size structured data

- Strings
- Sequences
- Trees
- Directed & Undirected graphs

- Texts & Document
- DNA/RNA/Protein sequences
- Evolutionary trees
- Molecular structures

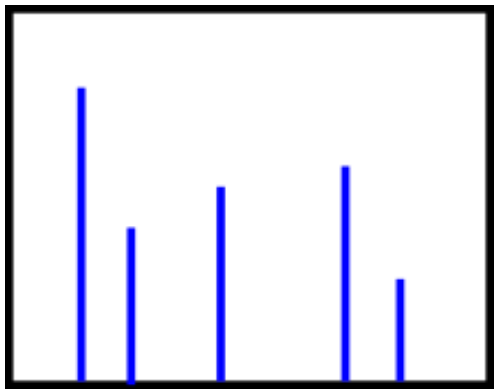
Fix versus variable size data

- Images can be considered fix size data if they are up/down samples to a fixed number of pixels

- Graphs are variable size data (they can have different number of edges / vertices.

Fix versus variable size data

- Mass spectra, in its simplest form, is a variable size data



(2,3,5,7,8)

- If we convert mass spectra to its binary representation (presence/absence of peaks), it becomes fixed size data

(2,3,5,7,8) ⟶ (0,1,1,0,1,0,1,1,0,0)

Learning methods for graph-structured data

(1) Inductive logic programming
(2) Genetic algorithm / Evolutionary methods
(3) Graphical models
(4) Recursive neural networks
(5) Kernel methods

Inductive logic programming

Represent domain & corresponding relationships between data in terms of first order logic

Learn logic theories from data via induction

Ordered search of space of all possible hypothesis and testing them against training data (positive & negative)

Features of Inductive Logic Programming

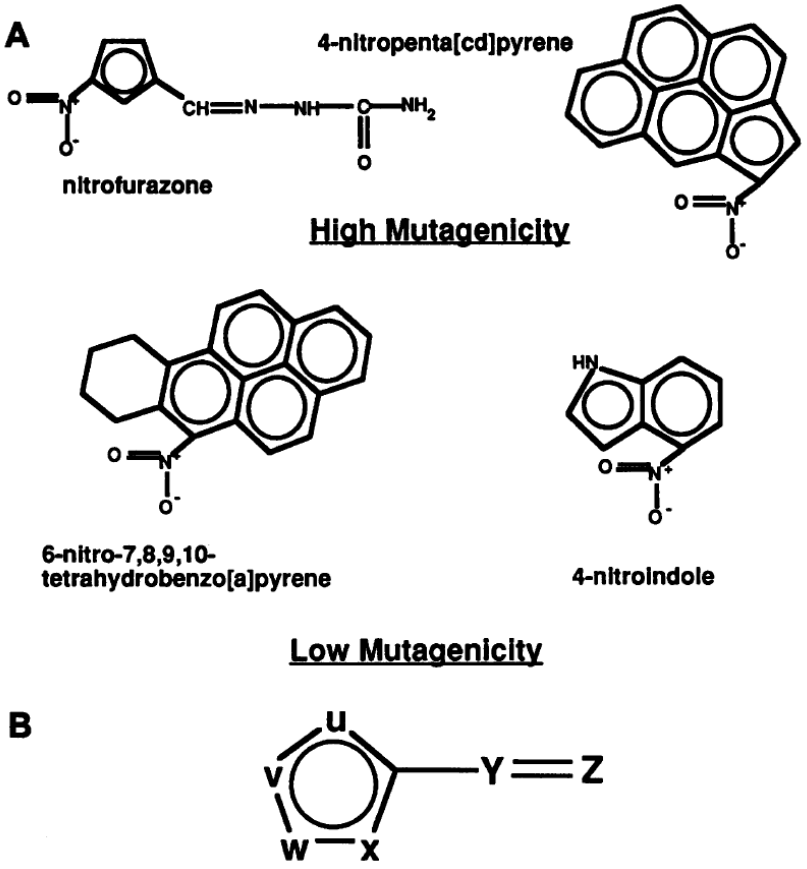
(1) Handles symbolic data in natural way

(2) Background knowledge (e.g. chemical expertize) easily incorporated

(3) Resulting theory & set of rules easy to understand

QSAR Datatset

- 230 compounds
- Ames test : Does a chemical cause mutation in the DNA of a test bacteria ?
- 188 positive
- 42 negative



Inductive Logic Programming Result

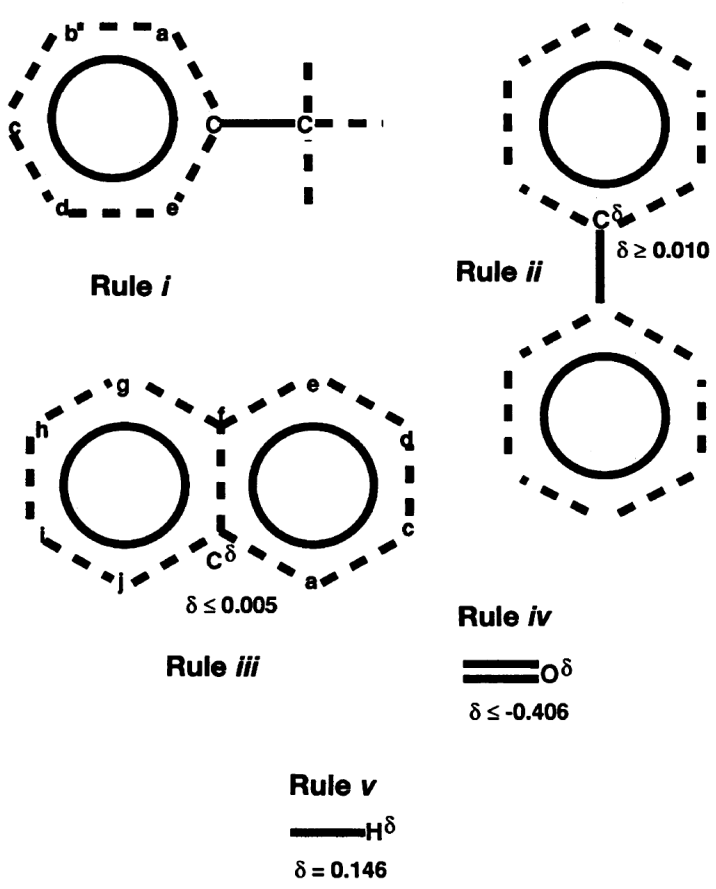
(i) it has an aliphatic atom carbon attached by a single bond to a carbon atom which is in a six-membered aromatic ring, or

(ii) it has a carbon atom in an aryl-aryl bond between two benzene rings with a partial charge greater than 0.010, or

(iii) it has an oxygen atom in a nitro (or related) group with a partial charge less than 0.406, or

(iv) it has a hydrogen atom with a partial charge of 0.146, or

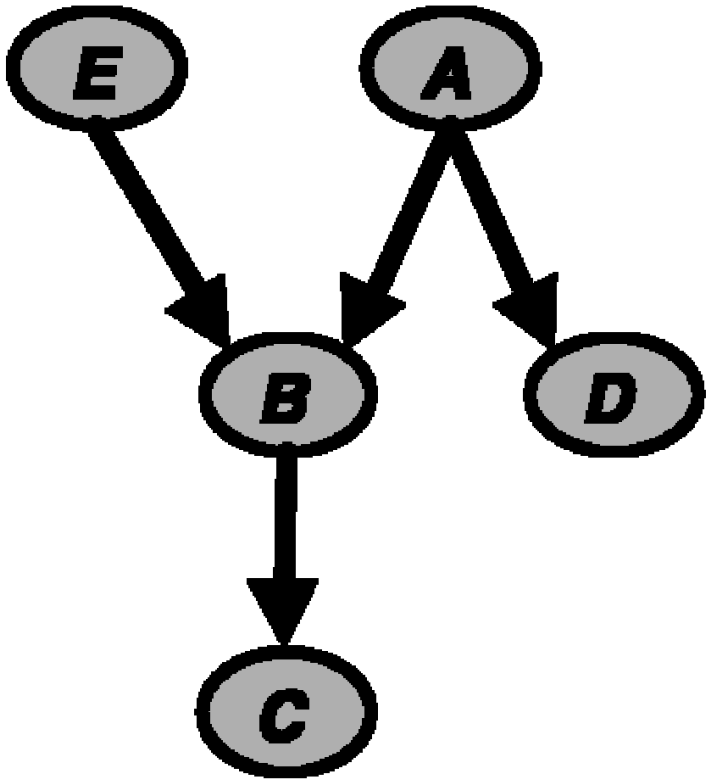
(v) it has a carbon atom that merges six-membered aromatic ring with a partial charge les than 0.005



