Graph Kernels and Support Vector Machines for Pattern Recognition

Léo Andéol[1]
Supervised by: Prof. Hichem Sahbi

Master DAC - Sorbonne Université

May 2019

[1] leo.andeol@gmail.com

Summary

Motivation

- A lot of data can be represented as graphs such as proteins or social networks
- Being able to compare them would be useful (classification, clustering)

Figure 1: A fragment of a protein transformed into a graph[1]

Current Methods

Support Vector Machine

SVMs are models used in classification introduced about 25 years ago. They have several advantages

- Great accuracy
- Great capacity of generalization
- Allows the use of kernels in its dual form

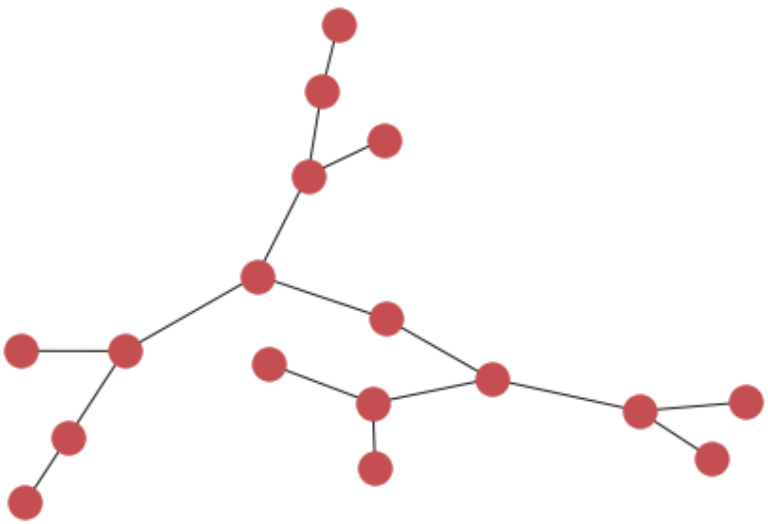
Kernels

The kernel trick can replace the dot product while implicitly projecting data to a feature space and combine very well with the SVMs

- Computes data projection faster implicitly (ex. RBF kernel)
- Improve the accuracy of SVM by making linear separation easier

Objective

- These methods are adapted to **vector data**
- Graphs and their adjacency matrices aren't and vectorizing implies a loss of information
- New types of kernels were discovered
- However, the complexity is a big problem, these kernels have to be accelerated



$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Summary

Background : graphs

Definition

A graph[2] is a type of mathematical structure that represents connections between objects. It is more precisely an ordered pair $G = (V, E)$ of two sets: vertices V (or nodes) and edges E that connect two vertices together.

$$E \subseteq \{(u, v) : (u, v) \in V^2\}$$

Properties

- Undirected
- Labeled or not
- Degree
- Path and Cycle
- Connected
- Tree
- Subgraph
- Line Graph

Background : Support Vector Machines

Classification

Classification is the problem of finding the best function $f : \mathbb{R}^d \longrightarrow \{0..k\}$ among a set of functions F while minimizing a risk function R

$$f\star = \text{argmin}_f R(f)$$

where k is the number of classes of the problem and d the number of features of data

Support Vector Machines

$$\gamma_i = \frac{y_i(\mathbf{x}_i \cdot \mathbf{w} + w_0)}{\|\mathbf{w}\|}$$

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{s.t.} \quad y_i(\mathbf{x}_i \cdot \mathbf{w} + w_0) \geq 1 \quad \forall i \in \{1..n\}$$

Background : kernels

Definition

In its dual form, the SVM problem only requires a dot product between the observations' vectors.

$$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^\top \mathbf{x_j}$$

This means the vectors can be mapped to higher dimensions with a function ϕ. Moreover, even the dot product itself can be replaced by a function κ without explicitly specifying the map ϕ as long as the function is positive semi definite.

$$\kappa(\mathbf{x_i}, \mathbf{x_j}) = e^{-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma^2}}$$

An example of kernel : the RBF kernel

Graph Kernels

Definition

Graph Kernels are a type of R-convolution kernels[3] applied to graphs, which are kernels that are based on decompositions of complex objects and comparisons of those decompositions.

