Graph Kernels and Support Vector Machines for Pattern Recognition

Léo Andéol*
Master DAC, Sorbonne Université
Paris, France

May 2019

Abstract

Nowadays the world is turning to a new era where data is the most valuable resource, and we are working on algorithms to exploit those the best way possible. Most research is currently focused on the so-called "big data", huge quantities of data which are very sparse and of poor quality. However, there also exist structured databases of good quality, in the form of graphs, which is the one we will be studying. Graphs can be very useful in various fields, but is mostly known for its use in biology, as it can be used to represent proteins or other types of molecules.
I chose to work on Support Vector Machines(cite qui/quand), a well known and powerful algorithm of machine learning. It is a perfect fit for studying graphs and learning to compare them because the algorithm allows through an elegant trick the use of several types of structured data.
The goal of this project is to implement a working algorithm and its improvements, test it on different types of data and try to improve it.

reecrire une fois rapport fini : Rajouter apport projet et facteur acceleration

*leo.andeol@gmail.com

Contents

1 Introduction

introduction générale plus étendue : 1 et 1/2 page :donner un aperçu/résumé du rapport : un paragraphe par élément - ECRIRE A LA FIN

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque

egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

2 Methodology

2.1 Background

sous section 1 : background : graphe, svm, noyaux, donner toutes les définitions.

2.1.1 Graphs

Reference pour graphes : ???

comment relier les définitions

A graph[1] is a type of mathematical structure that represents connections between objects. It is thus composed of vertices (or nodes) and edges. Vertices represent objects and are usually depicted as circles or spheres whereas edges link pairs or vertices.

Definition 1. A graph is an ordered pair $G = (V, E)$ where :

- V is a set of nodes
- $E \subseteq \{(u, v) : (u, v) \in V^2\}$

The edges of the graph can be weighted and can be either oriented or not. We will be focusing on undirected unweighted graphs.

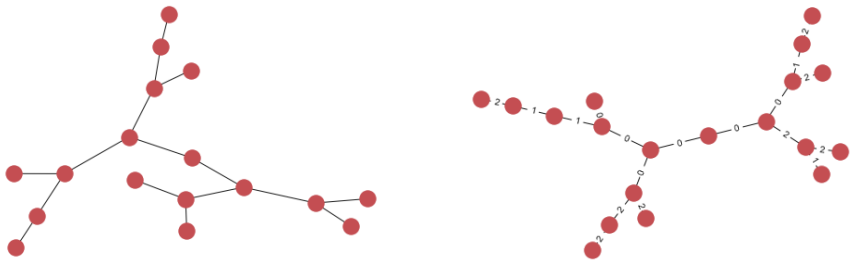

Figure 1: Two "tree" graphs, resp. unlabeled and labeled

Definition 2. Let $G = (V, E)$ be a graph, if G is undirected, then we have

- $\forall (u, v) \in E \implies (v, u) \in E$

Degree

Definition 3. The degree of a vertex $d(v)$ is the number of vertices it is connected to.

It is ... ?

Definition 4. A path

Definition 5. A cycle

Labels and colors Both vertices and edges can have labels (sometimes called colors), they can take various forms : they can be integers, elements of a finite or infinite set (such as integers but also strings), or even continuous (such as $\in \mathbb{R}$).

Connectivity Connectivity is another basic concept of graphs. It means that all vertices are accessible from any vertex.

Definition 6. Let $G = (V, E)$ be a graph,
G is connected if and only if there is a path between every pair of vertices

If a graph is a not connected, and a subgraph induced by it is, then this subgraph is called a connected component.
Graphs were first used in their modern form to represent the problem of Seven Bridges of Königsberg (cite), and have been ever since used to represent maps, and thus path-finding algorithms were developed. Graphs can also be used, with labels, to represent different type of molecules and interactions between them.(parler aussi des réseaux ? des problemes de flots etc ?)
During this project, we will focus on undirected graphs which can have any type

of labels on their edges only (if it is only on their nodes, the complementary graph can be studied).
We will be comparing graphs in order to find similarities between them, and classify them in different groups. It has been shown(cite) that it can be used effectively on proteins and (?).

