UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Topological data analysis: applications in machine learning

Sabrina Graciela Suárez Calcina

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Sabrina Graciela Suárez Calcina

Topological data analysis: applications in machine learning

Thesis submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Doctor in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Marcio Fuzeto Gameiro

USP – São Carlos
December 2018

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

S144t S. Calcina, Sabrina
Topological data analysis: applications in
machine learning / Sabrina S. Calcina; orientador
Marcio Fuzeto Gameiro. -- São Carlos, 2018.
121 p.

Tese (Doutorado - Programa de Pós-Graduação em
Ciências de Computação e Matemática Computacional) --
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2018.

1. Persistent homology. 2. Persistence diagrams.
3. Support Vector Machine. 4. Naive Bayes. 5.
Support Vector Regression. I. Fuzeto Gameiro,
Marcio, orient. II. Título.

Sabrina Graciela Suárez Calcina

Análise topológica de dados: aplicações em aprendizado de máquina

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutora em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Marcio Fuzeto Gameiro

USP – São Carlos
Dezembro de 2018

This is for you, Mom.
Thanks for always being there for me.

ACKNOWLEDGEMENTS

My immense gratitude to God, for giving me every day the strength not to desist in my goal.

I would like to express my sincere gratitude to my advisor Prof. Marcio Gameiro for the continuous support in our research, for his motivation, time, enthusiasm, patience, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

To Institute of Mathematics and Computer Sciences, ICMC-USP.

To my family, thank you for encouraging me in all of my pursuits and inspiring me to follow my dreams. I am especially grateful to my mother Julia, who supported me financially and spiritually. I always knew that you believed in me and wanted the best for me. To my uncles: Gregorio, Isidro, Francisco, Mario and Leonidas, and my brothers: Carlos and Nayeli. I love you all so much.

I must express my very profound gratitude to my husband Álvaro for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without him. Thank my love.

I thank my fellow labmates, especially my friends: Larissa, Caroline, Miguel, Alfredo, and Adriano. In particular, I thank my friend Stevens for his great support, for the sleepless nights we were working before deadlines, and for all the moments we have had in the last four years.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

*“Mathematics is a more powerful instrument of knowledge than
any other that has been bequeathed to us by human agency.”
(Descartes)*

RESUMO

CALCINA, S. S. **Análise topológica de dados: aplicações em aprendizado de máquina.** 2018. 121 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

Recentemente a topologia computacional teve um importante desenvolvimento na análise de dados dando origem ao campo da Análise Topológica de Dados. A homologia persistente aparece como uma ferramenta fundamental baseada na topologia de dados que possam ser representados como pontos num espaço métrico. Neste trabalho, aplicamos técnicas da Análise Topológica de Dados, mais precisamente, usamos homologia persistente para calcular características topológicas mais persistentes em dados. Nesse sentido, os diagramas de persistencia são processados como vetores de características para posteriormente aplicar algoritmos de Aprendizado de Máquina. Para classificação, foram utilizados os seguintes classificadores: Análise de Discriminantes de Mínimos Quadrados Parciais, Máquina de Vetores de Suporte, e Naive Bayes. Para a regressão, usamos a Regressão de Vetores de Suporte e KNeighbors. Finalmente, daremos uma certa abordagem estatística para analisar a precisão de cada classificador e regressor.

Palavras-chave: Homologia persistente, Diagramas de persistencia, Números de Betti, Classificação de proteínas, Classificador PLS-DA, Classificador SVM, Classificador Naive Bayes, Regressor SVR, Regressor KNeighbors.

ABSTRACT

CALCINA, S. S. **Topological data analysis: applications in machine learning**. 2018. [121](#) p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

Recently computational topology had an important development in data analysis giving birth to the field of Topological Data Analysis. Persistent homology appears as a fundamental tool based on the topology of data that can be represented as points in metric space. In this work, we apply techniques of Topological Data Analysis, more precisely, we use persistent homology to calculate topological features more persistent in data. In this sense, the persistence diagrams are processed as feature vectors for applying Machine Learning algorithms. In order to classification, we used the following classifiers: Partial Least Squares-Discriminant Analysis, Support Vector Machine, and Naive Bayes. For regression, we used Support Vector Regression and KNeighbors. Finally, we will give a certain statistical approach to analyze the accuracy of each classifier and regressor.

Keywords: Persistent Homology, Persistence diagrams, Betti numbers, Protein classification, PLS-DA classifier, SVM classifier, Naive Bayes classifier, SVR regressor, KNeighbors regressor.

LIST OF FIGURES

Figure 1	– Filtration of simplicial complexes $K^1 \subset K^2 \subset \dots \subset K^6$ and their Betti number β_0 and β_1 (top); and the corresponding persistence diagrams of connected components (bottom-left) and cycles (bottom-right).	32
Figure 2	– The k -simplices, for each $0 \leq k \leq 3$	39
Figure 3	– A simplicial complex (a) and disallowed collections of simplices (b).	40
Figure 4	– Construction of the Delaunay triangulation. (Left) Voronoï diagram for a set of points. (Middle) Delaunay triangulation for a set of points is obtained by connecting all the points that share common Voronoï cells. (Right) Associated Delaunay complex is overlaid.	41
Figure 5	– A set of points sampling the letter R, with its α -hull (left) and its α -shape (right).	41
Figure 6	– Construction of the α -shape. The α -shape of a set of non-weighted points. The dark coloured sphere is an empty α -ball with its boundary connecting M_1 and M_2 (left). The light coloured spheres represent a set of weighted points. The dark coloured sphere represents an α -ball B which is orthogonal to W_1 and W_2 (right).	42
Figure 7	– Intersection of the disks (left), and Čech complex (right).	42
Figure 8	– Intersection of the disks (left), and Vietoris-Rips complex (right).	43
Figure 9	– The Vietoris-Rips complex of six equally spaced points on the unit circle.	43
Figure 10	– Union of nine disks, convex decomposition using Voronoï cells. The associated alpha complex is overlaid.	43
Figure 11	– Convex decomposition of a union of disks. The weighted alpha complex is superimposed.	44
Figure 12	– Three consecutive groups in the chain complex. The cycle and boundary subgroups are shown as kernels and images of the boundary maps.	46
Figure 13	– From left to right, the simplicial complex, the disc with a hole, the sphere and the torus.	47
Figure 14	– The class γ is born at K^i and dies entering K^{j+1}	49
Figure 15	– Six different α -shapes for six values of radius increasing from t_1 to t_6 are shown. The first α -shape is the point set itself, for $r = 0$; the last α -shape is the convex hull, for $r = t_6$	50
Figure 16	– Increasing sequence of simplicial complex of Figure 15.	50

