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

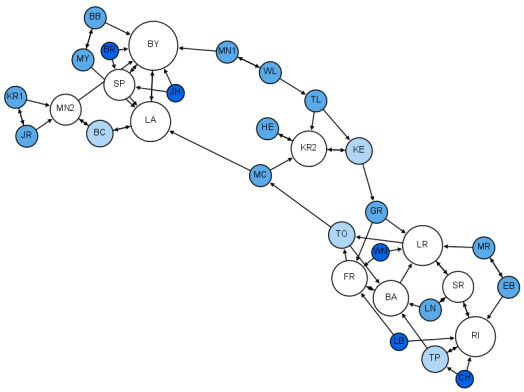
SCIENCES SORBONNE UNIVERSITÉ

Outline

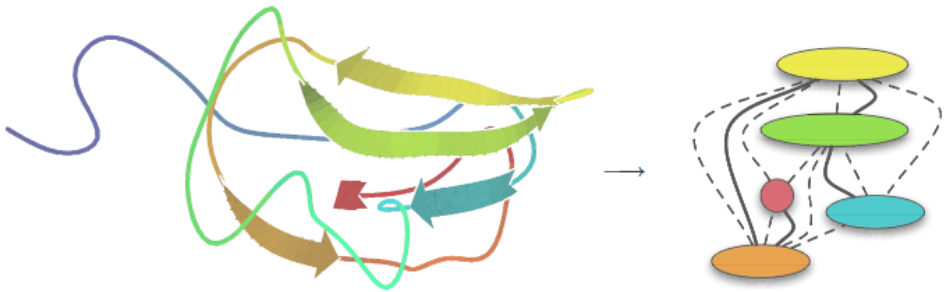
1. Motivation and Objectives
2. Methodology
3. Experiments
4. Conclusion
   - References

Motivation and Objectives I

- Data as graphs : proteins, social networks, ..
- Big Data : many, large graphs
- We want to tackle the problem of classifying them



Moreno's sociogram (Source : Wikipedia)



Fragment of a protein transformed into a graph (Vishwanathan et al., 2010)

Motivation and Objectives II : Current methods

Support Vector Machine (Cortes and Vapnik, 1995)

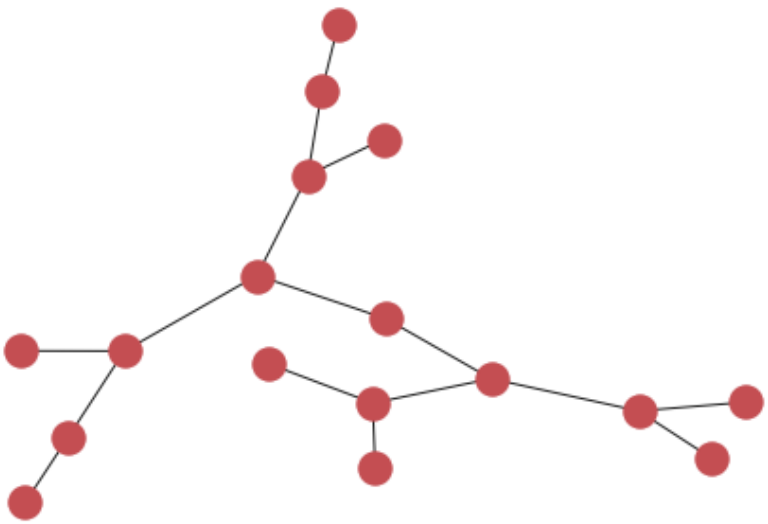
- Operates on vectorial data
- Great accuracy
- Great capacity of generalization
- Allows the use of kernels in its dual form

Kernels

- Maps data to higher dimensions
- Allows functions to replace the dot product
- Improves the accuracy of SVMs

Motivation and Objectives III

- These methods are adapted to **vectorial data**
- Graphs are not
- Vectorizing adjacency matrices does not solve the problem (automorphism, size)
- New types of kernels were discovered
- They are very **complex** to compute