2.1.2 Support Vector Machines

Support Vector Machines (SVM) are a type of machine learning algorithms discovered in the early 90s(cite). It was originally a classification algorithm however it has been expanded since to regression and clustering too. It is especially powerful and widely used for two main reasons :

Margin and support vectors The Support Vector Machines were based on the model of the Perceptron (cite), another classification algorithm that tried to find an hyperplane that discriminates the two sets.
The main issue with the Perceptron is that it had no formal guarantee to find an optimal hyperplane, and more often than not would not find it. Moreover, if the data weren't linearly separable, the algorithm would not converge.
In order to tackle theses issues, the SVM offered three fixes.
First, they added a margin in the loss function in order to not only classify well the data samples, but also with the maximum certainty, i.e. as far from the decision boundary as possible.

$$L(y, x, w) = \max(0, -y * x \cdot w) \implies L(y, x, w) = \max(0, 1 - y * x \cdot w) \quad (1)$$

Then, seeing this solution wasn't enough, as there was still an infinity of possible solutions, it was proven that the margin between points and the decision boundary was inversely proportional to the norm of the weights.

$$\gamma = \frac{y_i(x_i \cdot w + w_0)}{\|w\|} \quad (2)$$

These two fixes gave us the first version the SVM, the hard-margin SVM :

$$\min \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(x_i \cdot w + w_0) \geq 1 \quad \forall i \in \{1..n\} \quad (3)$$

However, if the data weren't linearly separable the algorithm wouldn't be optimizable since the quadratic programming problem requires all points to be correctly classified. Then, a new term ξ was introduced as an error tolerance, as well as a factor C that would determine the balance between error tolerance and weights minimization.

$$\begin{aligned} &\min \quad \tfrac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i \\ &\text{s.t.} \quad \forall i \in \{1..n\} \qquad y_i(x_i \cdot w + w_0) \geq 1 - \xi_i \end{aligned} \quad (4)$$

2.1.3 Kernels

In its dual form, the SVM problem only requires a dot product between the sample vectors. Then, we can use a nonlinear transformation $\phi(x)$ to replace the dot-product $\mathbf{x_i} \cdot \mathbf{x_j}$ by $\phi(\mathbf{x_i}) \cdot \phi(\mathbf{x}_j)$, effectively augmenting the dimensions of our vectors and thus the expressiveness of our model by enhancing the separability of the data. However, for complex transformations, such as in infinite dimensions, the function ϕ isn't defined. But it was found(cite) that the standard dot product can be replaced by another function as long as it is a bilinear positive semi definite form.

Definition 7. A kernel K is positive semi definite if and only if :

$$\sum_{i=1}^{n}\sum_{i=1}^{n} K(\mathbf{x_i}, \mathbf{x_j}) c_i c_j \geq 0 \qquad \forall i \in \{1..n\} \quad c_i \in \mathbb{R} \tag{5}$$

Thus, the dot product can be replaced by a function $K(x_1, x_2)$ which will indirectly (?) transform the data in larger dimensions, even infinite with the Gaussian kernel(cite) and return the dot product of those transformations, while remaining computable in all cases. This replacement is usually called the "Kernel Trick".
Ever since the foundation of SVMs, the kernel trick became a big focus of the machine learning community as it naturally fits in the algorithm and allows supervised learning on very complex data, and enjoying greater accuracy than most algorithms.

2.2 State of the Art

Mettre explication comme "ici nousallons parler de ça ça " ou pas

2.2.1 Introduction

In this chapter we will discuss the advance of research on graph kernels.Graph Kernels are a type of R-convolution kernels[2] applied to graphs, which are kernels that are based on decompositions of complex objects and comparisons of those decompositions.
Graph Kernels have been studied since Support Vector Machines started getting popular[3]. Since then, a lot of progress has been made, and several types of kernels have been discovered, such as random walk[3] and graphlet[4] kernels, the two we will be studying, but not the only ones. There are also graph kernels that use spanning trees, and several others that are unfortunately usually not positive semi-definite[5].

découverts ou inventés?

2.2.2 Graphlets

A faire plus tard

A graphlet[4] is a subgraph of 3,4 or 5 nodes taken from a much larger graph. The idea of a graphlet kernel is to count all types of graphlets that appear in a graph, and use that count to compare to another graph.
insert list of graphlets

Citer deux fois un seul article ok ou pas?