Figure 17 – Persistence diagrams of the filtration of Figure 16 corresponding to the connected components β_0 (left) and the cycles β_1 (right).	50
Figure 18 – A total order on simplices (compatible with the filtration of Figure 16). . . .	52
Figure 19 – Confusion matrix for a disjoint two-class problem.	70
Figure 20 – Confusion matrix for a disjoint three-class problem.	71
Figure 21 – Pipeline about entire proposed method.	76
Figure 22 – Pipeline about entire proposed procedure.	79
Figure 23 – Average accuracy values according to m for SVM, PLS-DA, and Naive Bayes classifiers for the 19 proteins dataset (<i>R-form</i> and <i>T-form</i>).	81
Figure 24 – Average accuracy values according to m for (a) SVM, (b) PLS-DA, and (c) Naive Bayes classifiers for the 900 proteins dataset.	82
Figure 25 – Projections and the validation tables of the proteins classification of G_1 (black), G_2 (green), and G_3 (red) group using PLS-DA classifier. (a), (b), G_1 , G_2 , and G_3 ; (c), (d), G_1 and G_2 ; (e), (f), G_1 and G_3 ; (g), (h), G_2 and G_3 . (a), (c), (e), and (g) are projections, the remaining are the respective confusion matrix.	83
Figure 26 – Projections and the validation tables of the proteins classification of G_1 (black), G_2 (green), and G_3 (red) group using SVM classifier. (a), (b), G_1 , G_2 , and G_3 ; (c), (d), G_1 and G_2 ; (e), (f), G_1 and G_3 ; (g), (h), G_2 and G_3 . (a), (c), (e), and (g) are projections, the remaining are the respective confusion matrix.	84
Figure 27 – Projections and the validation tables of the proteins classification of G_1 (black), G_2 (green), and G_3 (red) group using Naive Bayes classifier. (a), (b), G_1 , G_2 , and G_3 ; (c), (d), G_1 and G_2 ; (e), (f), G_1 and G_3 ; (g), (h), G_2 and G_3 . (a), (c), (e), and (g) are projections, the remaining are the respective confusion matrix.	85
Figure 28 – Classifiers comparison of the F_1 -score performance in function of m , for the 900 proteins in the cases: (a) G_1 and G_2 ; (b) G_2 and G_3 ; (c) G_1 and G_3 ; (d) G_1 , G_2 , and G_3 ; and (e) the 19 proteins in the case: <i>R-form</i> and <i>T-form</i>	86
Figure 29 – Filtration of cubical complexes $X^1 \subset X^2 \subset \dots \subset X^6$, and their Betti numbers β_0 and β_1	93
Figure 30 – Persistence diagrams PD_0 (left) and PD_1 (right) of the filtration in Figure 29. Notice that the fact that the point (5,6) appears twice in PD_1 is not visible in the plot.	93
Figure 31 – Level sets of solutions $u(x,y,t)$ of the predator-prey system (8.1). The solution on the first row correspond the $\beta = 2.0$, on the second row to $\beta = 2.1$, and on the third row to $\beta = 2.2$. The solutions on the first column correspond to $t = 100$, and the second column to $t = 200$, and on the third column to $t = 300$	95

Figure 32 – Some complexes on the filtration of the level sets of the solution corresponding to $\beta = 2.0$ on Figure 31 (top) and the corresponding persistence diagrams (bottom).	95
Figure 33 – Pipeline about the procedure for feature vector extraction.	96
Figure 34 – Average accuracy values versus the parameter m for (a) SVM, (b) PLS-DA, and (c) Naive Bayes classifiers.	98
Figure 35 – Classifiers comparison of the F_1 -score performance in function of m , for (a) P_1 and P_2 ; (b) P_2 and P_3 ; (c) P_1 and P_3 ; (d) P_1 , P_2 , and P_3 groups.	99
Figure 36 – Level sets of solutions $u(x,y,t)$ of the predator-prey system (9.1). The solution on the first row correspond the $\beta = 1.75$, on the second row to $\beta = 1.8$, on the third row to $\beta = 1.85$, on the fourth row to $\beta = 1.9$, on the fifth row to $\beta = 1.95$. The solutions on the first column correspond to $t = 301$, and the second column to $t = 350$, and on the third column to $t = 400$	103
Figure 37 – Level sets of solutions $u(x,y,t)$ of the predator-prey system (9.1). The solution on the first row correspond the $\beta = 2.0$, on the second row to $\beta = 2.05$, on the third row to $\beta = 2.1$, on the fourth row to $\beta = 2.15$, on the fifth row to $\beta = 2.2$. The solutions on the first column correspond to $t = 301$, and the second column to $t = 350$, and on the third column to $t = 400$	104
Figure 38 – Some complexes on the filtration of the level sets of the solution corresponding to $\beta = 1.95$ on Figure 36 (the first column-bottom) and the corresponding persistence diagrams (bottom).	105
Figure 39 – Level sets of solutions $u(x,y,t)$ of the Ginzburg-Landau Equation (9.2). The solution on the first row correspond the $\beta = 1.0$, on the second row to $\beta = 1.2$, on the third row to $\beta = 1.4$, on the fourth row to $\beta = 1.6$, on the fifth row to $\beta = 1.8$. The solutions on the first column correspond to $t = 100$, and the second column to $t = 200$, and on the third column to $t = 300$	106
Figure 40 – Some complexes on the filtration of the level sets of the solution corresponding to $\beta = 1.0$ on Figure 39 (the third column-top) and the corresponding persistence diagrams (bottom).	107
Figure 41 – Average prediction values (triangles) with standard deviation error bar versus the actual value of the parameter β (first column), and average prediction (triangles) plus all the predicted values (red dots) versus the actual value of the parameter β (second column) for $m = 10$. The regressor used was KNeighbors (first row) and SVR (second row).	109
Figure 42 – Average R^2 values with RMSE error bars as a function of the parameter m for KNeighbors and SVR regressor.	109