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

Outline

1. Motivation and Objectives
2. Methodology
3. Experiments
4. Conclusion
   - References

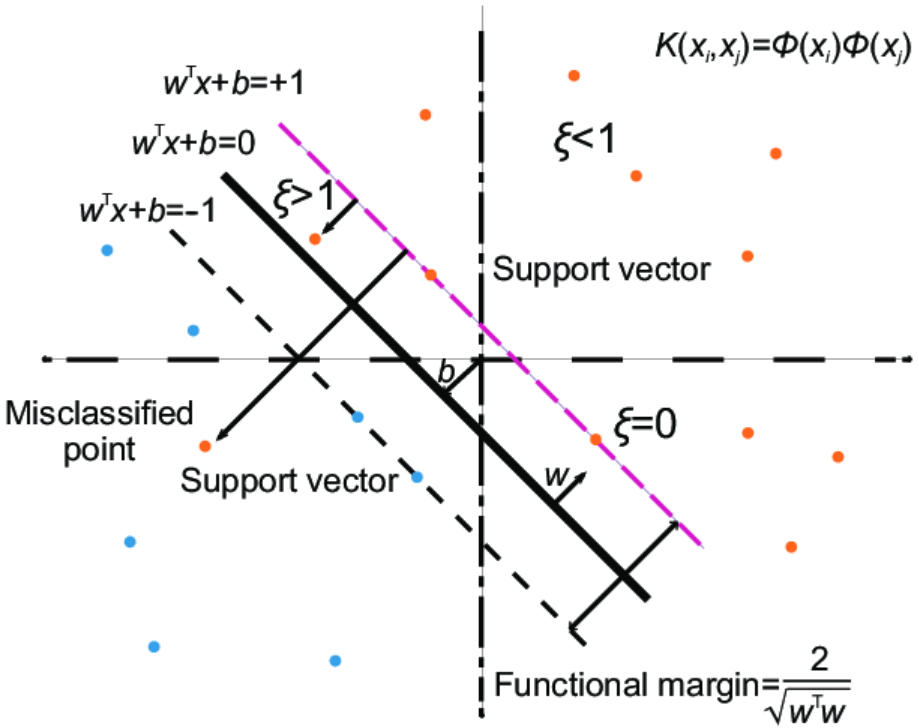
Support Vector Machines

- We have a set of pairs $\{(\mathbf{x_i}, y_i)\}$ of size N where $y_i \in \{-1, +1\}$
- We want to maximize the margin, and to minimize the generalization error
- We thus minimize

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t.} \quad y_i(\mathbf{x_i} \cdot \mathbf{w} + w_0) \geq 1$$

$$1 \leq i \leq N$$



Soft-margin SVM (Ma et al., 2017)

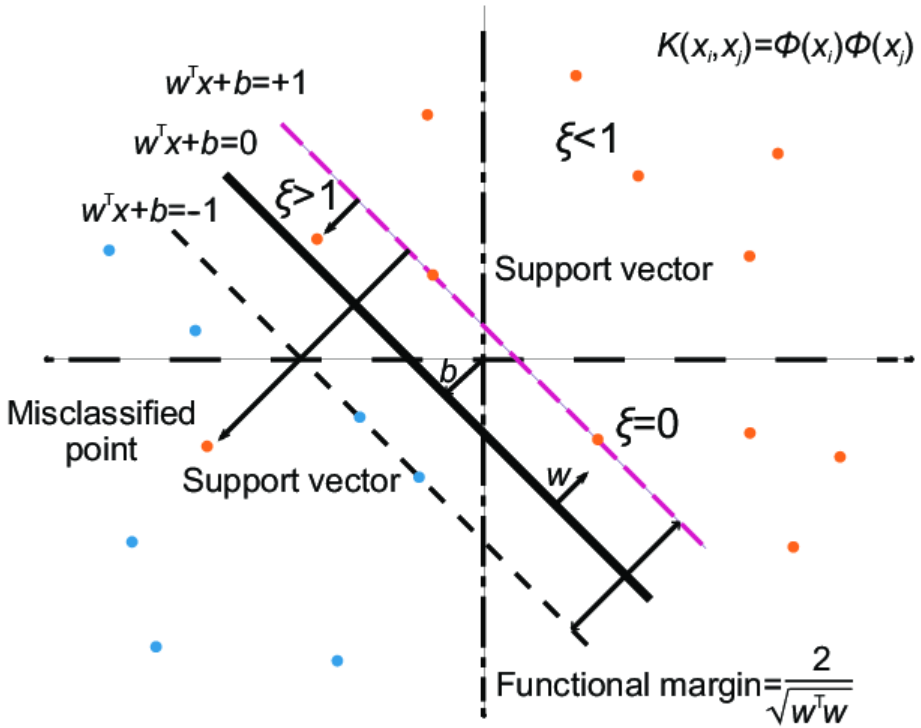
Support Vector Machines

- We have a set of pairs $\{(\mathbf{x_i}, y_i)\}$ of size N where $y_i \in \{-1, +1\}$
- We want to maximize the margin, and to minimize the generalization error
- We thus minimize

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{s.t.} \quad y_i(\mathbf{x_i}\cdot\mathbf{w} + w_0) \geq 1 - \xi_i$$

$$1 \leq i \leq N \qquad \xi_i \geq 0$$

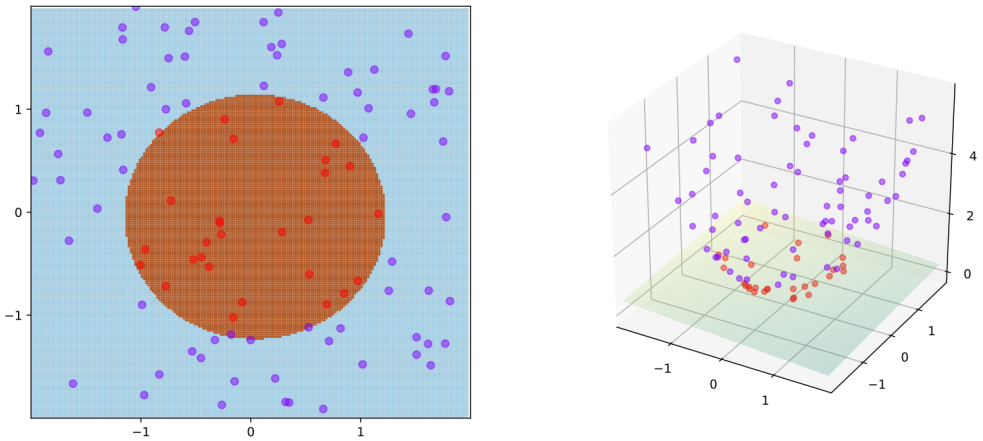


Soft-margin SVM (Ma et al., 2017)