Genetic Algorithms

- Evolve population of structures (or programs specifying structures)
- Use operators that simulates biological mutation or recombination
- filtering process that simulates natural selection
- Requires building representation & genetic operators fitted to problem
- Computationally intensive

Graphical Models

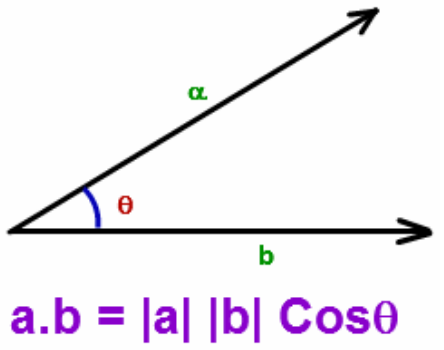


We will get to this soon

Kernels : similarity measure

- Given two molecular structures ***u*** and ***v***, a kernel *k(**u**,**v**)* is a measure of similarity between ***u*** and ***v***

- What if we define *k(**u**,**v**)* =<***u***, ***v***> ?



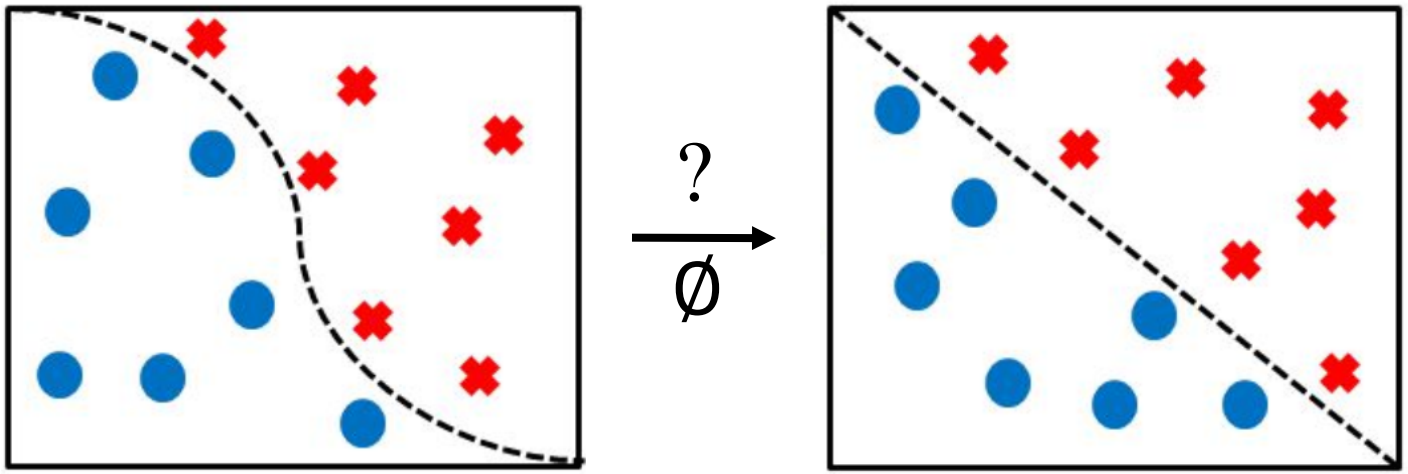
- Dot product is usually a good similarity measure in $\mathbb{R}^d$.

- It is high whenever the two vector have similar directions (angle small)

- But in case of variable-size data (e.g. graphs) dot product make no sense.

Kernels Trick

- Kernel trick is a way to map variable size data to a fixed size data



$$k(\boldsymbol{u},\boldsymbol{v}) = \langle \emptyset(\boldsymbol{u}), \emptyset(\boldsymbol{v}) \rangle$$

- In the mapped space, we can use dot-product as a measure of similarity.

Review of Support Vector Machines

- Training dataset is $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_l, y_l)\}$
- Test dataset is $\mathcal{S} = \{(\boldsymbol{x}_{l+1}, y_N), ..., (\boldsymbol{x}_{l+1}, y_N)\}$
- $\boldsymbol{x}_i \in \mathbb{R}^d$
- $y_i \in \{-1, +1\}$

- Learning is building a function $f: \mathbb{R}^d \longrightarrow \{-1, +1\}$ from training set $\mathcal{S}$ such that the error is minimal on test dataset

Review of Support Vector Machines

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\mathrm{y} = f(\mathbf{x}) = \operatorname{sgn}\left(\sum_i \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b \right)$$

Observations :

- $\boldsymbol{w}$ is a linear combination of $\boldsymbol{x_i}$
- The predictor depends only on dot prodcut of $\boldsymbol{x_i}$ and $\boldsymbol{x}$

Kernel learning

Support Vector Machine

- $f(\boldsymbol{x})=sign(\sum_{i=1}^{l} \alpha_i y_i < \boldsymbol{x}_i , \boldsymbol{x} > +b)$

Kernel trick : apply linear approach to transformed data $\emptyset(\boldsymbol{x}_1)$ ... $\emptyset(\boldsymbol{x}_n)$

- $f(\boldsymbol{x})=sign(\sum_{i=1}^{l} \alpha_i y_i < \emptyset(\boldsymbol{x}_i) , \emptyset(\boldsymbol{x}) > +b)$

Kernel trick

- Replace $<\emptyset(\boldsymbol{x}), \emptyset(\boldsymbol{x}')>$ with $k(\boldsymbol{x}, \boldsymbol{x}')$

- $f(\boldsymbol{x})=sign(\sum_{i=1}^{l} \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x})+b)$

Positive definite kernels

Let kernel $k: \chi \times \chi \rightarrow \mathbb{R}$ be a continuous and symmetric function

k positive definite if for all $l \in \mathbb{N}$ and $\boldsymbol{x}_1 \ldots \boldsymbol{x}_l \in \mathbb{R}$

$\lambda \times \lambda$ matrix $K=(k(\boldsymbol{x}_i, \boldsymbol{x}_j))$ $1 \leq i, j \leq \lambda$ is positive definite

Mercer's property

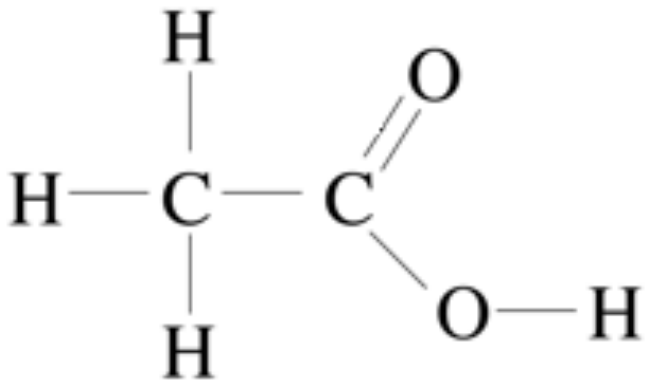
- For any (positive definite) kernel function, there is a mapping ϕ into the feature space $\mathcal{H}$ equipped with inner product such that

$$\forall\, \boldsymbol{x}, \boldsymbol{x}' \in \chi, \qquad k(\boldsymbol{x}, \boldsymbol{x}') = < \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') >_{\mathcal{H}}$$

Graph Kernel

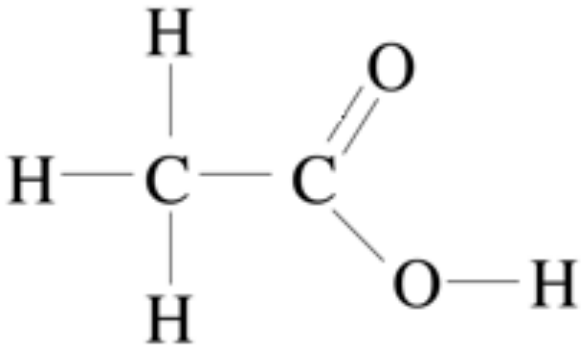
A proper graph kernel is a vector representation of graph

More similar graphs should have more similar representations



$\phi \rightarrow$ (4, 2, 5, 1, 6, 3, …)

Adjacency Matrix

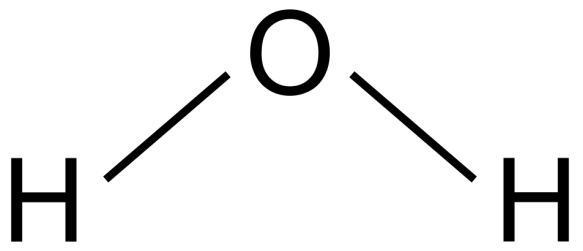


- $G = (\mathcal{V}, \mathcal{E})$
- $\mathcal{V} = \{v_1, \ldots, v_n\}, Lv\,(i) \in \{O, C, H, N\}$
- $\mathcal{E} = \{e_1, \ldots, e_m\}$,
- $n \times n$ adjacency matrix E of graph G
- $E_{ij}=1$ if there is an edge between nodes v_i & v_j

- The graph uniquely identified by $n \times 1$ label list L_v and $n \times n$ adjacency matrix E

Is there a unique adjacency matrix for each metabolite ?