Figure 43 – Average prediction values (triangles) with standard deviation error bar versus the actual value of the parameter β (first column), and average prediction (triangles) plus all the predicted values (red dots) versus the actual value of the parameter β (second column) for $m = 10$. The regressor used was KNeighbors (first row) and SVR (second row). 110

Figure 44 – Average R^2 values with RMSE error bars as a function of the parameter m for KNeighbors and SVR regressor. 111

LIST OF ALGORITHMS

Algorithm 1 – Incremental algorithm	47
Algorithm 2 – The standard algorithm for the reduction of the boundary matrix	51

LIST OF TABLES

Table 1 – Summary of several types of complexes that are used for persistent homology.	44
Table 2 – Comparison between some complexes that are used for persistent homology.	44
Table 3 – Overview of existing software for the computation of <i>Persistent Homology</i> .	59
Table 4 – Popular admissible Kernels.	65
Table 5 – List of Van Der Waals radii ($\text{\AA}$) of some chemical elements.	80
Table 6 – Protein molecules used for the Hemoglobin classification.	80
Table 7 – Comparative results for the performance of SVM classifier in the case of 900 proteins dataset	87
Table 8 – Comparative results for the performance of PLS-DA classifier in the case of 900 proteins dataset.	88
Table 9 – Comparative results for the performance of Naive Bayes classifier in the case of 900 proteins dataset.	88
Table 10 – Comparative results for the performance of classifiers in the case of 19 proteins dataset.	88
Table 11 – CV classification rates (%) of SVM with MTF-SVM (cited from Cang <i>et al.</i> (2015)) and our method.	89
Table 12 – CV classification rates (%) of SVM with MTF-SVM, PWGK-RKHS (cited from Cang <i>et al.</i> (2015), Kusano, Fukumizu and Hiraoka (2017), Kusano, Fukumizu and Hiraoka (2016)), and our method.	89
Table 13 – Comparative results for the performance of SVM, PLS-DA, and Naive Bayes classifier.	99
Table 14 – R^2 and RMSE measure using KNeighbors and SVR regressor for $m = 10$ for the predator-prey system (9.1).	108
Table 15 – R^2 and RMSE measure using KNeighbors and SVR regressor for $m = 10$ for the complex Ginzburg-Landau Equation (9.2).	111

LIST OF SYMBOLS

MTF — Molecular topological fingerprint

MTF-SVM — Molecular topological fingerprint based on support vector machine

RKHS — Reproducing kernel Hilbert spaces

PWGK — Persistence weighted Gaussian kernel

L^p — The space of measurable functions

$\mathbb{N}$ — The natural numbers

$\mathbb{Z}$ — The integers numbers

$\mathbb{Q}$ — The rational numbers

$\mathbb{R}$ — The real numbers

$\mathbb{Z}_p$ — The quotient ring of integers modulo p

$\mathbb{Z}_2 = \{\bar{0}, \bar{1}\}$ — The cycle group of integers modulo 2

$\emptyset$ — The empty set

$\mathbb{R}^n$ — The real coordinate space of n dimensions

σ^k — k -simplex

σ, τ — A simplex

K — Simplicial complex

K^j — The collection of j simplices

$\text{Nrv } X$ — The nerve of X

$\pi_u(x)$ — A weighted squared distance of a point $x \in \mathbb{R}^n$ from $u \in P$