Kernels

- Dual version of the SVM

$$\max_{\mathbf{w}} \sum^{n} \alpha_i - \frac{1}{2} \sum^{n} \sum^{n} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^\top \mathbf{x_j}$$



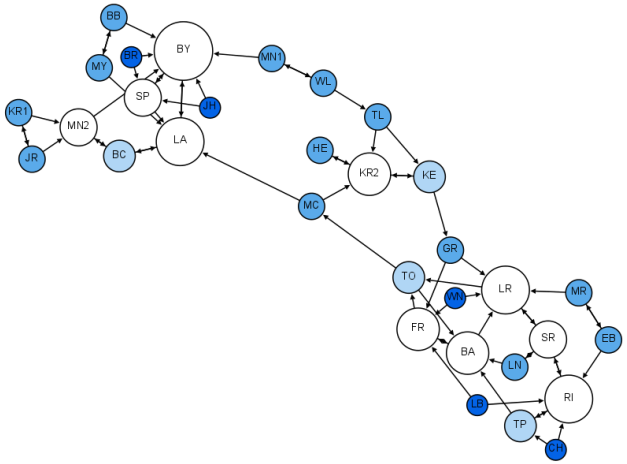
Kernel Trick (Source : Wikipedia)

- A map ϕ can be used
- The dot product can be replaced by a p.s.d function $\kappa(\mathbf{x_1}, \mathbf{x_2})$
- RBF $= e^{-\frac{\|\mathbf{x_1} - \mathbf{x_2}\|^2}{2\sigma^2}}$
- Polynomial $= (\mathbf{x_1} \cdot \mathbf{x_2} + c)^d$

Graph Kernels

We want to classify non **vectorial data**. There are two types of methods available :

- Kernels on graphs
- Kernels on graph nodes



Moreno's sociogram (Source : Wikipedia)

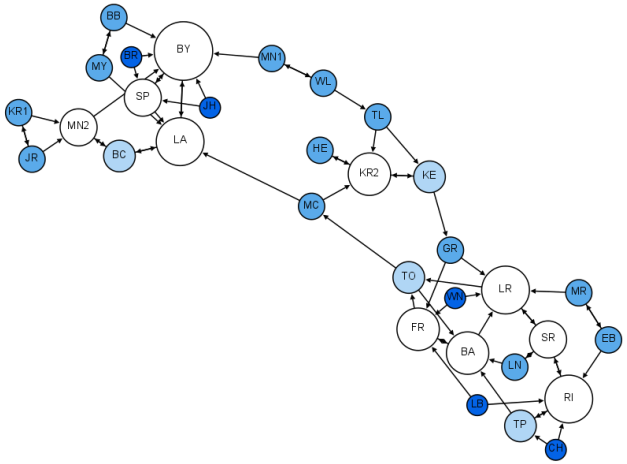
Graph Kernels

We want to classify non **vectorial data**. There are two types of methods available :

- Kernels on graphs
- Kernels on graph nodes

Some graph kernel families :

- Graphlets (Shervashidze et al., 2009)



Moreno's sociogram (Source : Wikipedia)

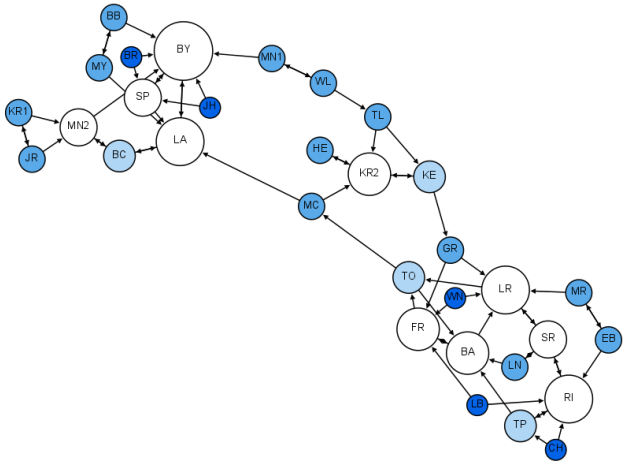
Graph Kernels

We want to classify non **vectorial data**. There are two types of methods available :

- Kernels on graphs
- Kernels on graph nodes

Some graph kernel families :

- Graphlets (Shervashidze et al., 2009)
- Shortest paths (Borgwardt and Kriegel, 2005)