- Random walks
- Graphlets
- Shortest paths, subtree

Graphlets

Definition

Let G and G_2 be two graphs, $\mathbf{f}_G$ and $\mathbf{f}_{G_2}$ the frequency vectors of respectively G and G_2, then the kernel κ is defined as

$$\kappa(G, G_2) = \mathbf{f}_G^\top \mathbf{f}_{G_2}$$

- Complexity of computing $\mathbf{f}_G$ is $O(n^k)$

Random Walks

Random Walks

A random walk is a path obtained from a chosen vertex, randomly picking an edge to follow iteratively, until the trail reaches a certain length. Comparing common random walks between graphs is an acceptable metric.

Product graph

It has been shown[4] that computing random walks on two separate graphs is equivalent to computing it on the product graph of the two. The product graph is computed using the Kronecker (or tensor) product $\otimes$

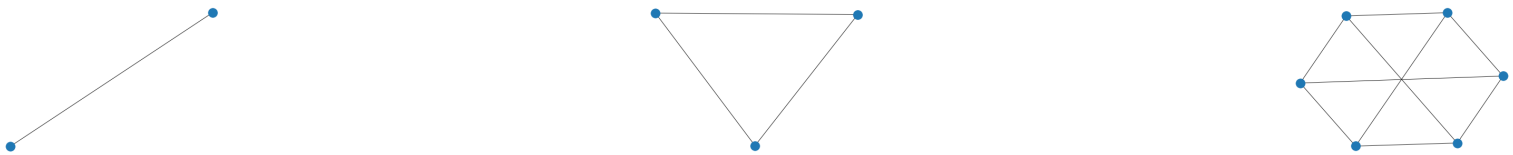


Figure 3: A graph G_1, a graph G_2 and the product graph $G_1 \otimes G_2$

Random Walk

Product graph

- $W_\times = A_1 \otimes A_2$
- $W_\times = \sum_{l=1}^{d} A_1^{(l)} \otimes A_2^{(l)}$

Kernel Definition

Let $p_\times$ and $q_\times$ be respectively the start and end probability of each node, and let $W_\times$ be the adjacency matrix of the product graph of G and G', and finally $\mu(k)$ be a convergent function of k.

$$\kappa(G, G') = \sum_{k=0}^{\infty} \mu(k) q_\times^\top W_\times^k p_\times$$

Acceleration methods

Inverse Kernel

A special case where $\mu(k) = \lambda^k$ leads to the following kernel

$$\kappa(G_1, G_2) = q_\times^\top (I - \lambda W_\times)^{-1} p_\times$$

- $O(kn^6)$ complexity
- Some methods will try to accelerate it

Sylvester Equation

Definition

Let A, B, and C be matrices of compatible shapes, then the Sylvester equation is

$$AX + XB = C$$

And the discrete-time Sylvester Equation is

$$AXB + C = X$$

Which can be generalized as

$$\sum_{i=0}^{d} A_i X B_i + C = X$$

Conjugate Gradient

Idea

- The idea is to make new gradient orthogonal to the former
- Convergence guaranteed in $|V|$ steps to solve the following problem

$$(I - \lambda W_{\times})x = p_{\times} \tag{1}$$

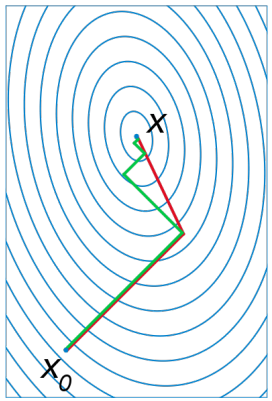


Figure 4: Conjugate gradient (red) compared to Gradient Descent (green)