w_u — The weighted of the point u

W — The set of the weighted points

$V(P)$ — The Voronoï diagram of a set of points P

V_p — The Voronoï cell of a point $p \in P$

$\mathcal{D}$ — The Delaunay complex

$D(P)$ — The Delaunay triangulation of a set of points P

α -shape — The alpha shape

$B_s(r)$ — The closed ball with center s and radius r

$\check{Cech}(r)$ — The Čech complex with radius r

$VR(r)$ — Vietoris-Rips complex with radius r

$\mathcal{A}$ — The weighted alpha complex

$R_u(r)$ — The convex region of a positive weighted point u and radius r

$C_n(K)$ — The n -chain group

∂_n — n -th Boundary operator

$\text{Ker } \partial_n$ — The Kernel of n -th boundary operator

$\text{Im } \partial_n$ — The image of n -th boundary operator

Z_n — The n -th cycle group

B_n — The n -th boundary group

H_n — The n -th homology group

$O(N)$ — The complexity N

$\lceil \cdot \rceil$ — The ceiling function

PH — The persistent homology

PH_n — The n -th persistent homology

$f_n^{i,j}$ — The inclusion map on the n -cycles

$\phi_n^{i,j}$ — The homology map

$H_n^{i,p}$ — The n -persistent n -th homology group

$\mathcal{K}$ — The alpha complexes filtration

PD_k — The k -th persistence diagram

$\overline{\mathbb{R}}^2$ — The extended plane

β_n — n -th Betti number

β_0 — The number of connected components

β_1 — The number of loops or tunnels

β_2 — The number of cavities

δ — The square matrix of dimension $n \times n$

$\text{low}(j)$ — The largest index value in the column j

$dg(\sigma)$ — The smallest number p such that a simplex $\sigma \in K^p$

CGAL — Computational Geometry Algorithms Library

C++ — Programming language

PHAT — Persistent Homology Algorithms Toolbox

DIPHA — Distributed Persistent Homology Algorithm

$\mathcal{W}$ — The weak witness complex

$\mathcal{W}_v$ — The parametrized witness complexes

WRCF — The weight rank clique filtration

ML — Machine learning

X — The vector space

$C = \{c_1, c_2, \dots, c_d\}$ — A set of class labels

$g: X \times \mathbb{R}^m \rightarrow C$ — The learning function

$f: X \rightarrow C$ — The trained function

$P(h)$ — The prior probability of the hypothesis h

$P(D)$ — The prior probability of the training data D

$P(D|h)$ — The probability of observing data D given hypothesis h

$P(h|D)$ — The posterior probability of h that holds after observing the training data D

MAP — Maximum a posteriori

h_{MAP} — A maximum a posteriori hypothesis

ml — A maximum likelihood

h_{ml} — A maximum likelihood hypothesis

S — A set of m classes

$P(T_{\text{new}} = c | x_{\text{new}}, X, S)$ — The prior probability that the class label T_{new} for an unseen object x_{new}

$p(x_{\text{new}} | T_{\text{new}} = c, X, S)$ — The distribution specific to class c evaluated at x_{new}

$p(x_{\text{new}} | X, S)$ — The marginal likelihood

$P(T_{\text{new}} = c | X, S)$ — The prior probability of the class c conditioned on just the training data X

SVM — Support vector machine

TAE — The statistical learning theory

$\mathcal{F}$ — The feature space

$\Phi: \mathbb{R}^n \rightarrow \mathcal{F}$ — A nonlinear map

$k: \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{K}$ — The kernel function

$\tanh$ — The hyperbolic tangent function

RBF — The Gaussian radial basis function

sign — The sign function

PLS-DA — Partial least squares-discriminant analysis

PLS-R — Partial least squares-regression

PCA — Principal component analysis

TP — The true positive

TN — The true negative

FP — The false positive

FN — The false negative

TP_k — The number of actual class samples correctly predicted in the class k

E_{ij} — The number of items with true class j that were classified as being in class i

Var — The variance

SVR — Support Vector Regression

KKT — Karush-Kuhn-Tucker

ν -SVR — The ν -support vector regression

R^2 — The coefficient of multiple determination for multiple regression

RMSE — The root mean square error

$\alpha_{\min}$ — The minimum birth value

$\alpha_{\max}$ — The maximum birth value

$\mathcal{X}$ — A filtration

$v^k(\mathcal{X})$ — The k -dimensional persistence feature vector of the filtration $\mathcal{X}$

$w(\mathcal{X})$ — The new persistence feature vector of the filtration $\mathcal{X}$

$W(\mathcal{X})$ — The general matrix of the filtration $\mathcal{X}$

TDA — Topological data analysis

γ — The k -dimensional hole

$\mathbb{R}^+$ — Space of positive real numbers

$\mathbf{C}_i$ — The PLS components

$\mathbf{pc}_i$ — The principal components

$\Omega := [a, b] \times [c, d]$ — The rectangular domain

X^r — The cubical complex filtration

(b, d) — The birth-death pairs

U_r — The sub-level sets of function u

$u(x, y, t)$ — The population densities of prey at time t and vector position (x, y)

$v(x, y, t)$ — The population densities of predators at time t and vector position (x, y)

$v^0(\mathcal{X})$ — The 0-dimensional persistence feature vectors

$v^1(\mathcal{X})$ — The 1-dimensional persistence feature vectors

Δt — The time step

CONTENTS

1	INTRODUCTION	31
1.1	Outline	33
2	RELATED WORKS	35
3	COMPUTATIONAL TOPOLOGY	39
3.1	Complexes construction	42
3.2	Homology group	45
3.3	Persistent Homology (<i>PH</i>)	48
3.3.1	<i>Birth and Death</i>	48
3.3.2	<i>Persistence diagrams</i>	49
3.4	Algorithms for computing <i>PH</i>	51
4	SOFTWARE FOR COMPUTING PERSISTENT HOMOLOGY . . .	55
4.1	CGAL	55
4.2	Software for computing <i>PH</i>	56
5	MACHINE LEARNING	61
5.1	Classification	62
5.1.1	<i>Naive Bayes</i>	62
5.1.2	<i>Support Vector Machine</i>	64
5.1.3	<i>Partial least squares-discriminant analysis</i>	66
5.2	The Nonlinear Regression	68
5.2.1	<i>Support Vector Regression</i>	69
5.3	Some statistical measures	70
6	PROPOSED METHOD	75
7	PROTEINS CLASSIFICATION	77
7.1	Proposed Method	78
7.2	Experiments and Results	79
7.2.1	<i>Classifiers evaluation</i>	81
7.2.2	<i>Visualization of classifiers for the 900 proteins</i>	82
7.2.3	<i>Comparing classifiers</i>	86
7.2.4	<i>Discussion</i>	88

7.3	Conclusions and Future Works	89
8	PARAMETER IDENTIFICATION IN A PREDATOR-PREY SYSTEM	91
8.1	Persistent Homology of Level Sets	92
8.2	Proposed Method	96
8.3	Experiments and Results	97
8.3.1	<i>Comparing classifiers</i>	98
8.4	Conclusions and Future Works	99
9	PARAMETER ESTIMATION IN SYSTEMS EXHIBITING SPATIALLY COMPLEX SOLUTIONS	101
9.1	Persistent Homology of Level Sets	102
9.1.1	<i>Predator-Prey System</i>	102
9.1.2	<i>Ginzburg-Landau</i>	105
9.2	Proposed Method	107
9.3	Experiments and Results	108
9.3.1	<i>Predator-Prey System</i>	108
9.3.2	<i>Ginzburg-Landau</i>	110
9.4	Conclusions	111
10	CONCLUSION AND FUTURE WORKS	113
	BIBLIOGRAPHY	115