2.2.3 Random walks

Graph kernels based on Random Walks have been studied since very early[3], several of them were created and were then united[6]. Random walks are a type of rather intuitive algorithms : the idea is to randomly walk through a graph and then compare it to random walks in another graph. Actually, all the random walks are computed at the same time by making the adjacency matrix stochastic and using it as a Markov chain. Moreover, it has also been shown[7] that performing a walk on two separate graphs at the same time is the same as performing a walk on the product graph. The product graph is in simpler terms the Cartesian product of the two graphs, taking all possible combinations of nodes and linking those nodes when there is an edge between two specific nodes in both the original graphs.

Definition 8. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs, the product graph $G_\times = (V_\times, E_\times)$ is then defined as

- $V_\times = \{(v, w) : v \in V_1, w \in V_2\}$
- $E_\times = \{((v_1, w_1), (v_2, w_2)) : v_1, v_2 \in V_1^2, w_1, w_2 \in V_2^2\}$

However, technically, the adjacency matrix $W_\times$ of a such graph is the result of the Kronecker (tensor) product of the two adjacency matrices A and A_2 of the two graphs [7, 8]:

$$W_\times = A \otimes A_2 \tag{6}$$

We will also need the *vec* operator that is very linked to the tensor product :

Definition 9. Let A be a matrix $n \times n$, $vec(A)$ is the vector obtained by stacking all columns of A.

2.2.4 Kernel definition

finir

The kernel is defined as follows

Definition 10. Let $p_\times$ and $q_\times$ be respectfully 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.

$$k(G, G') = \sum_{k=0}^{\infty} \mu(k) q_\times^\top W_\times^k p_\times \tag{7}$$

2.2.5 Sylvester Equation

A Sylvester equation, also sometimes called Lyapunov equation (we will stick to Sylvester here) is a matrix equation of the following shape :

Definition 11. Let A, B, and C be matrices of compatible shapes, then the Sylvester equation is

$$AX + XB = C \tag{8}$$

And the discrete-time Sylvester Equation is

$$AXB + C = X \tag{9}$$

Which can be generalized as

$$\sum_{i=0}^{d} A_i X B_i = X \tag{10}$$

We will be exclusively using the discrete-time Sylvester Equation and its generalization and will be referring to it as Sylvester equation. These equations are solved usually using Schur decompositions in $O(n^3)$ for the basic Sylvester equation and in unknown time for the generalized version[6].
We can use this equation to solve our kernel faster by replacing the A and B matrices by our adjacency matrices A_1 and A_2, adding λ, and C by our start probabilities $p_\times$ while vectorizing the whole equation thus obtaining :

colon or not?

$$vec(M) = vec(\lambda A_1 M A_2) + p_\times \tag{11}$$

From which we can obtain (where I is the identity matrix)

partie speciale pour notations?

$$q_\times^\top vec(M) = q_\times^\top (I - \lambda W_\times)^{-1} p_\times \tag{12}$$

Thus, having calculated the inverse kernel by an alternative method. The advantage of this method is that it is very fast, but unfortunately limited to unlabeled graphs, unless there is an implementation for the generalized Sylvester equation. Moreover, compared to the others, this option is very simple and does not require specific parameters, but like many others, suffers when it encounters singular matrices.

donner des exemples = comme quoi

sous section 2 : state of the art : graph kernels et acceleration des random walk, donner les complexité, donner des exemples, discuter avantages et inconvenients chacun

2.2.6 Conjugate Gradient Method

The Conjugate Gradient Method is an optimization algorithm used to find approximate solutions of linear systems that have a positive semi definite matrix. As its name suggests, it is a gradient based (usually) iterative algorithm. The Main idea is assumes as geometric $\mu(k)$ function

2.2.7 Fixed Point Iterations

Faire usage de fct paragraph ??

A fixed point is a value for which a function, returns that same value :

Definition 12. Let S a set and f a map $f : S \implies S$,
$x \in S$ is a fixed point of f if and only if $x = f(x)$

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.

$$x_{n+1} = f(x_n) \tag{13}$$

There is also a guarantee of convergence :

Theorem 1. *In order for the fixed-point iterations to converge, it is necessary that $\lambda < |\xi|^{-1}$ where ξ is the largest eigenvalue of $W_\times$*

le mettre en appendix?