Fixed Point Iterations

Definition

Fixed point iterations is a method of computing a fixed point of a function by applying repeatedly the following equation until $\|x_{n+1} - x_n\| < \epsilon$ where ϵ is the acceptable level of error.

$$x_{n+1} = f(x_n)$$

Since this method requires the computation of $W_\times$, the kernel value can be computed for any type of labeling. For unlabeled graphs, the complexity is $O(kn^3)$ and $O(kdn^3)$ for labeled graphs, where d is the number of labels and k the number of iterations which can be estimated by

$$k = O\left(\frac{\ln \epsilon}{\ln \lambda + \ln|\xi|}\right)$$

Spectral Decomposition

Definition

- Eigendecomposition of the adjacency matrix
- Inverse becomes trivial to compute on diagonal matrix

$$\kappa(G_1, G_2) = \sum_{k=0}^{\infty} \mu(k) q_\times^\top (V_\times D_\times V_\times^{-1})^k p_\times = q_\times^\top V_\times \left(\sum_{k=0}^{\infty} \mu(k) D_\times^k \right) V_\times^{-1} p_\times$$

However the following property is given from the Kronecker product

$$A_1 \otimes A_2 = (V_1 D_1 V_1^{-1}) \otimes (V_2 D_2 V_2^{-1}) = (V_1 \otimes V_2)(D_1 \otimes D_2)(V_1 \otimes V_2)^{-1}$$

Taking advantage of that property

$$\kappa(G_1, G_2) = (q_1^\top V_1 \otimes q_2^\top V_2)(\sum_{k=0}^{\infty} \mu(k)(D_1 \otimes D_2)^k)(V_1^{-1} p_1^\top \otimes V_2^{-1} p_2^\top) \quad (2)$$

Nearest Kronecker Product Approximation

- The idea is to approximate two matrices S and T such that $\|W_\times - A \otimes B\|_F$ is minimized.
- Labeled-graph kernel computation can be turned into into an unlabeled one with some loss in accuracy, but gain in computation time.
- Computed in $O(dn^2)$ time
- All methods such as Spectral Decomposition can then be applied

Summary

Databases and metrics

Synthetic Database

A database of toy data was required to verify claims made in the studied article. A generator was written that can make graphs of "star", "tree" and "ring" types, of different sizes with different labels.

Databases and metrics

Synthetic Database

A database of toy data was required to verify claims made in the studied article. A generator was written that can make graphs of "star", "tree" and "ring" types, of different sizes with different labels.

Real Databases

- Proteins : 1113 proteins, 2 classes
- Enzymes : 600 enzymes, 6 classes

Databases and metrics

Synthetic Database

A database of toy data was required to verify claims made in the studied article. A generator was written that can make graphs of "star", "tree" and "ring" types, of different sizes with different labels.

Real Databases

- Proteins : 1113 proteins, 2 classes
- Enzymes : 600 enzymes, 6 classes

Metrics

$$L(X, \mathbf{y}) = \frac{1}{|X|} \sum_{i=1}^{|X|} \begin{cases} 1 & \text{if } f(\mathbf{x}_i) \neq y_i \\ 0 & \text{otherwise} \end{cases}$$

Implementation

- Sylvester Equation
  $A_1 X A_2 + C = X$

Implementation

- Sylvester Equation
  $A_1 X A_2 + C = X$

- Conjugate Gradient
  $(I - \lambda W_\times)x = p_\times$

Implementation

- Sylvester Equation
  $A_1 X A_2 + C = X$
- Fixed point
  $x_{t+1} = \lambda W_\times x_t + p_\times$
- Conjugate Gradient
  $(I - \lambda W_\times)x = p_\times$

Implementation

- Sylvester Equation
  $A_1 X A_2 + C = X$
- Fixed point
  $x_{t+1} = \lambda W_\times x_t + p_\times$
- Conjugate Gradient
  $(I - \lambda W_\times)x = p_\times$
- Spectral Decomposition
  $q_\times^\top P_\times (I - \lambda D_\times)^{-1} P_\times^{-1} p_\times$