INTRODUCTION

Topology is a subfield of mathematics that in the last fifteen years has applications to many different real-world problems. One of its main tasks has been developing a tool set for recognizing, quantifying, and describing the shape of datasets ([ZOMORODIAN, 2005](#)). The approach to the analysis that extracts the topological characteristics in the data is known by *Topological Data Analysis* (TDA). More precisely, TDA provides tools to study the shape of the data. Further, it gives a powerful framework for analyzing the qualitative features and dimensionality reduction of data. One of the goals of TDA is to infer multi-scale and quantitative topological structures directly from the source (dataset). TDA provides a wealth of new insights into the study of data in a diverse set of applications, for example [Carlsson \(2009\)](#), [Epstein, Carlsson and Edelsbrunner \(2011\)](#), [Edelsbrunner and Harer \(2010\)](#).

Two of the most important topological tools to study data are homology and persistence. More specifically, homology is an algebraic and formal road to talk about the connectivity of a space. This connectivity is determined by its cycles that can be of distinct dimensions and be organized by abelian groups. Moreover, cycles form homology groups, and ranks of these groups, known as Betti numbers, count the number of independent cycles in each dimension ([EDELSEBRUNNER, 2014](#)). Even better known than Betti number is the Euler characteristic. In particular, Henri Poincaré proved that the Euler characteristic is equal to the alternated sum of the Betti numbers. Another important technique for topological attributes is persistence because this new measure enables us to simplify spaces topologically ([EDELSEBRUNNER, 2001](#); [ZOMORODIAN, 2005](#)).

This has led to the study of *Persistent Homology* (PH), in which the invariants are in the form of *Persistence Diagram* (PD) ([EDELSEBRUNNER H.; ZOMORODIAN, 2002](#)). Moreover, visualization of the data using the PD allows recognizing patterns in a faster fashion than examining by algebraic methods. Consequently, the central idea in PH is to analyze how holes appear and disappear, as simplicial complexes are created. Thereby, PH appears as a method used in TDA to study qualitative features of data that persist across multiple scales ([ZOMORODIAN;](#)

CARLSSON, 2005). In general, the types of datasets that can be studied with PH include finite metric spaces, level sets of real-valued functions, digital images, and networks (OTTER *et al.*, 2017). There is a wide range of studies that address the subject to be investigated in the present work, for example Kusano, Fukumizu and Hiraoka (2016), Chazal *et al.* (2015), Cang *et al.* (2015), Xia and Wei (2014), Xia and Wei (2015), Kasson *et al.* (2007), Lee *et al.* (2011), Singh *et al.* (2008), Gameiro *et al.* (2015), Hiraoka *et al.* (2016), Nakamura *et al.* (2015), Carlsson *et al.* (2008), Silva and Ghrist (2007), Garvie (2007), Holling (1965), Wang and Wei (2016).

In this work, we studied the persistent homology of a filtered d -dimensional cell complex K . A filtered cell complex is an increasing sequence of cell complexes, each contained in the next. In this context, for giving a better illustration to persistent homology, we presented one example related to filtration of simplicial complexes. Consider the finite collection of 2-dimensional simplicial complexes $K^1 \subset K^2 \subset \dots \subset K^6$ shown in Figure 1. For each simplicial complex X^i in this filtration, the connected components β_0 and the number of cycles β_1 are shown in Figure 1 (top). In this way, persistent homology represented by persistence diagrams in Figure 1 (bottom), tells us how long each of these topological properties (connected components and holes) persist. Notice that the point $(4, 6)$ in the diagram corresponding to β_1 , for example, tells us that a cycle was created at time $t = 4$ and destroyed at time $t = 6$. The point $(1, +\infty)$ in the diagram corresponding to β_0 indicates that one of the connected components that were created at time $t = 1$ never died.

Figure 1 – Filtration of simplicial complexes $K^1 \subset K^2 \subset \dots \subset K^6$ and their Betti number β_0 and β_1 (top); and the corresponding persistence diagrams of connected components (bottom-left) and cycles (bottom-right).

Source: Elaborated by the author.

Once we have obtained the persistence diagrams, we need to interpret the results of computations. One road is mapping the space of persistence diagrams to normed metric spaces that are amenable to statistical analysis and machine learning algorithms. More specifically, within the field of data analytics, *Machine Learning* (ML) is a tool used to devise complex models and algorithms that lent themselves to prediction. One important aspect of machine learning is that it can be used for tasks of clustering, classification, regression of parameters, parameter estimation, density estimation, dimensionality reduction, and so on.

In this sense, the main goal of this work is to apply techniques from *Topological Data Analysis*, more specifically, *Persistent Homology* combined with *Machine Learning* algorithms to (1) classify proteins datasets; (2) to study the parameter identification problem in models producing complex Spatio-temporal patterns; and last, (3) to estimate parameters in models exhibiting spatially complex patterns.

1.1 Outline

To present the proposal of this work, the remaining of this thesis is structured as described below.

Chapter 2 describes the related works with our research. More specifically, related to Topological Data Analysis and use of Persistent Homology.

Chapter 3 presents theoretical aspects relevant to the studies directed to the Persistent Homology, for example, α -shapes, complexes construction, persistence diagrams, and algorithms for computing persistent homology.

Chapter 4 presents an overview of existing software for computing Persistent Homology.

Chapter 5 shows a brief theoretical description of Machine Learning, supervised classification and regression. Further, it proposes some algorithms for supervised classification and regression, and last some statistical measures.

Chapter 6 covers the proposed method and how it will be developed. This methodology is basically composed of using Topological Data Analysis to calculate topological features more persistent in the simplicial complex of an object. In addition, this topological information (persistence diagram) is in turn used as features for the Machine Learning methods used for the classification and regression.