Proof. Let $x_0 = p_\times$ and $t >> 0$

$$\begin{aligned}
x_{t+1} - x_t &= p_\times + \lambda W_\times x_t - x_t \\
&= p_\times + (\lambda W_\times - 1)(p_\times + \lambda W_\times x_{t-1}) \\
&= p_\times - p_\times + \lambda W_\times p_\times + (\lambda W_\times)^2 x_{t-1} - \lambda W_\times x_{t-1} \\
&= \lambda W_\times (p_\times - x_{t-1} + \lambda W_\times x_{t-1}) \\
&= \lambda W_\times (p_\times - \lambda W_\times x_{t-2} - p_\times + \lambda W_\times (\lambda W_\times x_{t-2} + p_\times) \\
&= \lambda W_\times (\lambda W_\times (\lambda W_\times x_{t-2} + p_\times - x_{t-2})) \\
&= (\lambda W_\times)^2 (\lambda W_\times x_{t-2} + p_\times - x_{t-2})
\end{aligned}$$

We find a pattern

$$\begin{aligned}
\implies x_{t+1} - x_t &= (\lambda W_\times)^t (\lambda W_\times x_0 + p_\times - x_0) \\
\text{And because } x_0 = p_\times & \\
&= (\lambda W_\times)^{t+1} p_\times
\end{aligned}$$

Which decreases only when

$$\xi_1 < 1$$

The largest magnitude eigenvalue of$\lambda W_\times$

$$\implies \lambda < |\xi_1|^{-1}$$

□

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:

$$k = O\left(\frac{\ln \epsilon}{\ln \lambda + \ln |\xi|}\right) \tag{14}$$

Finally, the fixed point iterations is more demanding than the others since it has a condition on the value of λ and thus assumes as geometric $\mu(k)$ function, and the number of iterations can easily vary a lot. The only advantage of this method is that it is very simple while still being better than the raw kernel.

verifier sur graphes

2.2.8 Spectral Decomposition

2.2.9 Nearest Kronecker Product Approximation

https://mathematica.stackexchange.com/questions/91651/nearest-kronecker-product
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.1924&rep=rep1&type=pdf[9]

2.3 Contributions

en anglais ça donne ?

3 Experiments

A big part of this project consisted in implementing the different algorithms from the main paper that I had studied [6], verifying the results obtained in that article, and making new attempts to accelerate those algorithms while keeping the best accuracy possible.

experiences : détailler db, tests, méthodes, les parametres, construction label, donner tableaux, resultats, teps de calculs, précision, discuter tout cela

3.1 Implementation

Implementing code from the main paper was a challenge in itself. Several problems aren't really documented and finding functions to solve them, or even explanations on how to solve them is sometimes difficult.

3.1.1 Graphs

The first thing that had to be done was to create a graph database generator, of very simple and standard graphs that had expectable behaviors and could thus be used for tests. Those graphs would also be altered using simple transformations, in order to create different graphs belonging to the same class.
For simplicity, it was decided to only use 3 classes to stay as simple as possible.

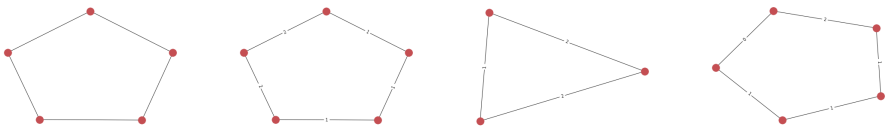

Figure 2: Ring graphs, resp. unlabelled, labelled, altered structured, altered labels

We studied 3 standard types of graphs : the ring, the star and the tree. A ring graph is a connected graph where each vertex is connected to exactly two vertices, it thus forms one big cycle. A star graph is composed of a central node, and of several other nodes that are all and only connected to the central node,

it should look like a star, or perhaps a flower.. Finally, a tree is a famous type of graph : it is a connected graph without any cycles, but here we will be studying a special type of tree, since each vertex is at most of degree 3.

