Kernel Methods on the Graph Coloring Problem

An exploration of Graph Kernels to solve k-Coloring

Marc Gàllego, Fernando Gastón & Aleix Seguí

January 2021

Course on Machine Learning 2
Universitat Politècnica de Catalunya
Grau en Ciència i Enginyeria de Dades

Contents

1 Introduction

The main goal of this project is to design a Machine Learning model that predicts the chromatic number of a graph.

A graph is a mathematical structure formed by a pair $G = (V, E)$ of sets such that $E \subseteq [V]^2$, that is the elements of E are 2-element subsets of V. We call V the set of vertices and E the set of edges. Given an edge $e = (v, w)$ we say v and w are adjacent.

A vertex colouring of G is a map $c : V \mapsto N$ such that every pair of adjacent vertices v and w satisfies $c(v) \neq c(w)$. The chromatic number of G, $\chi(G)$, is the smallest integer such that a colouring $c : V \mapsto \{1, ..., \chi(G)\}$ exists.

Finding the chromatic number of a graph is a well-known NP-complete problem and as such no efficient way of computing it has been found. That's why we thought it would be interesting to try to tackle this problem using Machine Learning techniques. Kernel methods have proven to be very useful in solving graph-related problems. Here, we explore their performance applied to the chromatic number calculation.

1.1 Previous work

As previously mentioned, finding the chromatic number of a graph is an NP-complete problem and as such can only be done using exponential algorithms. However, many greedy colouring algorithms have been proposed that have a polynomial running time and achieve pretty good results.

There have also been several attempts at solving this problem using Machine Learning. In particular, Henrique Lemos et. al [1] used Deep Learning and achieved a very high accuracy by using a Graph Neural Network with a simple architecture. However, they didn't use kernel methods for feature extraction. Hence, our proposal is novel in the field.

2 Graph Kernels and Kernelized methods

In this project we have used three Graph Kernels to kernelize a SVM and to implement kernelPCA. These kernels are the Shortest-Path Kernel, the Geometric

Random Walk kernel and the Graphlet Kernel.

2.1 Shortest-Path Kernel

Given a graph G we can transform it into a weighted graph S via the Floyd-Transformation. S will have the same set of nodes as G. There will be an edge from v to w if w is reachable from v and its weight will be the length of the shortest path between v and w. The shortest-path between all nodes can be found in $\mathcal{O}(n^3)$ using Floyd's algorithm, hence the name of the transformation.

Given two graphs G and G' with Floyd transforms $S = (V, E)$ and $S' = (V', E')$, the Shortest-Path Kernel is defined in [2] as:

$$k_{Shortest-Path} = \sum_{e \in E} \sum_{e' \in E'} k^{(1)}_{walk}(e, e')$$

where $k^{(1)}_{walk}$ is a kernel on edge walks of length 1.

2.2 Geometric Random Walk Kernel

Given two graphs G and G' and their tensor product $G_\times$ the Geometric Random Walk (GRW) Kernel is defined in [3] as:

$$K(G, G') = \sum_{p,q=1}^{|V_\times|} [\sum_{l=0}^{\infty} \lambda^l A^l_\times]_{pq}$$

Notice that performing a random walk on $G_\times$ is equivalent to performing a simultaneous random walk on G and G'.

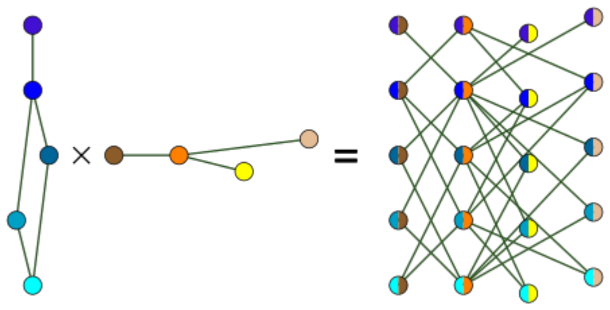

Figure 1: *Example of the tensor product of graphs. [4]*

2.3 Graphlet Kernel

A graphlet is a connected non-isomorphic induced subgraph of a large network.