Performance : Gram matrices

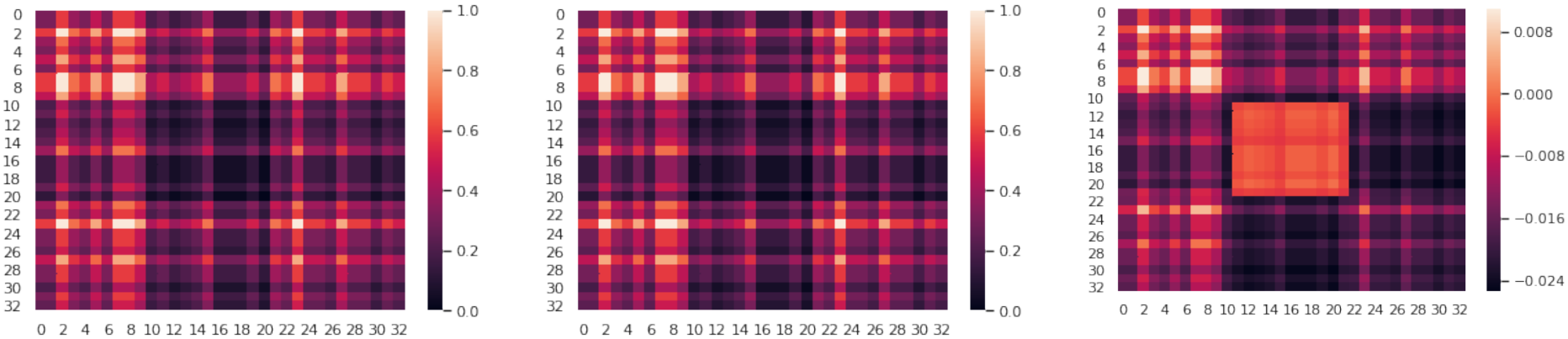


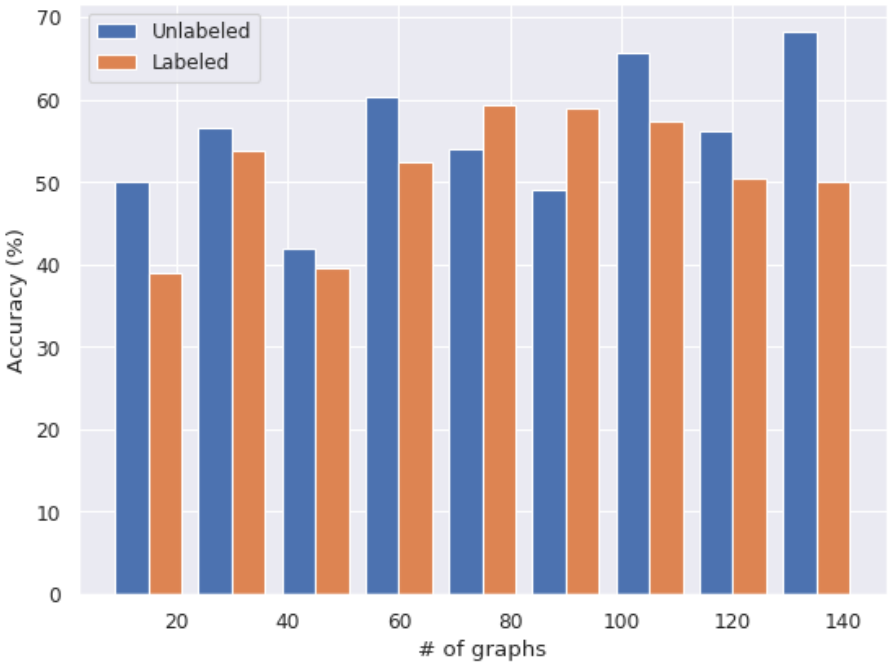
Figure 5: A gram matrix computed with the raw method, another with the fixed point method, and the difference between the two