Moreno's sociogram (Source : Wikipedia)

Graph Kernels

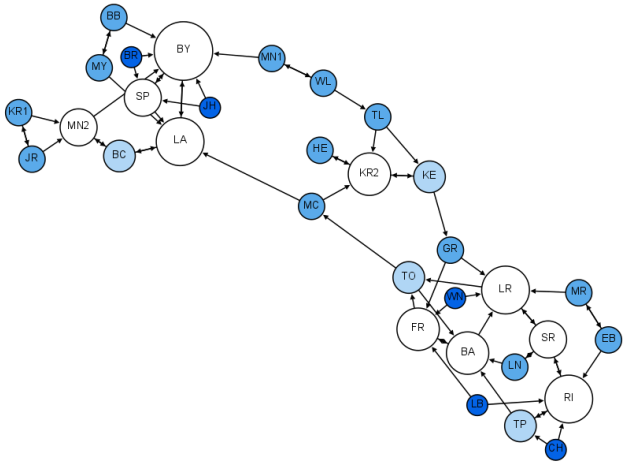
We want to classify non **vectorial data**. There are two types of methods available :

- Kernels on graphs
- Kernels on graph nodes

Some graph kernel families :

- Graphlets (Shervashidze et al., 2009)
- Shortest paths (Borgwardt and Kriegel, 2005)
- **Random walks** (Vishwanathan et al., 2010)

$\Longrightarrow$ The complexity is too large.

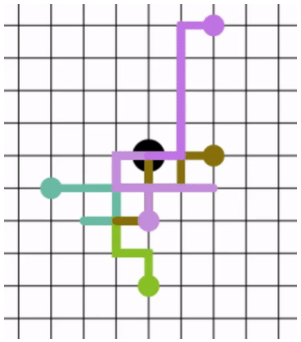


Moreno's sociogram (Source : Wikipedia)

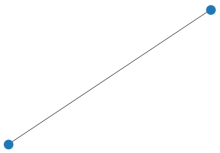
Random Walks I

Random walks on undirected graphs

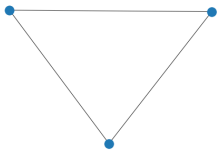
- Starts at a vertex
- Randomly picks an edge and moves to the next vertex
- Repeats it for a finite (or not) number of steps



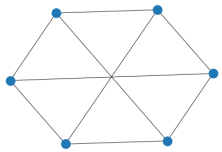
2D Random walk (Source : Wikipedia)



A graph G_1



A graph G_2



$G_\times = G_1 \otimes G_2$

- Computing a random walk on a product graph is equivalent to computing a common walk on both graph (Imrich and Klavzar, 2000)
- The product graph is computed using the Kronecker product $\otimes$

Random Walks II

- Unlabeled product graph : $W_\times = A_1 \otimes A_2$
- Labeled product graph : $W_\times = \sum_{l=1}^{d} A_1^{(l)} \otimes A_2^{(l)}$

$$\kappa(G_1, G_2) = \sum_{k=0}^{\infty} \mu(k) q_\times^\top W_\times^k p_\times$$

- $p_\times$ and $q_\times$ resp. start and end probabilities : priors $\Longrightarrow$ uniform probabilities
- $\mu(k)$ weighting function of walks length

Acceleration methods

Inverse Kernel

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

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

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

Sylvester Equation

- Applicable to labeled and unlabeled graphs
- Restricted to $\mu(k) = \lambda^k \quad \Longrightarrow$ inverse kernel

Method

The Sylvester equation we solve is the following

$$\sum_{l=1}^{d} A_1^{(l)} X A_2^{(l)} + C = X$$

by multiplying X to $q_\times$ we get

$$q_\times^\top \text{vec}(X) = q_\times^\top (I - \lambda W_\times)^{-1} p_\times$$

- The complexity is $O(n^3)$ for unlabeled graph
- It is yet unknown for labeled graphs (Vishwanathan et al., 2010)

Conjugate Gradient

- Applicable to unlabeled and labeled graphs
- Restricted to $\mu(k) = \lambda^k \quad \Longrightarrow$ inverse kernel
- The matrix should be symmetrized

Method

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

- The complexity is $O(rdn^3)$ for r iterations and d labels
- Efficient if the matrix has a small effective rank r

Fixed Point Iterations

- Applicable to labeled and unlabeled graphs
- Restricted to $\mu(k) = \lambda^k \quad \Longrightarrow$ inverse kernel

Method

The algorithm iterates on the following function

$$x_{t+1} = p_\times + \lambda W_\times x_t$$

Either until $\|x_{t+1} - x_t\| < \epsilon$ or a maximum number of iterations is reached.