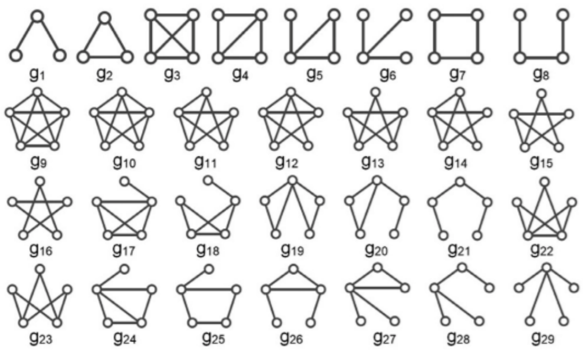

Figure 2: *Graphlets of size 3, 4 and 5. [5]*

Given a graph G, the frequency of a certain graphlet g_i, $f(g_i)$, is the number of ocurrences of that graphlet in G. Then, we can represent G using a vector $v_G = [f(g_1), f(g_2), ..., f(g_k)]$ where k is the number of graphlets considered. Given two graphs G and G', the graphlet kernel is defined in [5] as:

$$k_{Graphlet}(G, G') = v_G^t \cdot v_{G'}$$

All of these kernels are implemented in the `graphkernels` package in R [6].

2.4 Kernelized SVM

In this project we will be using the multi-class formulation of the SVM which supports kernelization using any of the kernels proposed above. We will also use a Support Vector Regression, in particular we will be using the ϵ-SVR formulation, implemented in the `ksvm` function from the `kernlab` package [7]. Each method has an associated hyperparameter: C for the case of classification and ϵ for regression.

2.5 Kernel PCA

Finally, we will also be using the kernelized version of PCA to obtain a visualization of the graphs in a low dimensional space. In the `kernlab` package it is implemented in the `kpca` function.

3 Dataset Generation

In order to generate a realistic dataset that can approach graphs from the real world, we have used 3 well known methods that create graphs with different statistical properties. Those are the following:

- **Erdös-Rényi (ER):** Generates a random graph based on two parameters: n and p, which are the number of vertices and the probability (between 0 and 1) that an edge exists over a pair of vertices, respectively. Each edge is added or not completely independently from the others.
  This results in graphs with small (logarithmic on n) diameters, which means that the longest shortest-path is small. They also present small clustering coefficients.
- **Watts–Strogatz (WS)**: It departs from a graph of n vertices arranged in a ring, where two of them are connected if they are at distance nei or less. Then, with probability p each edge is rewired to a randon vertex. This results in graphs with low diameter and high clustering coefficients.
- **Barabási-Albert (BA)**: This model tries to emulate real social networks and it is based on the principle that "rich get richer". That means that vertices with a high degree are more likely to receive new edges. That implies the presence of "hubs", vertices with very high degrees.

A third of our dataset has been generated with each method. In total we have 365 observations, as we wanted to make a balanced dataset: that is 90 observations for each colorability from 3 to 6. The other 5 observations come from some graphs that are 7-colorable that we have decided to keep. We have not included graphs that can be colored using 1 or 2 colors as that problem can be solved in polynomial time.

Our dataset only includes graphs with a number of vertices that ranges from 6 to 14, as we needed small enough graphs in order to be able to compute its chromatic number, which is solved in exponential time. We have also obtained the chromatic number using a greedy algorithm, in order to be able to compare our results with alternative solutions. Those numbers will act as the labels for our models.

We have used a Python script in order to create the dataset, using the `igraph` library for the generation, the `grinpy` library for the exact calculation of the chromatic number. The greedy implementation used is the `greedy_color()` function from the `networkx` package.

In order to ensure the correctness of the dataset we check that all the graphs generated are connected and that a new graph is not isomorphic to any of the ones already created. In order to do this last check we use the `isomorphic` method provided by the `igraph` package. The exact implementation can be found in the `data_gen.py` script.

4 Exploration

The dataset we generated contains x graphs and the chromatic number of these graphs range from 3 to 6. The greedy algorithm inplemented in the Python library NetworkX achieved an accuracy of around 90.4%.

4.1 Kernel Principal Components Analysis

What we wanted to do is visualize our dataset using kernel PCA which will allow us to plot the dataset in a low dimensional space. We will use all three graph kernels we previously showed to see how different are the features extracted by each of them. In parentheses, the variable that the colors correspond to is shown in the title of each plot.