Performance : Gram matrices

	Raw kernel	Inverse Kernel	Sylvester Equation	Conjugate Gradients	Fixed points	Spectral Decomp.
Raw.	0	1.1e-4	9.8e-5	8.9e-5	1.0e-4	1.0e-04
Inv.	-	0	2.1e-5	7.9e-5	4.0e-6	6.8e-6
Syl.	-	-	0	8.0e-5	1.7e-5	1.4e-5
Con.	-	-	-	0	7.9e-5	7.9e-5
Fix.	-	-	-	-	0	2.8e-6
Spe.	-	-	-	-	-	0

Table 1: Mean standard deviation of matrix entries

Performance : Label Use



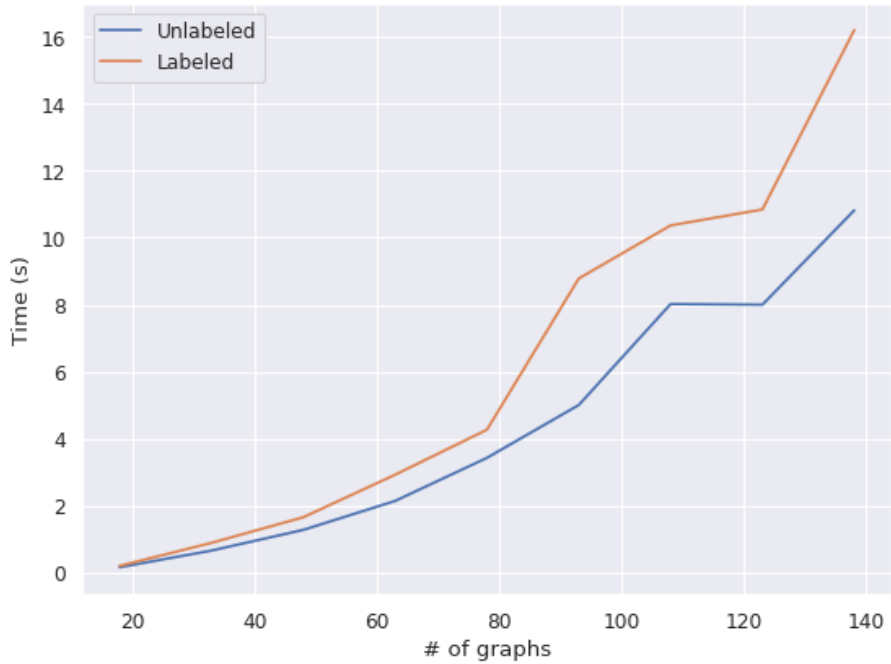
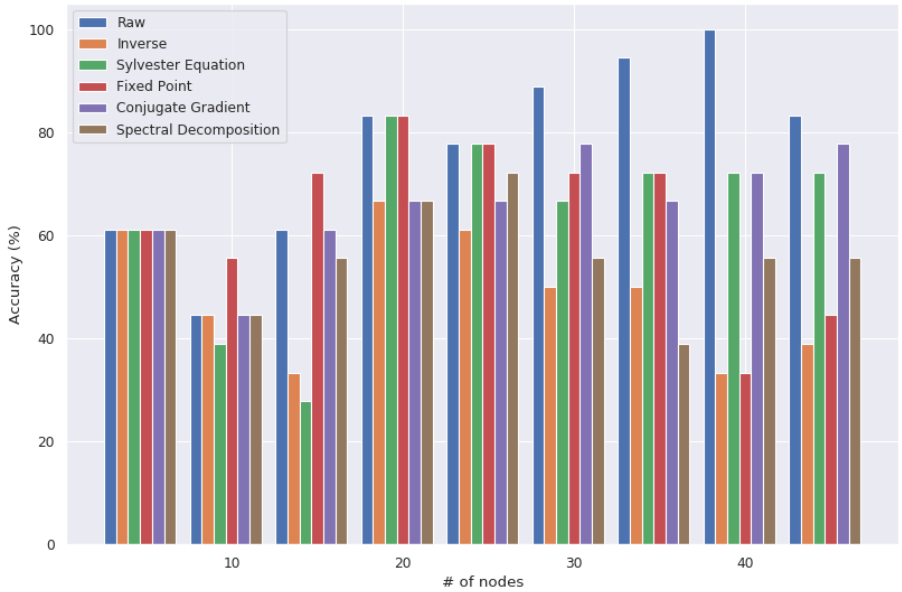