- The complexity is $O(kdn^3)$ for k iterations and d labels
- Convergence requires $\lambda < \frac{1}{\xi_1}$

Spectral Decomposition I

- Currently applicable only to unlabeled graphs
- No restrictions on $\mu(k)$

Method

- Eigendecomposition of the adjacency matrix

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

Spectral Decomposition I

- Currently applicable only to unlabeled graphs
- No restrictions on $\mu(k)$

Method

- Eigendecomposition of the adjacency matrix

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

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

Spectral Decomposition I

- Currently applicable only to unlabeled graphs
- No restrictions on $\mu(k)$

Method

- Eigendecomposition of the adjacency matrix

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

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

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

Spectral Decomposition II

Inverse Kernel

- By choosing $\mu(k) = \lambda^k$, this method computes the inverse kernel
- Inverse becomes trivial to compute on diagonal matrix

$$\kappa(G, G_2) = q_\times^\top V_\times (I - \lambda D_\times)^{-1} V_\times^{-1} p_\times$$

- The complexity is $O(pn^3)$ with p which depends on the complexity of $\mu(k)$
- It is even lower since the eigendecomposition of each graph is computed only once
- Currently our most efficient method on unlabeled graphs

Nearest Kronecker Product Approximation

- Labeled-graph kernel computation can be turned into an unlabeled one with some loss in accuracy, but gain in computation time
- The idea is to approximate two matrices A and B such that $\|W_\times - A \otimes B\|_F$ is minimized
- Computed in $O(dn^2)$ time (Van Loan and Pitsianis, 1993)
- All methods such as Spectral Decomposition can then be applied
- The overall complexity remains unchanged

Outline

1. Motivation and Objectives
2. Methodology
3. Experiments
4. Conclusion
   - References

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 Dataset

- MUTAG : 188 chemical compounds, 2 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 Dataset

- MUTAG : 188 chemical compounds, 2 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
  $(I - \lambda W_\times)x = p_\times$

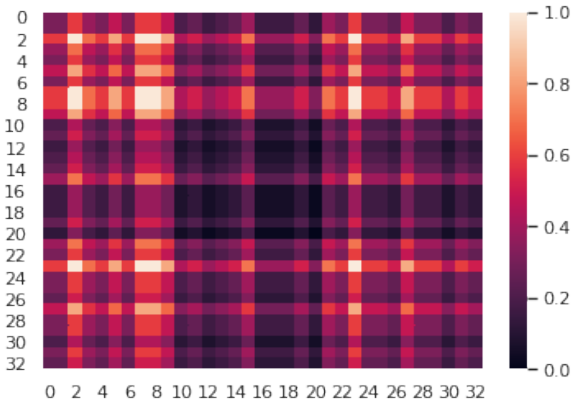
Implementation

- 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
  $q_\times^\top P_\times (I - \lambda D_\times)^{-1} P_\times^{-1} p_\times$

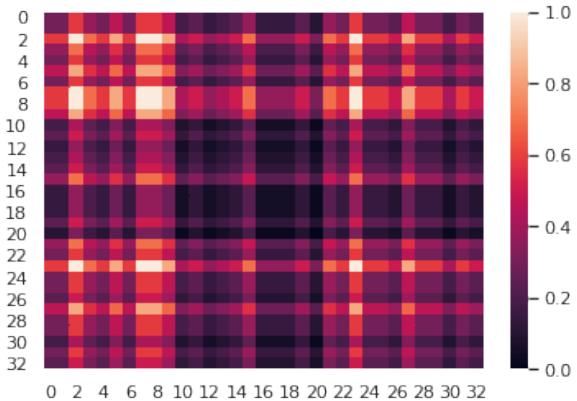
Performance : Gram matrices

	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

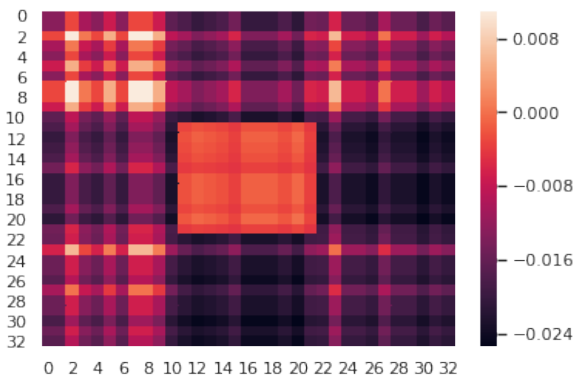
MSE of matrix entries normalized per entry