Chapter 7 presents the use of Topological Data Analysis combined Machine Learning to classify proteins datasets. Further, experimental results are presented to evaluate and verify our proposed method.

Chapter 8 applies techniques from Topological Data Analysis, more specifically Persistent Homology, combined with Machine Learning to study the parameter identification problem in models producing complex spatio-temporal patterns.

Chapter 9 applies Persistent Homology, combined with Machine Learning Regression model to estimate parameters in systems of equations exhibiting spatially complex patterns.

Chapter 10 summarizes the conclusions drawn from the discussions presented in Chapter 7, 8, and 9. Finally, we conclude this chapter by exposing some suggestions for future works.

RELATED WORKS

The following Chapter aims to present some of the works used in the literature related to Topological Data Analysis and the use of Persistent Homology, more specifically, the use of persistence diagrams. We begin by reviewing the papers that use the Persistent Homology to analyze data. Then, the papers that use the persistent homology to classify proteins, and last the papers related to study the parameter identification problem.

[Li, Ovsjanikov and Chazal \(2014\)](#) presented a framework for object recognition using topological persistence. In this sense, persistence diagrams were used as compact and informative descriptors for shapes and images. More specifically, these diagrams were used to characterize the structural properties of the objects since they reflect spatial information in an invariant way. For this reason, the authors proposed the use of persistence diagrams built from functions defined on the objects. Specifically, their choice function was simple: each dimension of the feature vector can be viewed as a function. In addition, they conducted experiments on 3D shape retrieval, text classification, and hand gesture recognition, obtaining good results.

There is an interesting work in the field of medicine, biology, and ecology relating to time-series approaches to persistent diagrams conducted by [Pereira and Mello \(2015\)](#). The authors proposed an approach for data clustering based on topological features computed over the persistence diagram. The main contribution of their paper is a framework to cluster time-series and spatial data based on topological properties, which can correctly identify qualitative aspects of a dataset currently missed by traditional distance-based techniques. The main advantages are that their technique can detect similarities in recurrent behaviour for spatial structures in spatial datasets and time series datasets.

Some statistical approaches related to persistence diagrams were presented in the work of [Bubenik \(2015\)](#), [Robins and Turner \(2016\)](#). Their studies discussed how to transform a persistence diagram into a vector. In these methods, a transformed vector is typically expressed in a Euclidean space $\mathbb{R}^k$ or a function space L^p . Simple statistics like variances and means

are used for data analysis, and as well as Principal Component Analysis and Support Vector Machines.

For the first time, [Xia and Wei \(2014\)](#) introduced Persistent Homology to extract Molecular Topological Fingerprints (*MTFs*) based on the persistence of molecular topological invariants. *MTFs* were utilized for classification, protein characterization, and identification. More specifically, *MTFs* were employed to characterize protein topological evolution during protein folding and quantitatively predict the protein folding stability. So an excellent consistency between their molecular dynamics simulation and persistent homology prediction was found. In summary, this work revealed the topology-function relationship of proteins.

A little later, [Cang et al. \(2015\)](#) examined the uses of persistent homology as an independent tool for protein classification. For this, they introduced a *Molecular Topological Fingerprint (MTF)* model, based on a Support Vector Machine classifier (*MTF-SVM*). This *MTF* is given by the 13-dimensional vector whose elements consist of the persistence of some specific generators (the length of the second longest Betti 0 bar, the length of the third longest Betti 0 bar, etc) in persistence diagrams. The authors used two databases, specifically, all alpha, all beta, and mixed alpha and beta protein domains with nine hundred proteins, and the discrimination of hemoglobin molecules in relaxed and taut forms with 17 proteins.

[Xia, Li and Mu \(2016\)](#) introduced multiscale persistent functions for biomolecular structure characterization. Their essential idea was to combine the multiscale rigidity functions with persistent homology analysis, so as to construct a series of multiscale persistent functions, in particular multiscale persistent entropies, for structure characterization. Moreover, their method was successfully used in protein classification. For a test database used in [Cang et al. \(2015\)](#) with around nine hundred proteins, a clear separation between all alpha and all beta proteins was achieved, using only the dihedral and pseudo-bond angle information.

A recent study conducted by [Kusano, Fukumizu and Hiraoka \(2016\)](#), [Kusano, Fukumizu and Hiraoka \(2017\)](#) proposed a kernel method on persistence diagrams to develop a statistical framework in *Topological Data Analysis*. Specifically, to vectorize the persistence diagrams they employed the framework of kernel embedding of measures into *reproducing kernel Hilbert spaces (RKHS)*. Besides, [Kusano, Fukumizu and Hiraoka \(2016\)](#) proposed a useful class of positive definite kernels for embedding persistence diagrams in *RKHS* called *persistence weighted Gaussian kernel (PWGK)*. A theoretical contribution of *PWGK* allows one to control the effect of persistence and to discount the noisy topological properties in data analysis. In addition, [Kusano, Fukumizu and Hiraoka \(2017\)](#) presented one of the main theoretical results, the stability of the *PWGK*. Moreover, the method can also be applied to several problems including practical data in physics. To validate the performance of *PWGK*, they used synthesized and protein datasets of [Cang et al. \(2015\)](#).

[Gameiro, Mischaikow and Kalies \(2004\)](#) proposed the use of computational homology to measure the spatial-temporal complexity of patterns for systems that exhibit complicated spatial patterns and suggested a tentative step towards the classification and identification of patterns within a particular system. In this way, the authors showed that this technique can be used as a means of differentiating between patterns at different parameter values. Although it is computationally expensive to measure spatial-temporal chaos, the computations necessary to do such discrimination are relatively cheap. Last, one important feature of the proposed method by authors is that it is fairly automated and it can be applied to experimental data.