Est-ce que je devrais dire ça ?

The database generator works as follows: for each type of graph, it will create

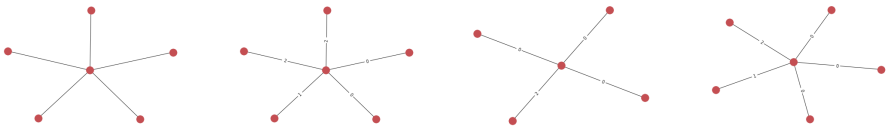

Figure 3: Star graphs, resp. unlabelled, labelled, altered structured, altered labels

a graph of predetermined size with randomly generated edge labels in a certain given interval. Then, for each graph, it will alter it a predefined number of times, and each time will create a new graph that has been both altered on structure and on labels. Alteration on structure involves removing or adding a predefined number of nodes in respect of the type of the graph thus staying in the same class as the source.

verifier si anglais, et c'est ok mettre deux points ?

1. Ring graphs are simply regenerated slightly longer or shorter randomly.
2. Star graphs are also regenerated either with more or less nodes, randomly.
3. Tree graphs are either expanded by adding randomly leaves anywhere in the graph, or reduced by randomly removing leaves.

Alteration on labels will randomly switch a predefined number of edge labels. Once this is done, the generator will create two databases out of the set of previously generated graphs, one will be made of the adjacency matrix of each graph without looking at labels, and the second one will be an array of adjacency matrices taken from graphs induced by selecting only edges with a specific label, and that for all labels in the label set.

predefined beaucoup répété, c'est un probleme ?

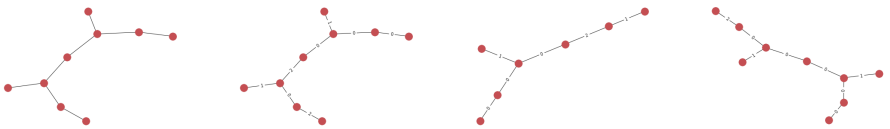

Figure 4: Tree graphs, resp. unlabelled, labelled, altered structured, altered labels

3.1.2 Raw Kernel

3.1.3 Inverse Kernel

3.1.4 Sylvester Equation

The Sylvester Equation is a good example of poorly documented problems (compared to the others). There are very few resources on the subject, and even less implementation of solvers. We could theoretically use the Sylvester Equation on both labeled and discrete-labeled graphs which are respectfully represented as Sylvester and Generalized Sylvester Equations (involving a sum of matrices), however, the only library available solves the first one, but not the second one. There seems to be confusing terminology since there is a solver for the Generalized Sylvester Equation, but a different one than the one described in our source[6]. The same source also mentions a way to solve the generalized version[10], however, this option wasn't explored.

correct ?

changé ça ?

Rajouter papier à citer

3.1.5 Conjugate Gradient Method

Note : Preconditioner random vector : $x \cdot x^T$ fonctionne bien
The Conjugate Gradient Method is however very well documented[11] and there are several libraries implementing this algorithm (we will be using scipy).

a faire

3.1.6 Fixed Point Iterations

Rapidité de la convergence : géométrique avec un ratio $\lambda 1/\lambda 2$??? si lambda trop proche de l'inverse de la valeur propre peut etre tres lent à converger quand la matrice d'adjacence est très dense
The Fixed Point Iterations method was very easy to implement and gave rapidly good results, however there is a constraint on the lambda(the geometric factor) used in the kernel as we have proved. Experiments showed that a lambda very close to the constraint increases significantly the time of computation to convergence.

image

3.1.7 Spectral Decompostion

can't inverse Left eigenvectors : `https://arxiv.org/pdf/1708.00064.pdf`

3.1.8 Nearest Kronecker Product

`https://www.sciencedirect.com/science/article/pii/S0377042700003939?via%3Dihub` `https://www.imageclef.org/system/files/CLEF2016_Kronecker_Decomposition.pdf` `http://dx.doi.org/10.1007/978-94-015-8196-7_17` `https://mathematica.stackexchange.com/questions/91651/nearest-kronecker-product`

3.2 Test of the random walk kernel

3.2.1 Accuracy comparison between labelled and unlabelled graphs

Même score, car donne même matrice ???