- Consider metabolite H_2O



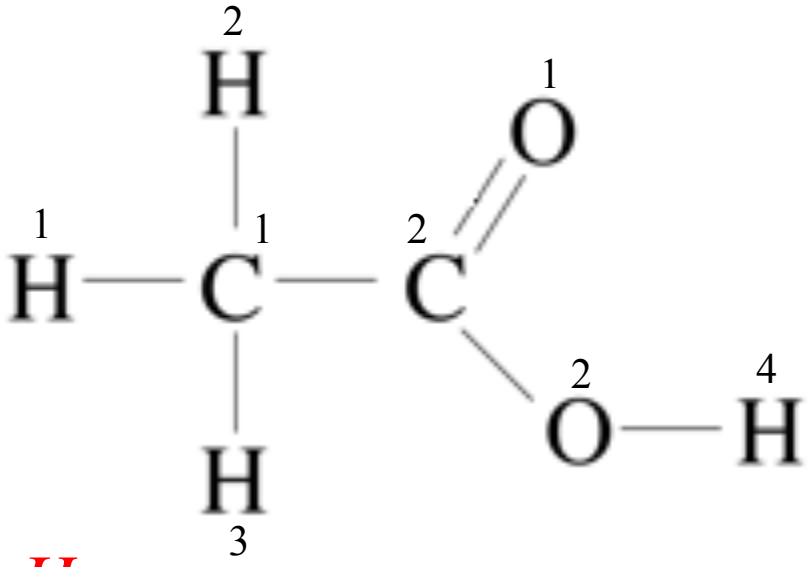
$$L_v = [O\ H\ H]$$

$$\mathrm{E} = \begin{matrix} 0 & 1 & 1 & O \\ 1 & 0 & 0 & H \\ 1 & 0 & 0 & H \end{matrix}$$

$$L^v = [H\ O\ H]$$

$$\mathrm{E} = \begin{matrix} 0 & 1 & 0 & H \\ 1 & 0 & 1 & O \\ 0 & 1 & 0 & H \end{matrix}$$

Example



$$L_v = [\overset{1}{H}\ \overset{1}{C}\ \overset{2}{H}\ \overset{3}{H}\ \overset{2}{C}\ \overset{1}{O}\ \overset{2}{O}\ \overset{4}{H}]$$

$$E = \begin{matrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & H \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & C \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & H \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & H \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & C \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & O \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & O \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & H \end{matrix}$$

Kernels and Graph Representation

- Lets consider metabolite graphs G with representations (E, L_v)

- A Kernel map $\phi(G) = \phi(E, L_v)$ defined by the graph representation is consistent, if it does not depend on the specific representation

- If (E', L'_v) are alternative representations of G :

$$\phi(E, L_v) = \phi(E', L'_v)$$

Example 1

- $\phi(H_iO_jC_lN_r)=(i,j,l,r)$

- $k(H_iO_jC_lN_r, H_{i'}O_{j'}C_{l'}N_{r'}) = <\phi(H_iO_jC_lN_r), \phi(H_{i'}O_{j'}C_{l'}N_{r'})>$
  $= <\phi(i, j, l, r), \phi(i', j', l', r')> = i.i' + j.j' + l.l' + r.r'$

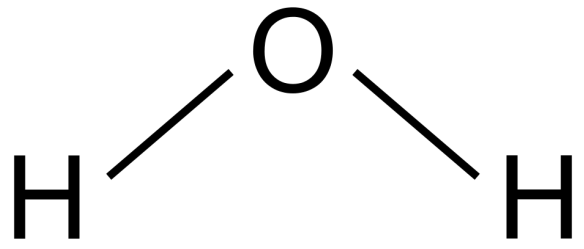
- $H_2O \rightarrow (2,1,0,0)$
- $CO_2 \rightarrow (0,2,1,0)$
- $k(H_2O, C_2O) = (2,1,0,0).\ (0,2,1,0) = 2$

- Consistent
- Not a good kernel
- depends only on *L* (labels) and not *E* (metabolite structure)

Example 2

$\phi(E,L)=E_{13}$

This kernel is not consistent



$L_v = [O\ H\ H]$

$$\mathrm{E} = \begin{matrix} 0 & 1 & 1 & O \\ 1 & 0 & 0 & H \\ 1 & 0 & 0 & H \end{matrix}$$

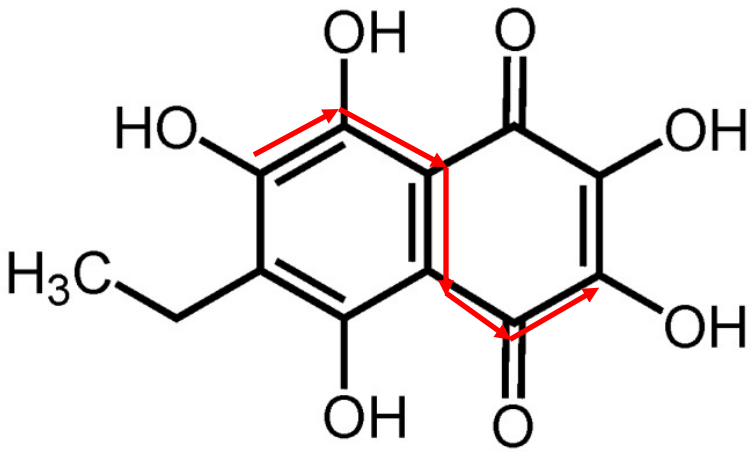
$\phi(E,L)=1$

$L'_v = [H\ O\ H]$

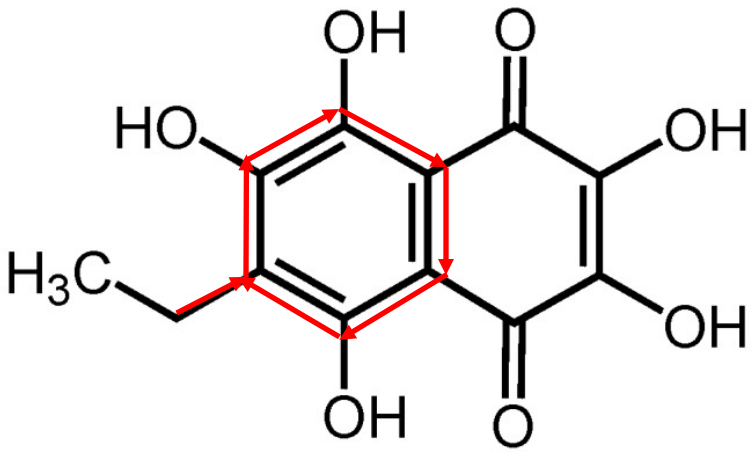
$$\mathrm{E} = \begin{matrix} 0 & 1 & 0 & H \\ 1 & 0 & 1 & O \\ 0 & 1 & 0 & H \end{matrix}$$