A little later, [Gameiro, Mischaikow and Wanner \(2005\)](#) presented the use of computational homology as an effective tool for quantifying and distinguishing complicated microstructures. Rather than discussing experimental data, the authors considered numerical simulations of the deterministic *Cahn–Hilliard* model, as well as its stochastic extension due to Cook. The method was illustrated for the microstructures generated during spinodal decomposition. These structures are fine-grained and snake-like. The microstructures are computed using two different evolution equations which have been proposed as models for spinodal decomposition.

The work of [Garvie \(2007\)](#) used two finite-differences algorithms for studying the dynamics of spatially extended predator-prey interactions with the Holling type II functional response, and logistic growth of the prey. The algorithms presented are stable and convergent provided the time step is below a (non-restrictive) critical value. Further, there are implementational advantages due to the structure of the resulting linear systems, iterative solvers, and standard direct are guaranteed to converge. The ecological implication of these results is that in the absence of external influences, certain initial conditions can lead to spatial and temporal variations in the densities of predators and prey that persist indefinitely. Finally, the results of this work are an important step toward providing the theoretical biology community with simple numerical methods to investigate the key dynamics of realistic predator-prey models.

COMPUTATIONAL TOPOLOGY

This chapter aims to present briefly some concepts necessary for this work given by Edelsbrunner (2001), Zomorodian (2005), Kaczynski, Mischaikow and Mrozek (2006), Edelsbrunner (2014). We begin by reviewing the definition of α -shapes, alpha complexes, homology group, persistent homology, and persistence diagrams. Additionally, some algorithms proposed for computing persistent homology are presented.

Let $P = \{p_0, p_1, \dots, p_k\}$ ($k \in \mathbb{N} \cup \{0\}$) be a finite set of points in $\mathbb{R}^n$. A point x is a *linear combination* of P if $x = \sum_{i=0}^k \lambda_i p_i$, for suitable real numbers λ_i . An *affine combination* is a linear combination with $\sum_{i=0}^k \lambda_i = 1$. A *convex combination* is an affine combination with $\lambda_i \geq 0$, for all i . The set of all convex combinations is the *convex hull*.

Let $S = \{v_0, v_1, \dots, v_k\}$ ($k \in \mathbb{N} \cup \{0\}$) be a finite set of vectors in $\mathbb{R}^n$. The set S is *linearly independent* if the equation $\alpha_0 v_0 + \alpha_1 v_1 + \dots + \alpha_k v_k = \vec{0}$, can only be satisfied by $\alpha_i = 0$ for $i = 0, \dots, k$. The set P of $k + 1$ points is *affinely independent* if the k vectors $p_i - p_0$, $1 \leq i \leq k$, are linearly independent.

A k -simplex σ^k ($k \in \mathbb{N} \cup \{0\}$) is the convex hull of $k + 1$ affinely independent points $P \subseteq \mathbb{R}^n$. The dimension of k -simplex σ^k is given by $\dim \sigma^k = k$. The points in P are the *vertices* of the k -simplex. Geometrically, a 0-simplex is a vertex, a 1-simplex is an edge, a 2-simplex is a triangle, and a 3-simplex is a tetrahedron (See Figure 2).

Figure 2 – The k -simplices, for each $0 \leq k \leq 3$.

Source: Adapted from Zomorodian (2005).

Let σ be a k -simplex ($k \in \mathbb{N} \cup \{0\}$). A *face* of σ is the convex hull of a non-empty subset of the vertices of σ . A *simplicial complex* K is a finite collection of simplices, such that (i) for every simplex $\sigma \in K$, every face of σ is in K ; (ii) for every two simplices $\sigma, \tau \in K$, the intersection, $\sigma \cap \tau$, is either empty or a face of both simplices (See Figure 3). The *dimension* of K is the largest dimension of any simplex in K . A *subcomplex* is a subset of the simplices that is itself a simplicial complex.

Figure 3 – A simplicial complex (a) and disallowed collections of simplices (b).

(a) The middle triangle shares an edge with the triangle on the left-and a vertex with the triangle on the right.

(b) In the middle, the triangle is missing an edge. The simplices on the left and right intersect, but not along shared simplices.

Source: Adapted from Zomorodian (2005).

Now we are ready to introduce the construction of some simplicial complexes from an arbitrary collection of sets.

Let X be a finite collection of sets. The *nerve* of X consists of all non-empty subcollections of X whose sets have a non-empty common intersection, that is, $\text{Nrv } X = \{V \subseteq X \mid \bigcap_{v \in V} v \neq \emptyset\}$.

Let P be a finite set of points in $\mathbb{R}^n$. For each $u \in P$, its *weight* is given by $w_u \in \mathbb{R}$. The *weighted squared distance* of a point $x \in \mathbb{R}^n$ from $u \in P$ is defined as $\pi_u(x) = \|x - u\|^2 - w_u$. For positive weight, we imagine a sphere with center u and radius $w_u^{1/2}$ such that $\pi_u(x) < 0$ inside the sphere, $\pi_u(x) = 0$ on the sphere, and $\pi_u(x) > 0$ outside the sphere.

The *Voronoi cell* of a point $u \in P$ is the set of points for which u is the closest, that is, $V_u = \{x \in \mathbb{R}^n \mid \|x - u\| \leq \|x - v\|, \forall v \in P\}$. Further, any two Voronoi cells meet at most in a common piece of their boundary, and together the Voronoi cells cover the entire space. In this way, given a finite set of weighted points of $u \in P$, the *weighted Voronoi cell* of $u \in P$ is the set of points $x \in \mathbb{R}^n$ with $\pi_u(x) \leq \pi_v(x)$, for all weighted points of $v \in P$. The *Voronoi diagram* of P is the collection of Voronoi cells of its points (See Figure 4). Last, the *weighted Voronoi diagram* is the set of weighted Voronoi cells of the weighted points.