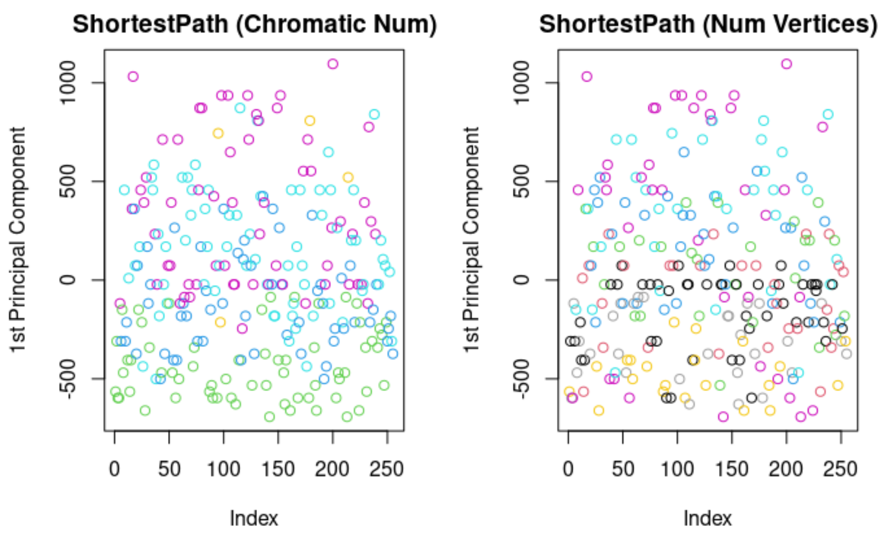

Figure 3: *kPCA using Shortest-Path Kernel. The full variance is explained by the vertical component.*

The Kernel PCA using the Shortest-Path kernel only gave one Principal Component. The 1st Principal Component is given by the ordering of the records. However, we are able to see some patterns within the structure of the 2nd Component. Graphs with chromatic number 3 (green) are at the bottom while graphs with higher chromatic number such as 6 and 7 (purple and yellow) are at the top.

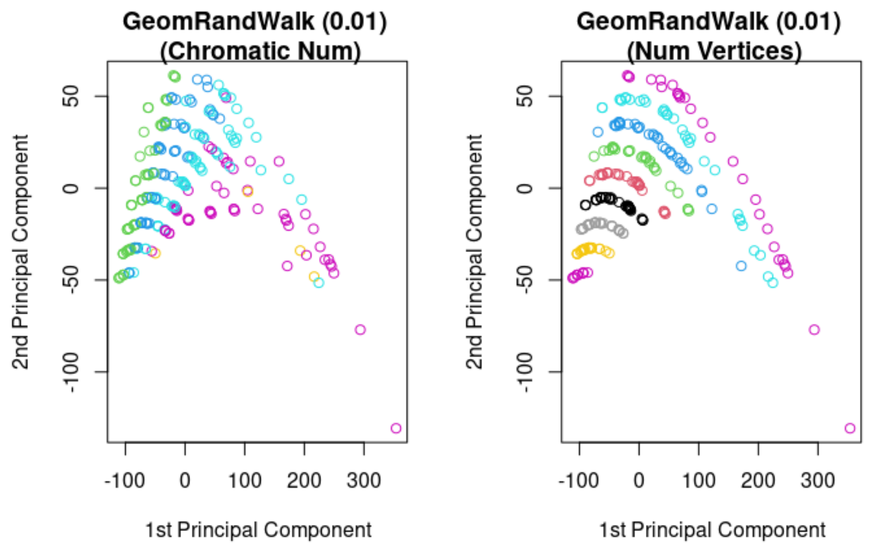

Figure 4: *kPCA using GRW with* $\lambda = 0.01$*. The first 2 principal components explain 95% of the variance.*

When looking at the principal components for the Geometric Random Walk with parameter 0.01, things get interesting. We can clearly see in the rightmost plot in Figure 4 that the graphs are arranged in curves that depend on the number of vertices of the graph. What's more interesting though is the patterns that we can see in the third plot of that same figure. It is very apparent that the graphs with the same chromatic number are also arranged in a pattern. This pattern though becomes more apparent when we look at Figure 5.

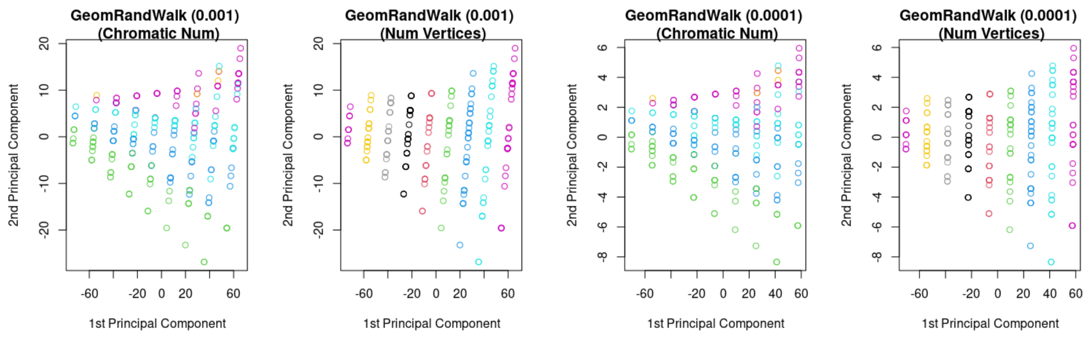

Figure 5: *kPCA using GRW (*$\lambda = 0.001$ *and 0.0001). The first two components explain 99% of the variance in both cases.*