$\phi(E',L')=0$

Walks in a graph

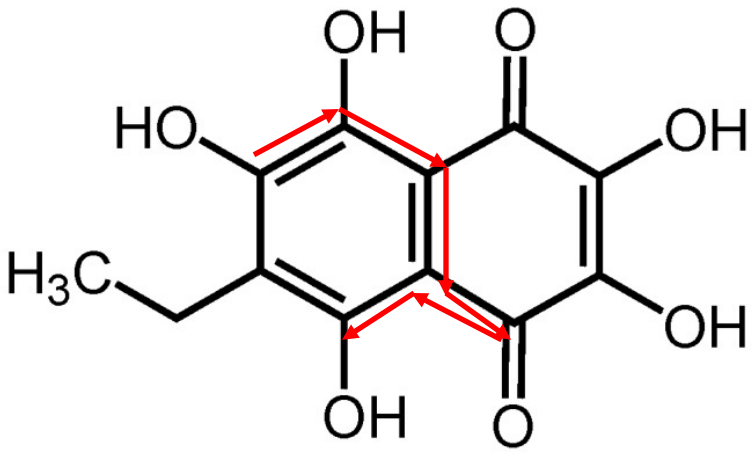


Walks in a graph



Walk in a graph with cycle

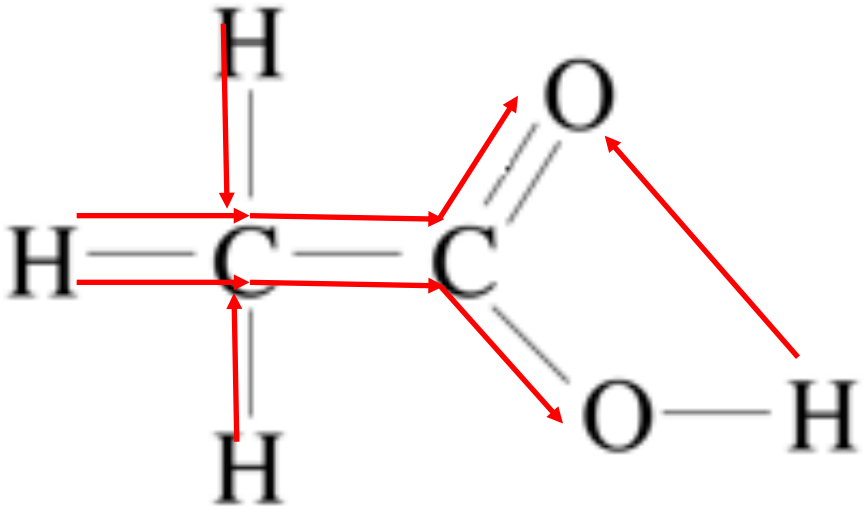
Walks in a graph



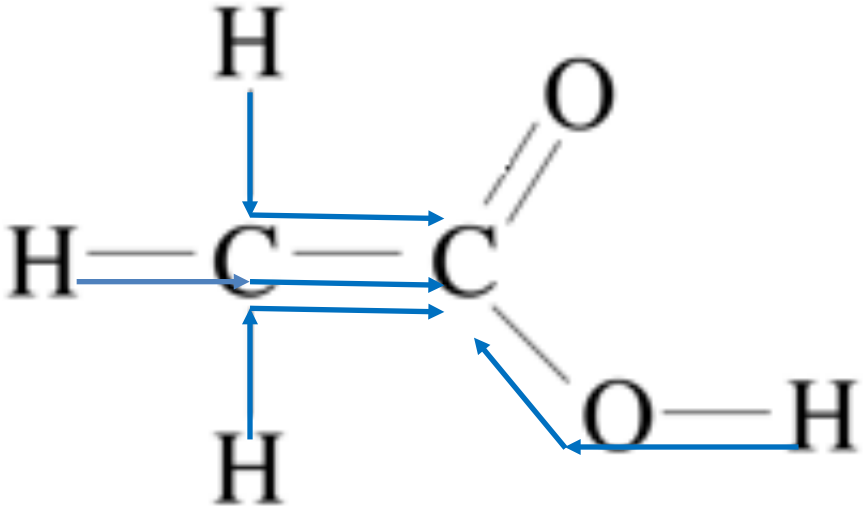
Walk in a graph with double traverse

Example 3 : Label paired kernels

length 3 walks H → O : 7



length 2 walks H → C : 4

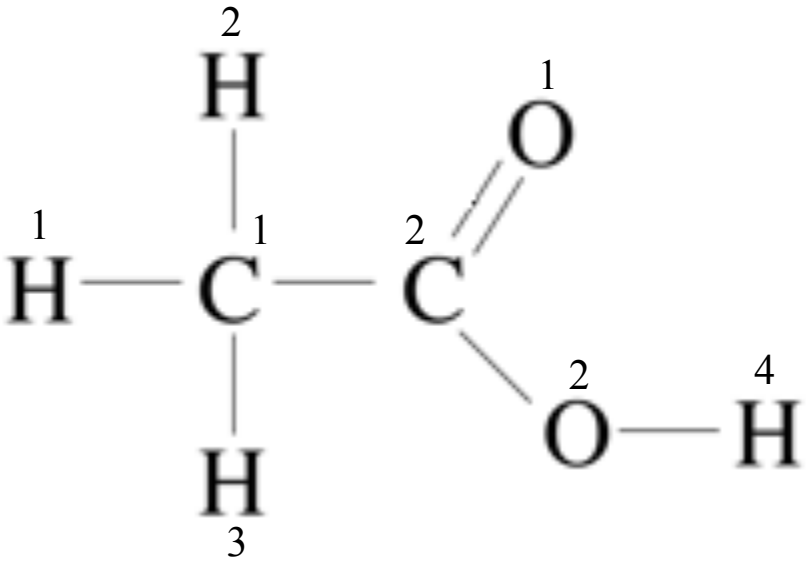


Example 3 : Label paired kernels

- Given graphs G_1 & G_2, count the number of walks in G_1 and G_2 of the same length i and with the label a at first and label b for last node
- $a, b \in \{H, O, C, N\}$

- How can we compute these numbers from E and L *?*

Example



$L_v = [\underset{1}{H}\ \underset{1}{C}\ \underset{2}{H}\ \underset{3}{H}\ \underset{2}{C}\ \underset{1}{O}\ \underset{2}{O}\ \underset{4}{H}]$

E =

0	1	0	0	0	0	0	0	H
1	0	1	1	1	0	0	0	C
0	1	0	0	0	0	0	0	H
0	1	0	0	0	0	0	0	H
0	1	0	0	0	1	1	0	C
0	0	0	0	1	0	0	0	O
0	0	0	0	1	0	0	1	O
0	0	0	0	0	0	1	0	H

length 1 walks

$H \rightarrow C$

1+1+1+0+0+0+0+0 = 3

$$L_v = \begin{matrix}[H & C & H & H & C & O & O & H] \\ 1 & 1 & 2 & 3 & 2 & 1 & 2 & 4\end{matrix}$$

$$\mathcal{L} = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & O \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & H \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & C \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & N \end{matrix}$$

$$E = \begin{matrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{matrix}$$

$$\mathcal{L}^t = \begin{matrix} O & H & C & N & \\ 0 & 1 & 0 & 0 & H \\ 0 & 0 & 1 & 0 & C \\ 0 & 1 & 0 & 0 & H \\ 0 & 1 & 0 & 0 & H \\ 0 & 0 & 1 & 0 & C \\ 1 & 0 & 0 & 0 & O \\ 1 & 0 & 0 & 0 & O \\ 0 & 1 & 0 & 0 & H \end{matrix}$$

Computing the number of length 1 walks

$$\phi(E, \mathcal{L}) = \mathcal{L}\, E \mathcal{L}^{\mathrm{t}} =$$

O	H	C	N	
0	1	2	0	O
1	0	3	0	H
2	3	2	0	C
0	0	0	0	N

- Each derivative of this matrix corresponds to the number of length 1 walks from one element to another

Computing the number of length i walks

$$\phi(E, \mathcal{L}) = \mathcal{L}\, Ei\mathcal{L}^{\mathrm{t}} =$$

O	H	C	N	
0	1	2	0	O
1	0	3	0	H
2	3	2	0	C
0	0	0	0	N

- E^i contains information about the number of walks of length i

Label paired kernels

- $k^i(G_1, G_2) = < \mathcal{L}E_1^i\mathcal{L}^{\mathrm{t}}, \mathcal{L}E_2^i\mathcal{L}^{\mathrm{t}} >$

- Inner product is Frobenius matrix norm

Label paired kernels

- $k(G_1, G_2) =$

$$< \mathcal{L}_1 \left(\sum_{i=1}^{\infty} \lambda_i E_1^i \right) \mathcal{L}_1^t, \mathcal{L}_2 \left(\sum_{i=1}^{\infty} \lambda_i E_2^i \right) \mathcal{L}_2^t >$$