Figure 6: Accuracy and computation time of learning for unlabeled and labeled graphs

Performance : Complexity



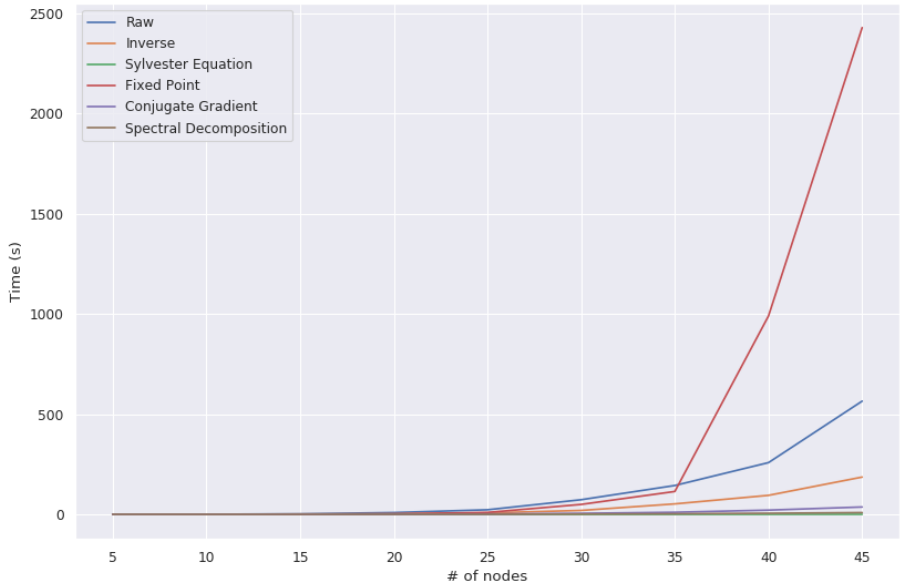


Figure 7: Accuracy and Computation time of different methods depending on the size of graphs

Protein and Enzymes

Conclusion

Conclusion

- Random walks : decent accuracy
- Several acceleration methods were introduced
- Performances are greatly improved
- conclusion on real data : todo

Future work

- Generalized Sylvester Equation : algorithm and complexity
- Generalizing the Spectral Decomposition to labeled graphs

References

[1] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph Kernels," *Journal of Machine Learning Research*, vol. 11, no. Apr, pp. 1201–1242, 2010.

[2] J. A. Bondy, U. S. R. Murty, *et al.*, *Graph theory with applications*, vol. 290. Citeseer, 1976.

[3] D. Haussler, "Convolution kernels on discrete structures," Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA, 1999.

[4] W. Imrich and S. Klavzar, *Product graphs: structure and recognition*. Wiley, 2000.

[5] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics (Oxford, England)*, vol. 21 Suppl 1, pp. i47–56, June 2005.

[6] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Min. Knowl. Discov.*, vol. 2, pp. 121–167, June 1998.

[7] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley-Blackwell, Oct. 1998.

[8] Y. Nesterov, *Lectures on Convex Optimization*. Springer Optimization and Its Applications, Springer International Publishing, 2 ed., 2018.

[9] N. Shervashidze, S. V. N. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Artificial Intelligence and Statistics*, pp. 488–495, Apr. 2009.

??

??