Raw Gram matrix

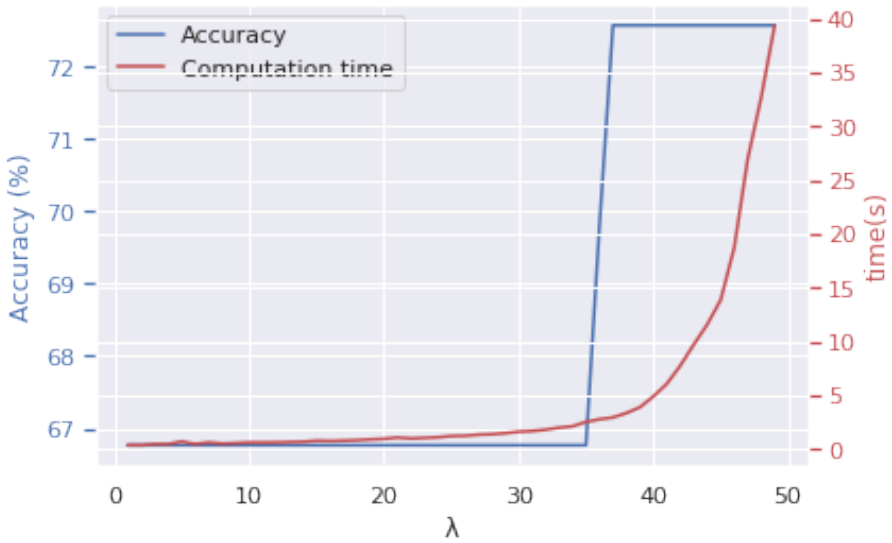


Fixed Point matrix

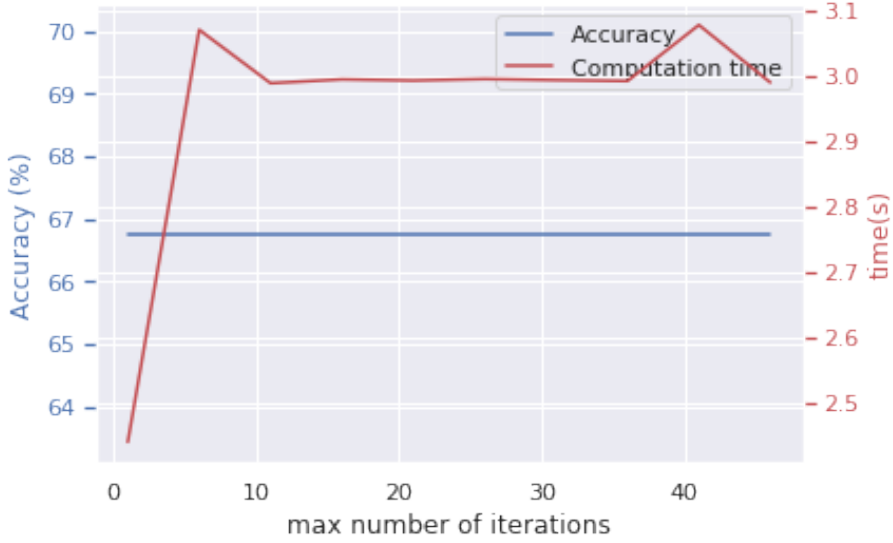


Difference

Parameter tuning

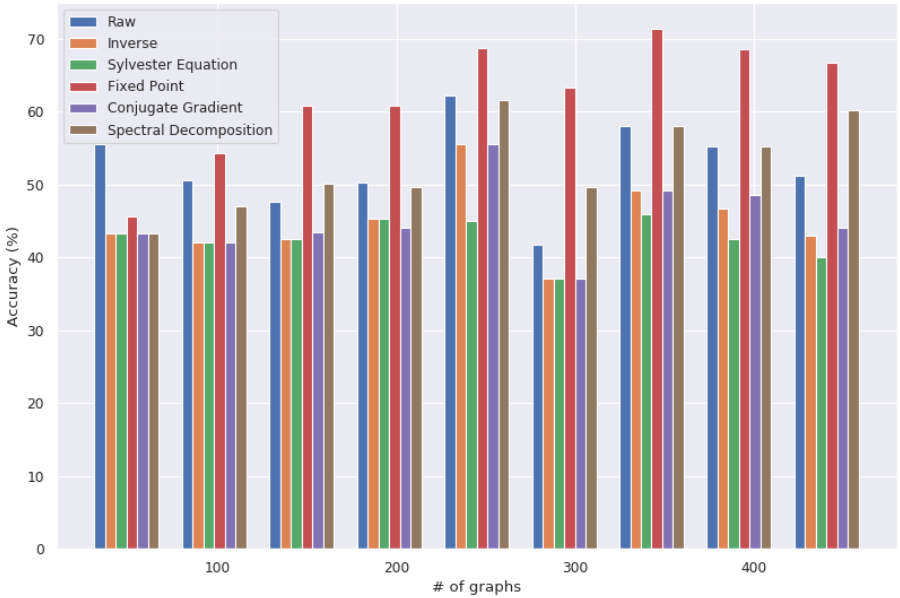


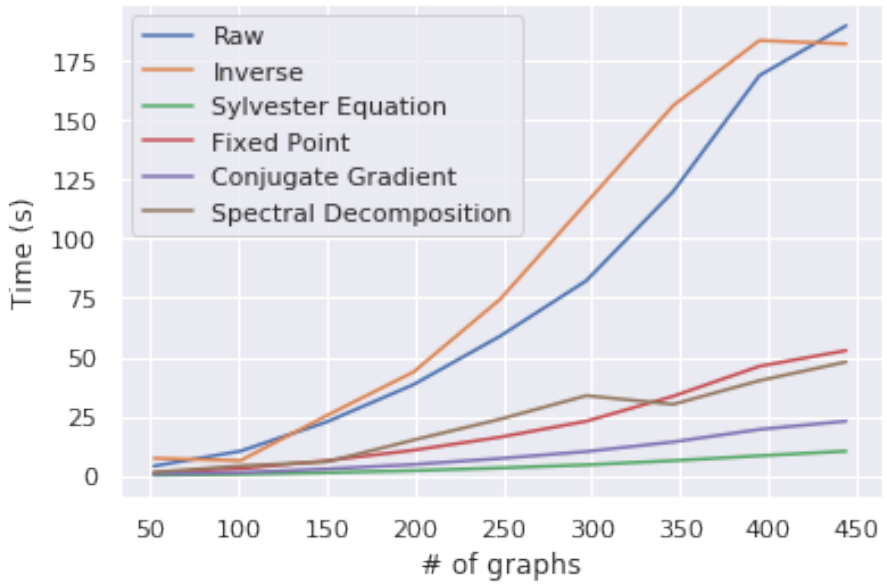
Accuracy and computation time for the fixed point method depending on λ



Accuracy and computation time for the conjugate gradient method depending on the maximum number of iterations

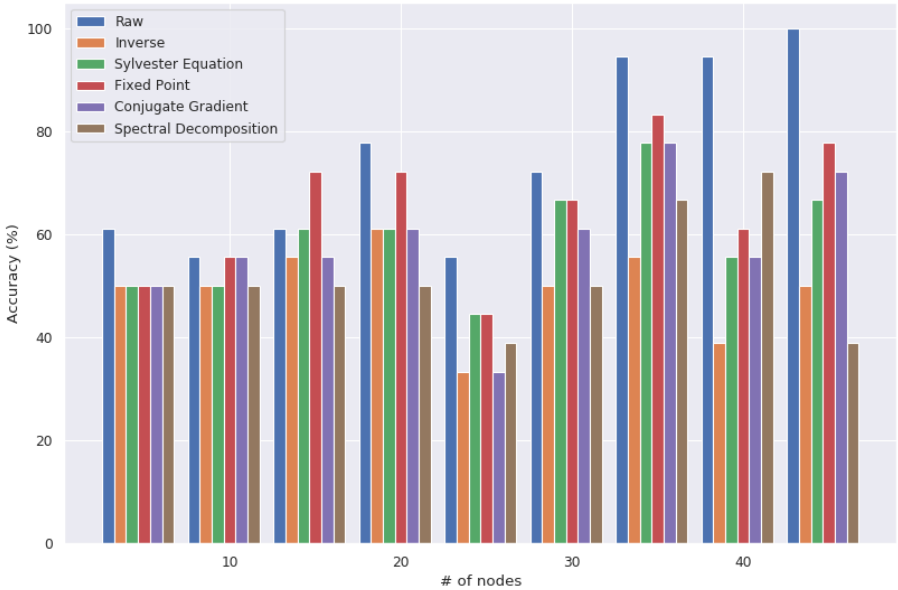
Performance : Complexity I

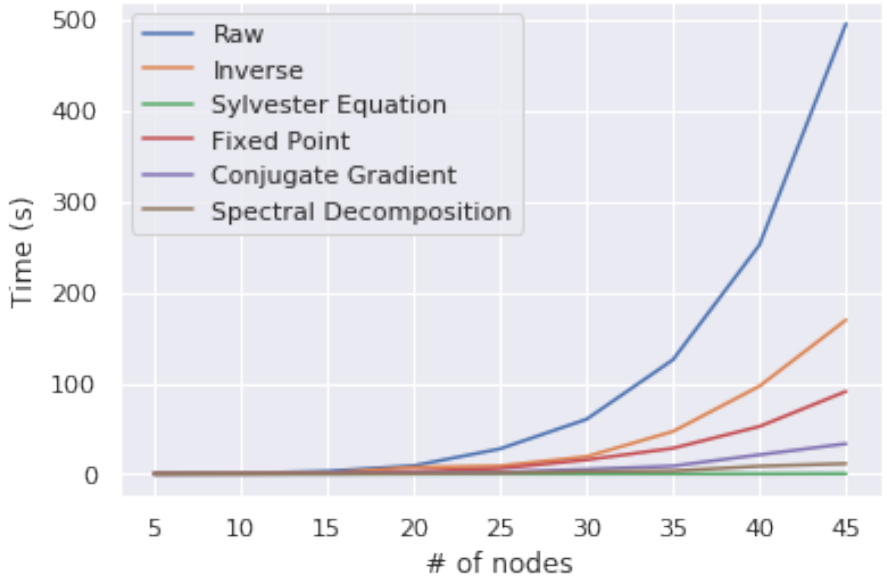




Accuracy and Computation time of different methods depending on the number of graphs

Performance : Complexity II





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

MUTAG

Method	MUTAG	
	Accuracy	Comp. time
Raw Kernel	83.9 ± 3.1	1'54"
Sylvester Equation	83.5 ± 2.8	5"
Conjugate Gradient	83.5 ± 2.8	18"
Fixed Point Iterations	83.5 ± 2.8	47"
Spectral Decomposition	82.4 ± 1.3	43"

Accuracy and Computation time on the dataset

Conclusion

Conclusion

- Random walks : decent accuracy
- Several acceleration methods were introduced
- Performances are greatly improved
- Performances on MUTAG are on par with the state of the art

Future work

- Generalizing the Spectral Decomposition to labeled graphs

References

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. [Online]. Available: http://www.jmlr.org/papers/v11/vishwanathan10a.html

C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: https://doi.org/10.1007/BF00994018

Y. Ma, W. Chen, X. Ma, J. Xu, X. Huang, R. Maciejewski, and A. Tung, "Easysvm: A visual analysis approach for open-box support vector machines," *Computational Visual Media*, vol. 3, 03 2017.