Three examples of label based kernel

- Exponential kernel

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

- Truncated power series

$$\sum_{i=0}^{\infty} \lambda_i E^i = \sum_{i=0}^{p} (\gamma E)^i$$

- Convergent geometric kernel

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

Bottlenecks of labeled pair kernels

- Limited expressivity of kernel (feature space $|\mathcal{A}|^2$=16
- Important feature of chemical compound missed)
- No consideration of the sequence of nodes traversed in the walk
- when counting $\mathrm{H} \rightarrow C$ walks with length 2, $\mathrm{H} \rightarrow O \rightarrow C$ and $\mathrm{H} \rightarrow N \rightarrow C$ treated similarly
- Equal importance to uninformative/noisy/self intersecting walks

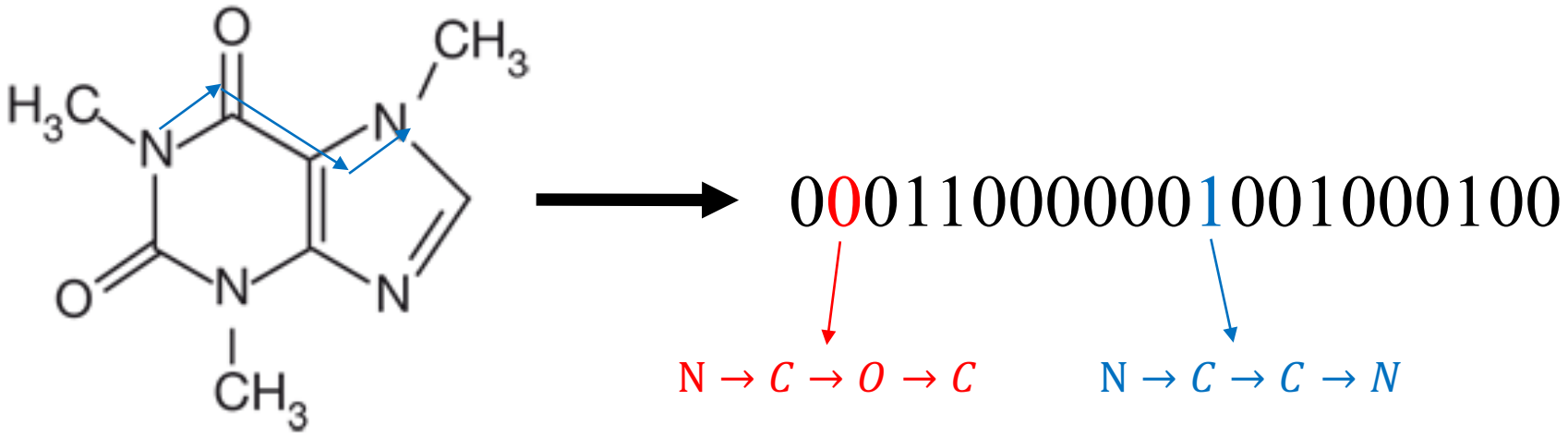
Kernels based on sequence of labels

- To resolve bottlenecks of the label paired kernel, we can consider the number occurences of a specific sequence, e.g. H $\rightarrow O \rightarrow C$

- Called the "sequence of labels" kernel

From chemical structures to molecular fingerprints

Find all paths of length *d* in graph using depth-first-search (*d*=8 or 10)

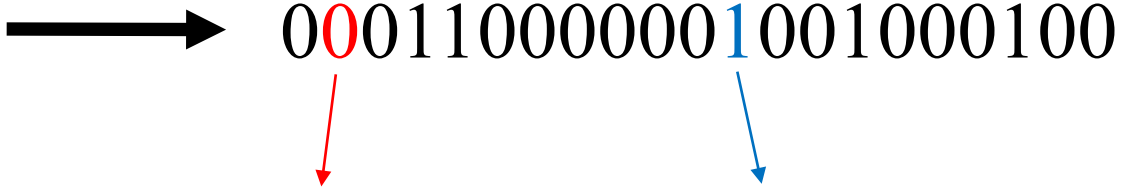


Each bit corresponds to presence / absence of a path

Modeling Molecular Fingerprint as documents with word

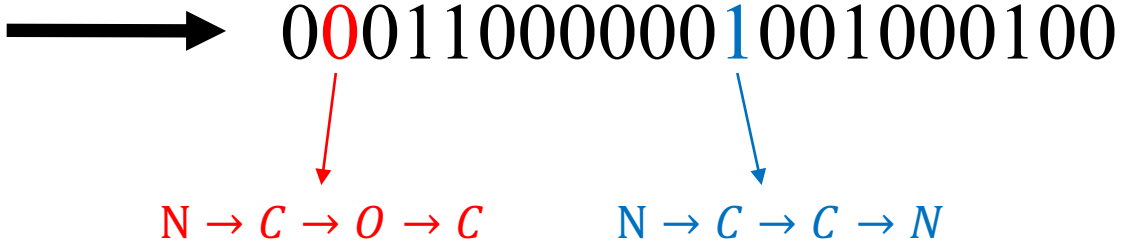
$$\phi_d(\mathbf{u}) = (\phi_{\text{path}}(\mathbf{u}))_{\text{path} \in \wp(d)}$$

$\phi_{path}(u)$ is one if at list one depth first search produces path



$$k_d(\mathbf{u}, \mathbf{v}) = \langle \mathbf{u}, \mathbf{v} \rangle_d = \sum_{\text{path} \in \wp(d)} \phi_{\text{path}}(\mathbf{u}) \phi_{\text{path}}(\mathbf{v})$$

Compressed represention



- The size of such bit vector can grow very large
- One strategy to reduce the size, is to select vector size l (e.g. l=1024), and for present path with index I, set I module l to 1.
- Result in representation of size l

0001100 0000100 1000100 →

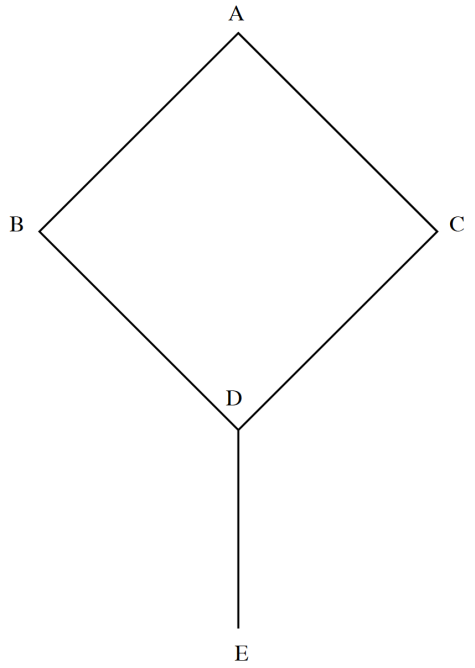
0001100
0000100
1000100

1001100

$l=7$

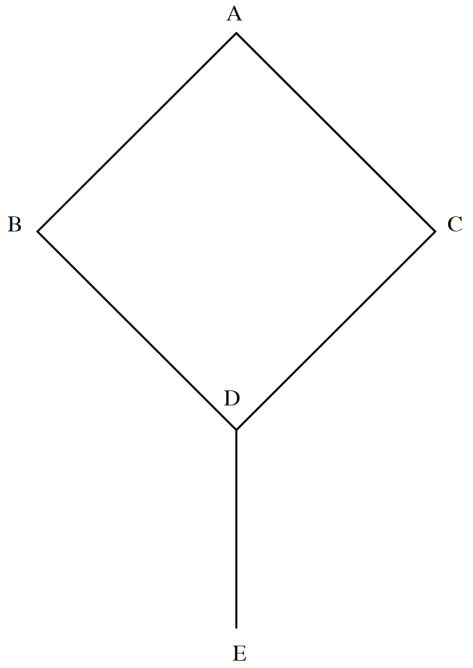
(1) Depth First Search (no cycle, no double traverse)

- A
- A-B
- A-B-D
- A-B-D-C
- A-B-D-E
- A-C



- A-B-D-C-A not allowed because of cycle
- A-C-D not allowed because C-D already traversed

(2) Depth First Search (cycle OK, no double traverse)