In Figure 5 we have the Principal Components using the GRW Kernel with parameters 0.001 and 0.0001. The results achieved with these values are not very different between them, but they also show the patterns in Figure 3. In fact, these become more apparent and they become straight lines and show near-linear separability between graphs with different chromatic number.

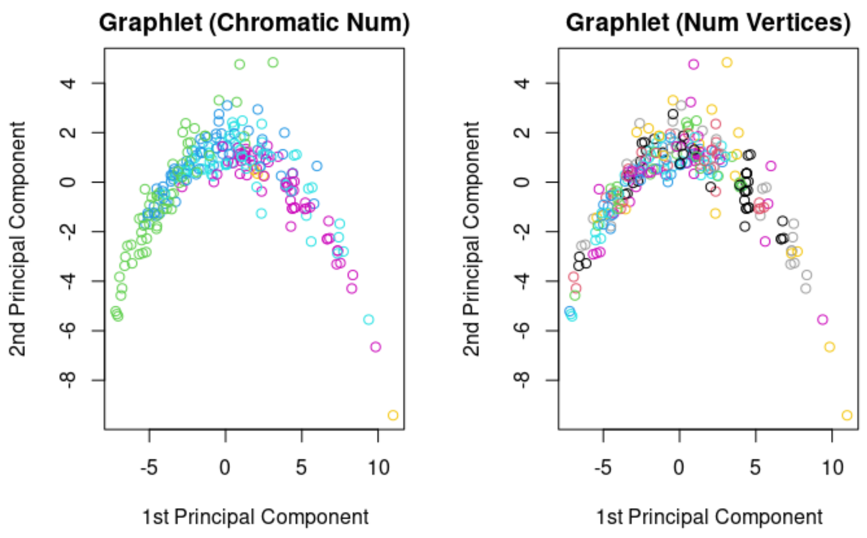

Figure 6: *kPCA using Graphlet Kernel. The components explain 94% of the variance.*

Here we see the kernel Principal Components using the Graphlet Kernel. We are also able to see some patterns relating to the chromatic number of the graphs, mainly using the 1st Principal Components but it's far from being as apparent as the components found using the Geometric Random Walk.

Given these results, it is not daring to say that the selected kernels are able to extract meaningful information about the structure of the networks that will help in classifying the chromatic numbers of the graphs.

5 Results

We split our dataset into a train and a test set, 70% and 30% respectively. We use the train set to perform 3-fold Cross-Validation (CV) in order to see which of the methods achieve better results and to choose the best combination of hyperparameters in both the (parametrized) kernel methods and the SVM. Finally, we assess the generalization performance of the selected methods using the held-out test set. All the metrics shown in the tables have been estimated using the 3-fold CV.

5.1 Regression

First, we treated the problem as a regression task, using Support Vector Regression, ϵ-SVR in particular. We use the Root Mean Square Error (RMSE) as the metric.

Table 1: *RMSE of every pair Kernel-hyperparameter, in the Chromatic number regression.*

Kernel	$\epsilon = 1$	$\epsilon = 0.1$	$\epsilon = 0.01$
Shortest Path	0.794	**0.776**	0.784
GRW ($\lambda = 0.01$)	0.568	0.567	**0.564**
GRW ($\lambda = 0.001$)	0.605	0.588	0.583
GRW ($\lambda = 0.0001$)	0.702	0.614	0.605
Graphlet	0.857	0.811	0.819

When we round the predicted value, we get the estimation of the chromatic number. We can compute accuracy for this prediction. Using the best performing methods, we obtain 43.9% in 0.1-ShortestPath and 34.1 % in 0.01-0.01 GeomRandWalk and 52.5% error in the Graphlet. This is not bad considering that we are dealing with 4 different (balanced) classes.

We can also look at the distribution of the errors using a histogram as shown in figure 7.