	Unlabelled	Labelled
Accuracy	?	?
Time	?	?

Table 1: Time and Accuracy of learning resp. for unlabelled and labelled graphs

3.2.2 Efficiency of alternate methods

	Raw kernel	Inverse Kernel	Sylvester Equation	Conjugate Gradients	Fixed points	Spectral Decomp.	Nearest Kronecker Product
Accuracy	67%	67%	?	67%	67%	?	?
Time	18.40s	5.72	?	7.62s	5.94s	?	?

Table 2: Time and Accuracy of learning for the raw kernel and other methods

3.2.3 Experiments on a biology dataset

`https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets`
`http://members.cbio.mines-paristech.fr/~nshervashidze/code/` enzymes

3.3 Comparison of kernels

3.3.1 Label use

Comparison using labels and no labels

3.3.2 Comparison of their gram matrices

The algorithms approximate the kernel using very different techniques, thus the gram matrices obtained from the kernels were similar in appearance, but had significant differences of scale. It was then decided to normalize very simply the gram matrices so they could be more easily compared.

$$M = (M - min(M))/max(M) \tag{15}$$

Afterwards, in order to verify all the algorithms gave similar results, it was decided to compute the Frobenius norm of the difference of two gram matrices (divided by the size of the matrix, thus giving the mean standard deviation). The following results were obtained and were satisfying. Indeed, the

correct nom ?

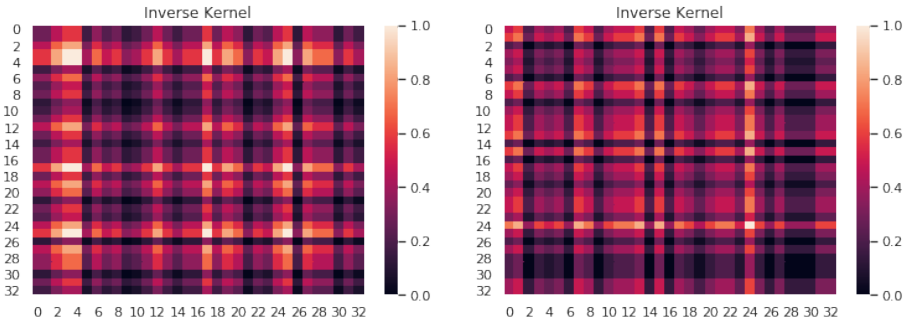

Figure 5: Two gram matrices computed using the Inverse Kernel on different datasets

	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 3: Approximate standard deviation of matrix entries (?)

3.3.3 Complexity and Accuracy

3.3.4 title

3.4 Individual kernel analysis ?

mettre ici les images variations degré d'approximation

3.5 Nearest Kronecker product ?

3.6 On molecules

3.7 Improvements

4 Conclusion and Future Work

experiences : détailler db, tests, méthodes, les parametres, construction label, donner tableaux, resultats, teps de calculs, précision, discuter tout cela section 4 : 1 page ou page et demi : conclusion et discussion

A Appendix

B Annex 1

C Acknowledgements

This work was done during the first year of my master.

List of Figures

List of Tables

References

[1] Wikipedia, "Graph theory - Wikipedia," 2019.

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

[3] H. Kashima, K. Tsuda, and A. Inokuchi, "Marginalized kernels between labeled graphs," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pp. 321–328, AAAI Press, 2003.

[4] 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.

[5] N. Shervashidze, *Scalable graph kernels*. Dissertation, Universität Tübingen, 2012.

[6] 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.

[7] W. Imrich and S. Klavžar, *Product graphs*. Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley-Interscience, New York, 2000. Structure and recognition, With a foreword by Peter Winkler.

[8] Wikipedia, "Kronecker product," Mar. 2019. Page Version ID: 887195556.

[9] C. F. Van Loan and N. Pitsianis, *Approximation with Kronecker Products*, pp. 293–314. Dordrecht: Springer Netherlands, 1993.

[10] L. De Lathauwer, B. De Moor, and J. Vandewalle, "Computation of the canonical decomposition by means of a simultaneous generalized schur decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 26, no. 2, pp. 295–327, 2004.

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