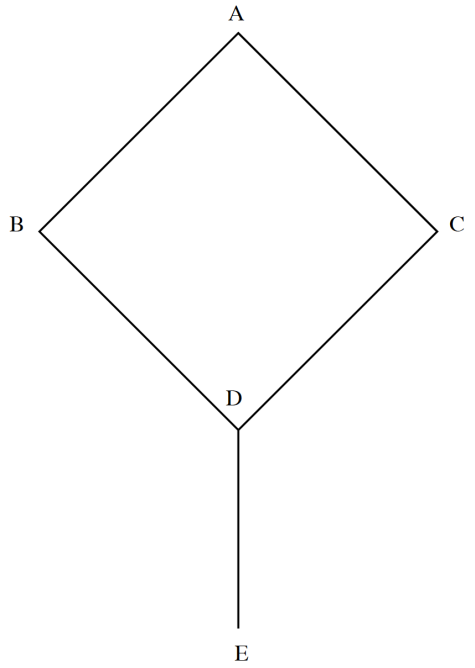


- A
- A-B
- A-B-D
- A-B-D-C
- A-B-D-C-A
- A-B-D-E

- A-C not allowed because already traversed

(3) Depth First Search (no cycle, double traverse OK)

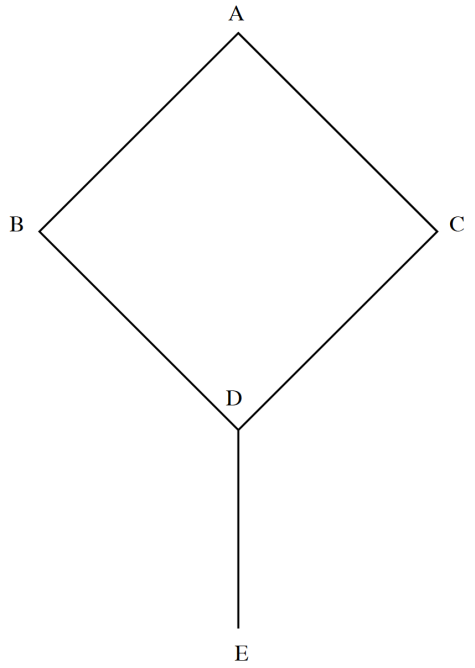
- A
- A-B
- A-B-D
- A-B-D-C
- A-B-D-E
- A-C
- A-C-D
- A-C-D-B
- A-C-D-E



- A-B-C-D-A not allowed because of cycle

(4) Depth First Search (cycle OK, double traverse OK)

- 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



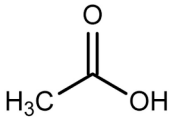
Complexity

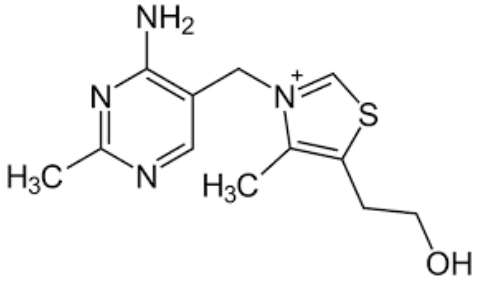
- Molecule with *n* atoms and *m* edges
- For case (1) & (2) complexity is *O(mn)*
- For case(3) & (4), complexity grow with $O(n\alpha^d)$
- *d* is depth and α branching factor of the graph

Normalizing the kernels

- This kernel is not normalized for different sizes of molecules.

- The larger the molecules are, the higher is the kernel





$\phi_d(u)$ = 000100000000100

$\phi_d(u)$ = 010110011010110

Tanimoto kernel

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

$\phi_d(u)$ = 100101000100100
$\phi_d(v)$ = 010100010101100

$k_d(u,u)$ = 5
$k_d(v,v)$ = 6
$k_d(u,v)$ = 3

$k^t{}_d(u,v)$ = 3/(5+6-3)=3/8

- Tanimoto kernel is a normalized kernel
- Always between 0 and 1

Hybrid kernel

- Tanimoto Kernel only counts the number of common paths between two structures

- Hybrid Kernel tries to score both the paths that are common to the two structures, and the paths that are missing from both

$$k^H{}_d(u,v) = k^t{}_d(u,v) + {\sim}k^t{}_d(u,v)$$

$$k_d(u,v) = < \phi_d(u), \phi_d(v) >$$

$${\sim}k_d(u,v) = < {\sim}\phi_d(u), {\sim}\phi_d(v) >$$

MinMax kernel

$$k_d^m(\mathbf{u},\mathbf{v}) = \frac{\sum_{\text{path}\in\wp(d)} \min(\varphi_{\text{path}}(\mathbf{u}), \varphi_{\text{path}}(\mathbf{v}))}{\sum_{\text{path}\in\wp(d)} \max(\varphi_{\text{path}}(\mathbf{u}), \varphi_{\text{path}}(\mathbf{v}))}$$

$\phi_d(u)$ = 502040120
$\phi_d(v)$ = 215030041
Min = 202030020
Max = 515040141

- Take into account the count of paths
- For binary input, identical to Tanimoto

MinMax and Tanimoto are the same for binary data

$\phi_d(u)$ = 100101000100100
$\phi_d(v)$ = 010100010101100

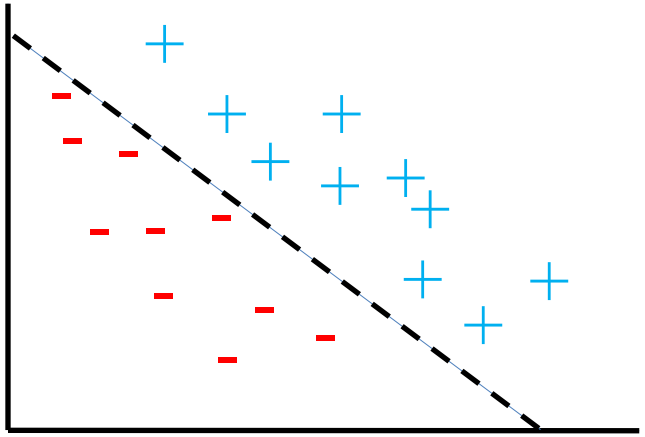
$k^t{}_d(u,v)$ = 3/(5+6-3)=3/8

$\phi_d(u)$ = 100101000100100
$\phi_d(v)$ = 010100010101100
Min = 000100000100100
Max = 110101010101100

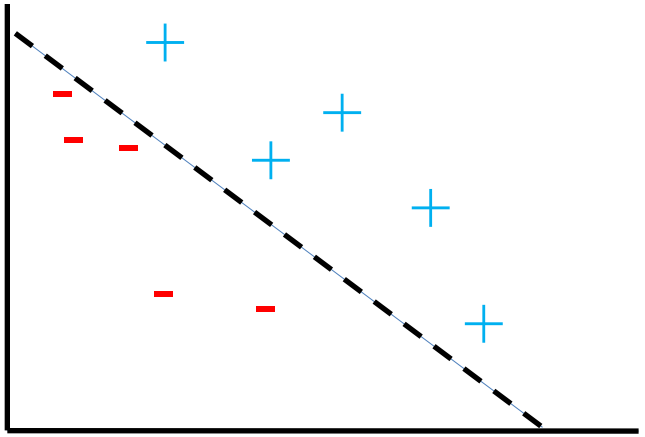
$k^M{}_d(u,v)$ = 3/8

Cross validation

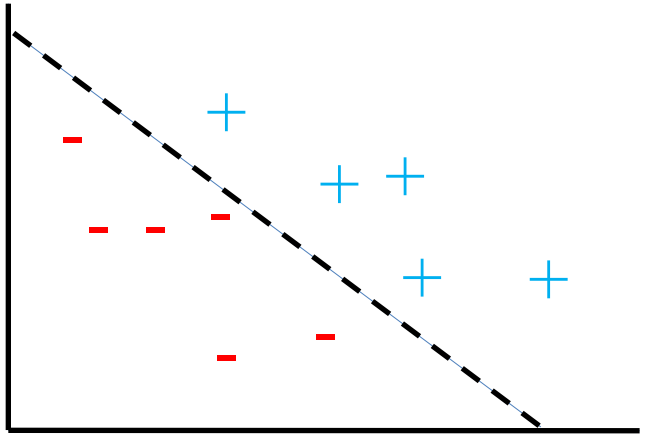
- To limit over-fitting
- Divide data into training and test