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*, Apr. 2009, pp. 488–495. [Online]. Available: http://proceedings.mlr.press/v5/shervashidze09a.html

K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," in *Fifth IEEE international conference on data mining (ICDM'05)*. IEEE, 2005, pp. 8–pp.

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

C. F. Van Loan and N. Pitsianis, "Approximation with kronecker products," in *Linear algebra for large scale and real-time applications*. Springer, 1993, pp. 293–314.

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

Appendix : Graphs

Definition

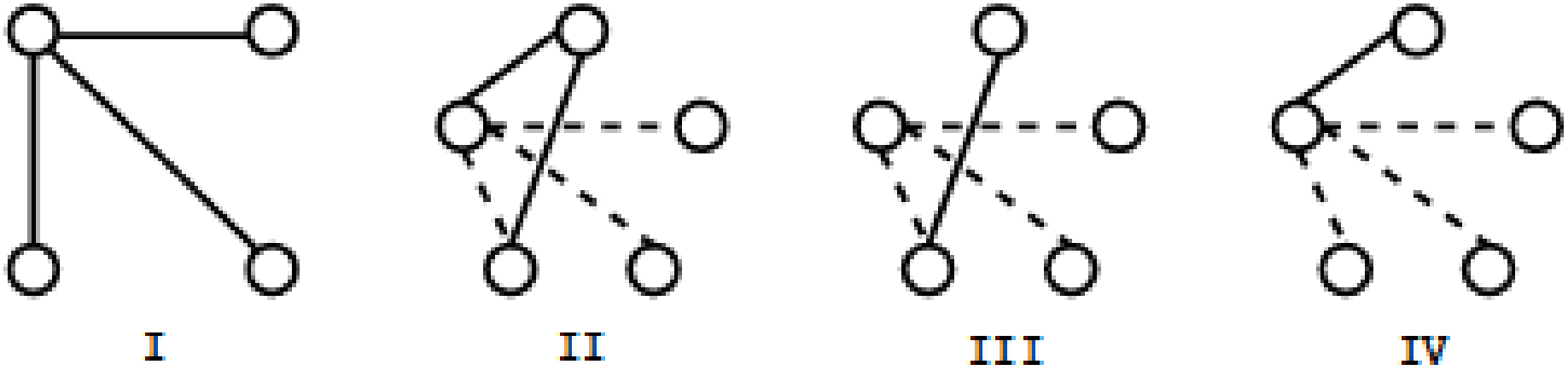
A graphBondy et al. (1976) is a type of mathematical structure that represents connections between objects. It is more precisely an ordered pair $G = (V, E)$ of two sets: vertices V (or nodes) and edges E that connect two vertices together.

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

Properties

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

Appendix : Graphlets



Some graphlets of size 4 and 5 (Shervashidze et al., 2009)

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

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

Appendix : P.S.D

Definition

A kernel K is positive semi definite if and only if

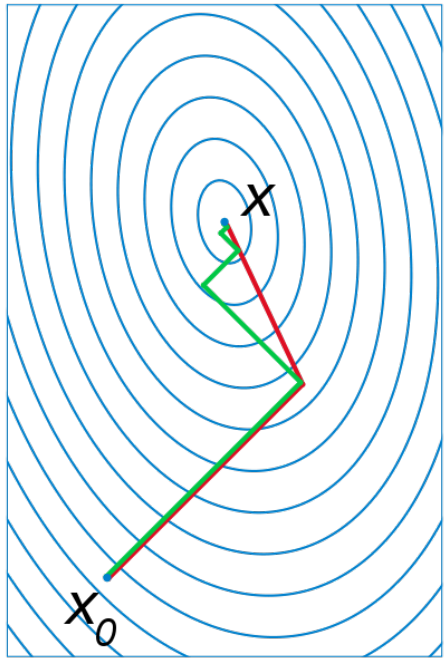
$$\sum_{i=1}^{n}\sum_{i=1}^{n}\kappa(\mathbf{x_i},\mathbf{x_j})c_ic_j \geq 0 \qquad \forall i \in \{1..n\} \quad c_i \in \mathbb{R} \tag{1}$$

It is also p.s.d if its gram matrix have non-negative eigenvalues.

Appendix : Conjugate Gradient

- The idea is to make the new gradient orthogonal to the former

- Convergence guaranteed in n steps to solve $Ax = b$ where $A \in R^{n \times n}$



Conjugate gradient (red) compared to Gradient Descent (green)