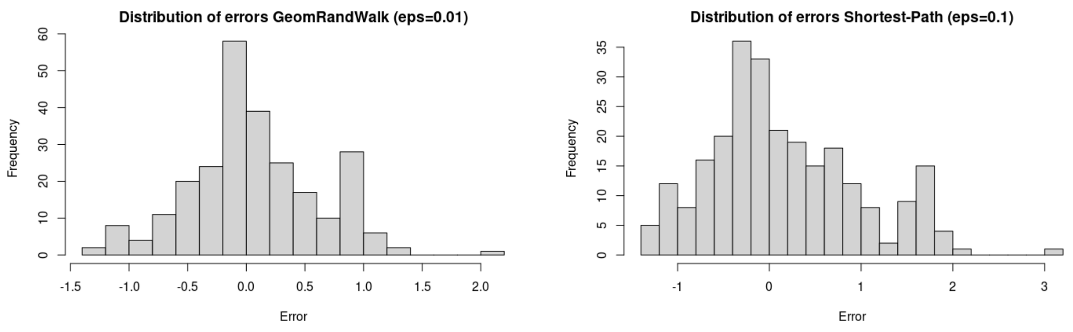

Figure 7: *Distribution of errors in the regression method.*

We can see that the errors are centered around 0 and the frequency of the errors drop as we get further from the center. We can also check that the Geometric Random Walk Kernel performs better than the Shortest-Path Kernel (heavier tails) just like we see when looking at the RMSE of the methods.

5.2 Classification

Also, we were interested in tackling this problem as a classification task because the target value is an integer with only a few possible values in our dataset. That's why we also used a formulation of a SVM that allowed for multi-class classification.

Table 2: *Classification error of the kernel-hyperparameter combinations.*

Kernel	$C = 1$	$C = 0.1$	$C = 0.01$
Shortest Path	0.737	0.729	0.552
GRW ($\lambda = 0.01$)	0.474	**0.321**	0.353
GRW ($\lambda = 0.001$)	0.509	0.478	0.651
GRW ($\lambda = 0.0001$)	0.666	0.674	0.710
Graphlet	0.53	0.765	0.694

It is obvious from the table above that the best performing method is the Geometric Random Walk with $\lambda = 0.01$. Using $C = 0.1$ as the hyperparameter for the SVM we achieve an error of 32.1% showing a slight improvement from the results we saw using Support Vector Regression.

5.3 Overall Classifying Analysis

By using the obtained models, we can look into deeper detail by comparing the the prediction error of each individual class, that is, the marginal prediction error. The results are shown in table 3, which were obtained from the best performing models in the sections above. We can see that in the two last methods, which correspond to classification, the results are particularly good in the case of class 3, that is, 3-colorability. This is result is explored in the following section. It is also worth noting that we ignore the class 7 due to the few available observations.

Table 3: *Marginal classification error for the majoritary classes. The first two methods correspond to regression and the last two, to classification.*

Kernel-Method	**3**	**4**	**5**	**6**
Shortest-Path $\epsilon = 0.1$	0.34	0.31	0.42	0.63
GRW $\lambda = 0.01$, $\epsilon = 0.01$	0.46	0.25	0.19	0.44
GRW $\lambda = 0.01$, $C = 0.1$	**0.13**	0.46	0.23	0.54
Graphlet $C = 1$	0.38	0.64	0.59	0.56

5.4 3-colorable Binary Classification

In certain applications it is interesting to know if the colourability of a graph is a certain value. Therefore we also wanted to look at a binary classifier that would check if the colourability of a graph is 3 or not. This is interesting because a 3-colourable graph is planar, meaning it can be embedded in the plane. Notice, that telling if a given graph is 3-colourable is also an NP-complete problem.

Table 4: *Binary classification error in the 3-colorability problem.*

Kernel	$C = 1$	$C = 0.1$	$C = 0.01$
Shortest Path	0.491	0.325	0.125
GRW ($\lambda = 0.01$)	**0.03**	**0.06**	0.07
GRW ($\lambda = 0.001$)	0.51	**0.06**	0.28
GRW ($\lambda = 0.0001$)	0.50	0.32	0.37
Graphlet	0.22	0.25	0.34

The GRW with $\lambda = 0.01$ and $C = 1$ has an amazing performance with just a 3%

error rate.

5.5 Feature Extraction

We now propose a method to use features learned via kernelization applying traditional Machine Learning algorithms. We use the Principal Components we extracted using kernel PCA as features. We use the kernel that showed to have the best performance in this problem, that is, the ($\lambda = 0.01$)-GRW and ($\lambda = 0.001$)-GRW. We use all the components extracted using kPCA as attributes of the models, and we include n (graph size) as an additional attribute.