training



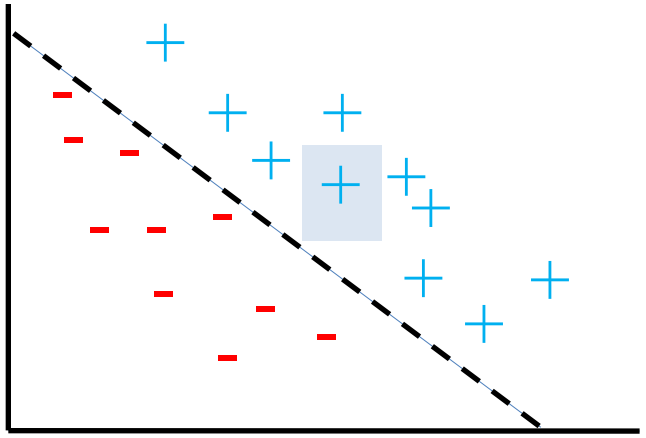
test

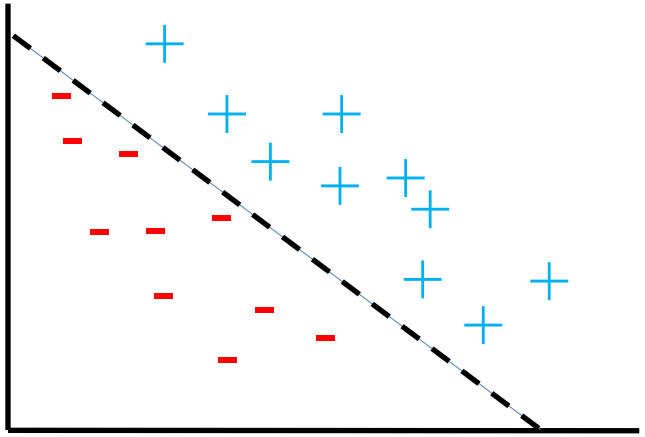


Leave one out strategy

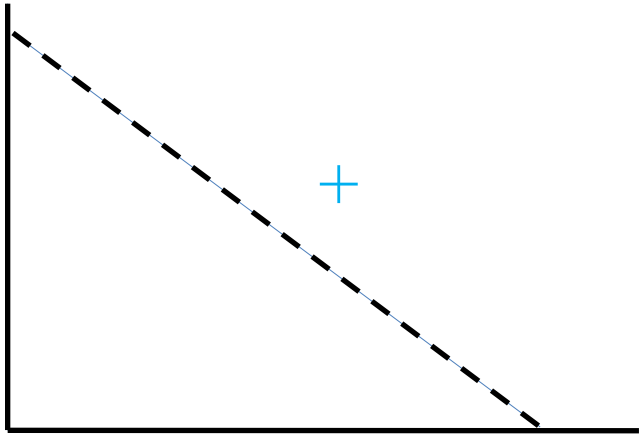
- At each step, remove one datapoint from the training set, and only test that one
- Repeat for all data points

training





test



Datasets : Muatg

Mutagenecity of molecules (ability to change DNA/ increase frequency of mutation)

- Total : 188
- Number of positives : 125 (66%)
- Number of negatives : 63 (33.5%)
- Average #atom/mol : 17.9
- Average #bond/mol : 19.7
- Average degree : 2.21

Results :

Kernel/Method	Mutag
Tanimoto	87.8
MinMax	86.2
Tanimoto, $l=1024$, $b=1$	87.2
Hybrid, $l=1024$, $b=1$	87.2
Tanimoto, $l=512$, $b=1$	84.6
Hybrid, $l=512$, $b=1$	86.7

Leave one out strategy

Application of Graph Kernels : protein function prediction

BIOINFORMATICS

Vol. 21 Suppl. 1 2005, pages i47–i56
doi:10.1093/bioinformatics/bti1007

Protein function prediction via graph kernels

Karsten M. Borgwardt[1,], Cheng Soon Ong[2], Stefan Schönauer[1], S. V. N. Vishwanathan[2], Alex J. Smola[2] and Hans-Peter Kriegel[1]*

[1]Institute for Computer Science, Ludwig-Maximilians-University Munich, Oettingenstraße 67, 80538 Munich, Germany and [2]National ICT Australia, Canberra, 0200 ACT, Australia

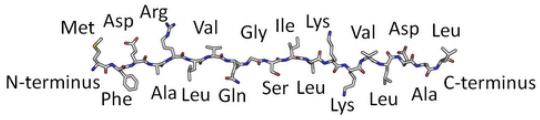
Received on January 15, 2005; accepted on March 27, 2005

Graph Structure of a Protein

- Model the protein as a graph
- Nodes & edges of the graph contain information about the secondary structure
- Graph model contains information about structure, sequence, and chemical properties of the protein

Protein Structure

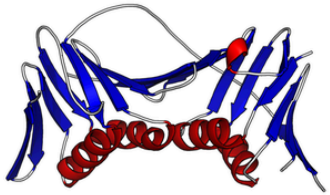
Primary structure



Secondary structure

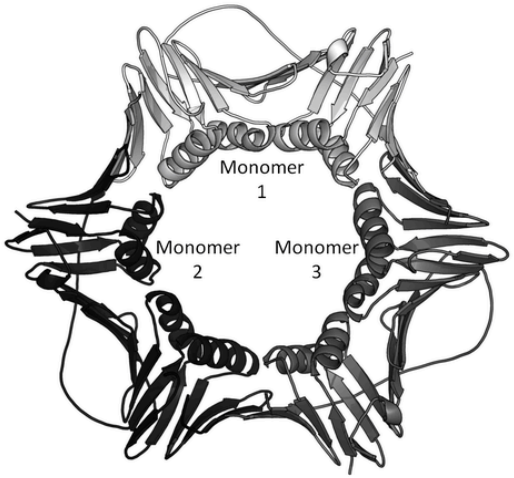
regular local sub-structures of polypeptide backbone chain

Tertiary structure



3-D structures possessing discrete functions

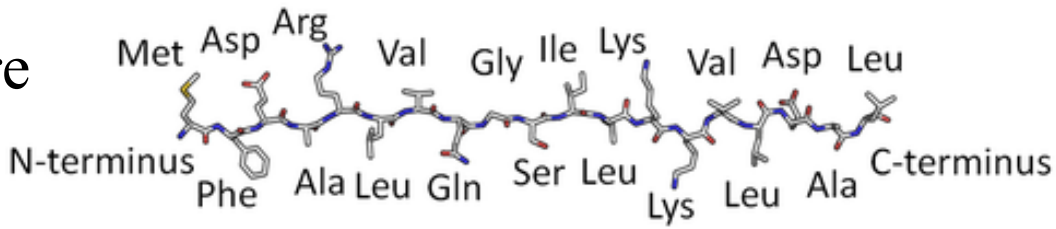
Quaternary structure



aggregation of two or more individual polypeptide chains that operate as a single functional unit

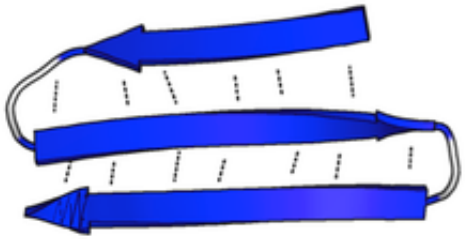
Protein secondary Structures

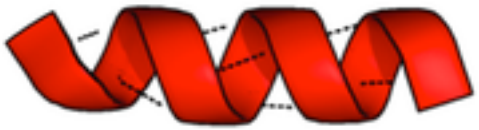
Primary structure

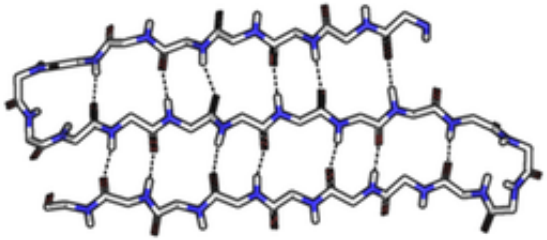


Secondary structure

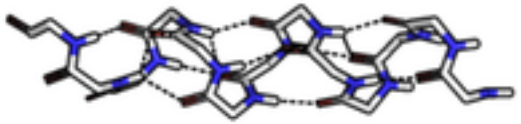
The three dimensional form of a part of protein







β-Sheet (3 strands)

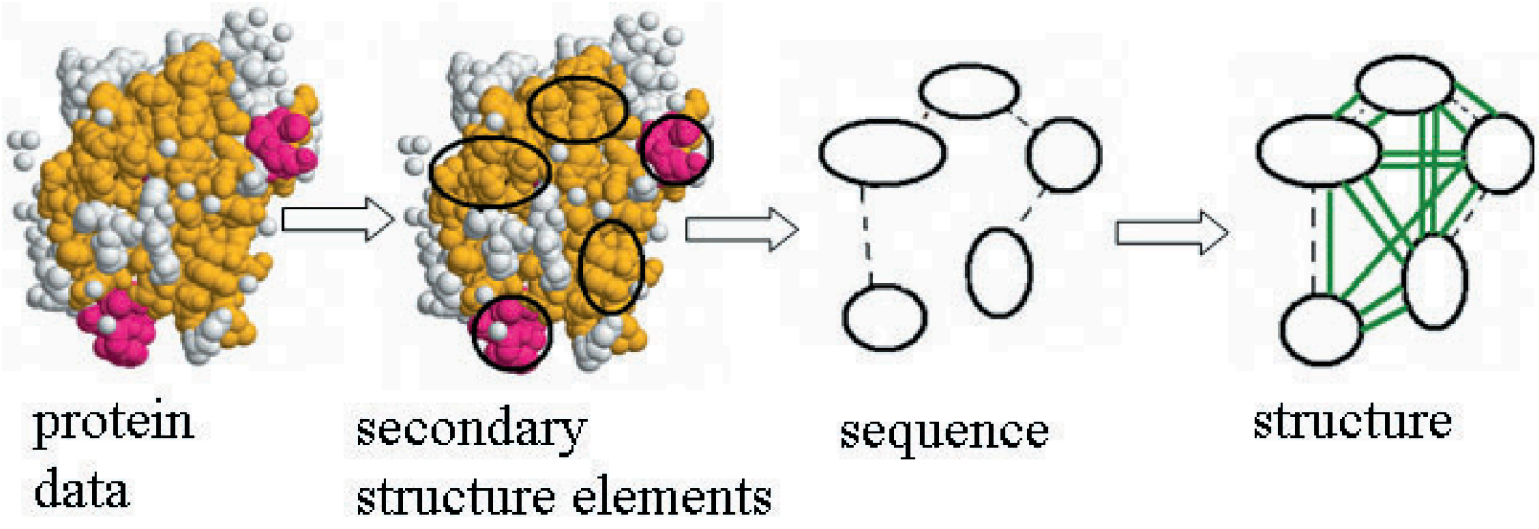


Alpha-helix

From Protein to graphs

- Each node represent a secondary structure

- Two nodes connected if either they are (i) close in 3-dimensional space (structural edge), or (ii) next to each other in primary structure



Node label

- Node labels contain information about
  - Structure (Helix, Sheet, Turn?)
  - Hydrophobicity
  - Van der Waal Volume
  - Polarity

Protein function

- Catalyzing metabolic reactions
- DNA replication
- Responding to stimuli
- Transporting molecules

Predicting protein function

- Proteins with similar functions have similar structures

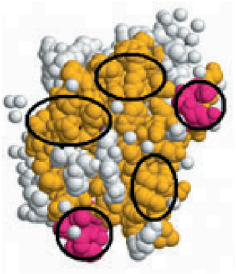
- Proteins with similar structures have similar graphs

The learning problem

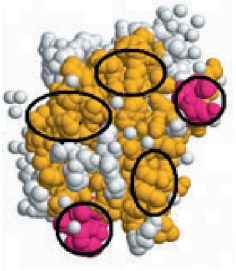
- We have a set of c functions F=$\{1,\ldots,c\}$
- We have a training set of proteins (represented by graphs) along with their known functions (G_i, f_i), $f_i \in F$
- Our goal is to learn the relationship between graph structure & function
- And predict function for new graph structures

The learning problem

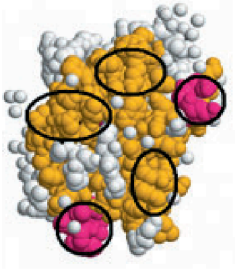
Training



f_1

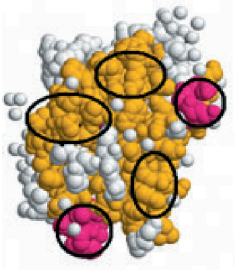


f_2



f_3

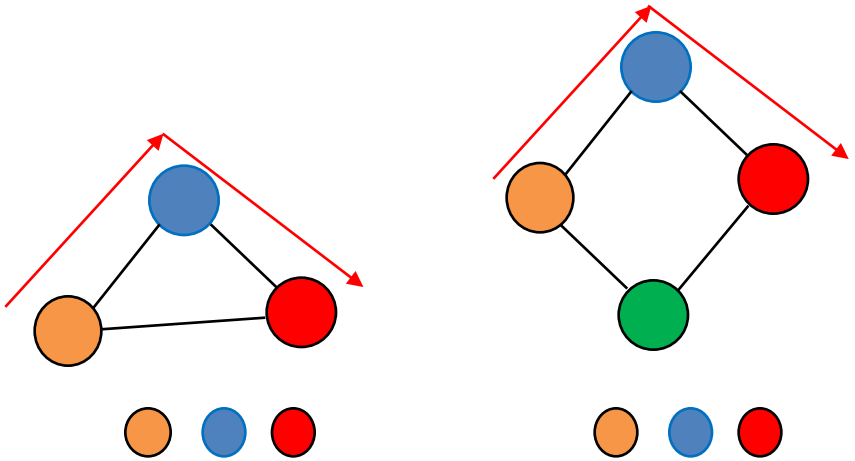
New



?

Random walk kernel

- Counts the number of walks of a specific length in the two graphs that go through the same set of labels.



Random Walk Kernels

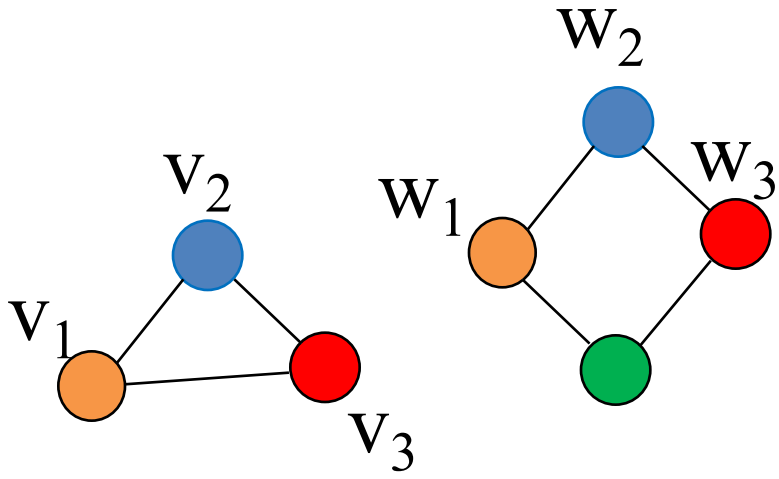
- In practice, the labels of nodes on the walks are not identical
- We can first define kernels on similarity of walks
- Then extend kernel from walk to graph :

$$k_{\text{graph}}(G_1, G_2) = \sum_{\text{walk}_1 \in G_1} \sum_{\text{walk}_2 \in G_2} k_{\text{walk}}(\text{walk}_1, \text{walk}_2).$$

Walk kernel

$$k_{walk}(walk_1, walk_2) = \prod_{i=1}^{n-1} k_{step}((v_i, v_{i+1}), (w_i, w_{i+1})).$$

$$k_{step}((v_i, v_{i+1}), (w_i, w_{i+1}))$$
$$= k_{node}(v_i, w_i) * k_{node}(v_{i+1}, w_{i+1})$$
$$* k_{edge}((v_i, v_{i+1}), (w_i, w_{i+1})),$$



$$k_{node}(v_i, w_i)$$
$$= k_{type}(v_i, w_i) * k_{node\ labels}(v_i, w_i) * k_{length}(v_i, w_i).$$

Classifying enzymes from non-enzymes

- Enzymes are proteins that are responsible for accelerating chemical reactions.

Kernel type	Accuracy	SD
Vector kernel	76.86	1.23
Optimized vector kernel	80.17	1.24
Graph kernel	77.30	1.20
Graph kernel without structure	72.33	5.32
Graph kernel with global info	84.04	3.33