Table 5: *Traditional methods on features extracted from kPCA.*

	LDA	**3-KNN**	**5-KNN**	**11-KNN**
GRW $\lambda = 0.01$	0.28	0.25	0.29	0.31
GRW $\lambda = 0.001$	0.27	0.27	0.33	0.48

Using the features extracted using kPCA we see that the CV-error for 3-kNN using the Geometric Random Walk kernel achieves a lower error rate, 25%, than both the regression and classification approaches.

5.6 Discussion

Now we will see how the best performing methods perform in the test set.

The method that solved with the highest accuracy the regression approach was the SVM with hyperparameter $\epsilon = 0.01$ and using the GRW kernel with $\lambda = 0.01$. The CV error was 34.1% and the test error is 36%.

The method that solved with the highest accuracy the multi-class classification approach was the SVM with hyperparameter $C = 0.1$ and using the GRW kernel with $\lambda = 0.01$. The CV error was 32.1% and the test error is 42%.

The method that predicted with the highest accuracy using the features extracted using kPCA was the kNN with $k = 3$ and using the Geometric Random Walk kernel with $\lambda = 0.01$. The CV error was 25% and the test error is 18%. This is the method that gives the lowest test error.

The method that solved with the highest accuracy the binary classification task (3-colourability) was the SVM with hyperparameter $C = 1$ and using the Geometric Random Walk kernel with $par = 0.01$. The CV error was 3.1% and the test error is 6%.

The kernel that performs the best for all problems is the $\lambda = 0.01$ GRW. Notice how the SVM classifier has not generalized well to the test data (the CV error was around 32%).

6 Conclusions and Future Work

In conclusion, we have found that Kernels are able to extract meaningful information about the colorability of graphs in such a way that kernelizable methods such as Support Vector Machines can be used to predict the chromatic number of these networks with a pretty good accuracy.

We have tackled this problem from several angles using three different Graph Kernels (Shortest-Path Kernel, Geometric Random Walk and Graphlet Kernel). First we kernelized a Support Vector Regressor (ϵ-SVR) to tackle this problem as a regression one. Then we used a multi-class formulation of the SVM to approach the problem as a classification one. We also used Kernel PCA as a way to extract features from the graphs and then used LDA and kNN using the Principal Components as the representation for the graphs. Finally, we wanted to see if we obtained a higher accuracy by focusing on a single chromatic number and setting up a binary classification problem, in particular we wanted to know whether or not a graph was 3-colourable or not.

In general, the kernel that gave the best results for all of these problems was the Geometric Random Walk with parameter equal to 0.01. The most successful approach to predict the chromatic number of the graphs was using kNN with $k = 3$ and using the features obtained via kernel Principal Components achieving an 18% error rate in the test set.

Further work could extend the proposed analysis using bigger generated datasets. This would require great computational power, but could also provide great insight on the problem. Our work has shown that it is an area worth exploring and that an extensive analysis of features extracted from Graph Kernels could be of big use to direct approximation of the Chromatic Number. This would also contribute in the field of approximate solutions of class-NP problems.

References

[1] Henrique Lemos et al. *Graph Colouring Meets Deep Learning: Effective Graph Neural Network Models for Combinatorial Problems.* 2019. arXiv: `1903.04598 [cs.LG]`.

[2] Karsten M. Borgwardt and Hans-Peter Kriegel. *Shortest-path kernels on graphs.* URL: `https://www.dbs.ifi.lmu.de/~borgward/papers/BorKri05.pdf`.

[3] URL: `https://ysig.github.io/GraKeL/0.1a8/kernels/random_walk.html`.

[4] Reinhard Diestel. *Graph Theory.* 5th edition. Springfield, 2016/17.

[5] Faiza Shah Furqan Aziz Afan Ullah. "Feature selection and learning for graphlet kernel". In: *Elsevier* 136 (August 2020), pp. 63–70. DOI: `https://doi.org/10.1016/j.patrec.2020.05.023`.

[6] Mahito Sugiyama. *CRAN - Package Graph Kernels.* 2018. URL: `https://cran.r-project.org/web/packages/graphkernels/index.html` (visited on 12/27/2020).

[7] Alexandros Karatzoglou. *CRAN - Package Kernlab.* 2019. URL: `https://cran.r-project.org/web/packages/kernlab/index.html` (visited on 12/25/2020).