Let P be a finite set of points in $\mathbb{R}^n$. We get the *Delaunay triangulation* $D(P)$ of P by connecting two points of P by a straight edge whenever the corresponding two Voronoi cells share an edge. Also, the Delaunay triangulation of P is a simplicial complex that decomposes the convex hull of the points in P . Generically, the intersection of any four or more Voronoi cells is empty. If three Voronoi cells intersect at a common point, they form a triangle. The *Delaunay complex* of a finite set of points $P \subseteq \mathbb{R}^n$ is isomorphic to the nerve of the Voronoi diagram, that is, $\mathcal{D} = \{\sigma \subseteq P \mid \bigcap_{u \in \sigma} V_u \neq \emptyset\}$. In Figure 4, the construction of the Delaunay triangulation is

presented.

Figure 4 – Construction of the Delaunay triangulation. (Left) Voronoï diagram for a set of points. (Middle) Delaunay triangulation for a set of points is obtained by connecting all the points that share common Voronoï cells. (Right) Associated Delaunay complex is overlaid.

Source: Adapted from Zhou and Yan (2012).

Let P be a finite set of points in $\mathbb{R}^n$ and $\alpha \geq 0$ a real number. An α -ball is an open ball with radius α , for $0 \leq \alpha \leq \infty$. An α -ball B is *empty* if $P \cap B = \emptyset$. The α -hull of P is the set of points that don't lie in any α -balls (See Figure 5). The boundary of the α -hull consists of circular arcs of constant curvature $1/\alpha$. So, if the circular arc is substituted by a straight line, we obtain the α -shape of P (See Figure 5). In this way, the α -shape is a polyhedron in the general sense because it doesn't have to be convex and it can have different intrinsic dimension at different places (EDELSEBRUNNER, 2014). Moreover, the α -shape can be obtained as a subset of the Delaunay triangulation which is controlled by the value of α , for $0 \leq \alpha \leq \infty$. The definition of weighted α -shape is similar but now considering a set of the weighted points $W = \{W_1, W_2, \dots, W_n\} \subset \mathbb{R}^n$. For this, we first defined orthogonal points, this is, the points P_1 and P_2 with radius $r_1, r_2 \geq 0$ are said to be *orthogonal* if $\|P_1 - P_2\|^2 = r_1^2 + r_2^2$. Similarly, P_1 and P_2 are defined as *suborthogonal* if $\|P_1 - P_2\|^2 > r_1^2 + r_2^2$. In this sense, for a given value α , the *weighted α -shape* contains all k -simplex σ such that there is an α -ball B orthogonal to the points in σ , and suborthogonal to the other points in W (ZHOU; YAN, 2012). In Figure 6, the construction of the (weighted) α -shape is presented.

Figure 5 – A set of points sampling the letter R, with its α -hull (left) and its α -shape (right).

Source: Adapted from Edelsbrunner (2014).

In the next section, we present the construction of several simplicial complexes and introduce the definition of alpha complex filtration.

Figure 6 – Construction of the α -shape. The α -shape of a set of non-weighted points. The dark coloured sphere is an empty α -ball with its boundary connecting M_1 and M_2 (left). The light coloured spheres represent a set of weighted points. The dark coloured sphere represents an α -ball B which is orthogonal to W_1 and W_2 (right).

Source: Adapted from Zhou and Yan (2012).

3.1 Complexes construction

Let P be finite set points in $\mathbb{R}^2$ and $B_s(r)$ the closed ball with center s and radius $r \geq 0$. The Čech complex is isomorphic to the nerve of the disk. This complex is denoted by

$$\check{Cech}(r) = \{\emptyset \neq T \subseteq P \mid \bigcap_{s \in T} B_s(r) \neq \emptyset\}.$$

The nerve of a cover $\{B_s(r) | s \in P\}$ constructed from the union of disks $\bigcup_{s \in P} B_s(r)$ is a Čech complex. To construct the Čech complex, we need to test whether a collection of disks has a non-empty intersection or not (See Figure 7), which can be difficult in some metric spaces.

Figure 7 – Intersection of the disks (left), and Čech complex (right).

Source: Elaborated by the author.

Similarly, we define a complex that needs only the distances between the points in P for its construction. Let $r \geq 0$ be a real number, the Vietoris-Rips complex of P is denoted as

$$VR(r) = \{\sigma \subseteq P \mid \|x - y\| \leq 2r, \forall x, y \in \sigma\},$$

and it consists of all abstract simplices in 2^P whose vertices are at most a distance $2r$. More specifically, we connect any two vertices at distance at most $2r$ from each by an edge, and add a triangle or higher-dimensional simplex to the complex if all its edges are in the complex (See Figures 8, and 9).

Figure 8 – Intersection of the disks (left), and Vietoris-Rips complex (right).

Source: Elaborated by the author.

Figure 9 – The Vietoris-Rips complex of six equally spaced points on the unit circle.

Source: Adapted from Edelsbrunner (2014).

The *alpha complex* of P is the Delaunay triangulation of P restricted to the α -balls. A simplex belongs to the alpha complex if the Voronoï cells of its vertices have a common non-empty intersection with the set of α -balls. Note that for $\alpha = 0$, the alpha complex consists just of the set P , and for α sufficiently large, the alpha complex is the Delaunay triangulation $D(P)$ of P (See Figure 10). Now, to formalize the definition of weighted alpha complex, consider W a finite set of positive weighted points of u and denoted the convex regions as $R_u(r) = B_u(r) \cap V_u$, where $B_u(r)$ is the closed ball with center u and radius $r \geq 0$, and the weighted Voronoï cells V_u . In this sense, the *weighted alpha complex* of W is isomorphic to the nerve of the convex regions $R_u(r)$ (See Figure 11), that is,

$$\mathcal{A}(r) = \left\{ \sigma \subseteq W \mid \bigcap_{u \in \sigma} R_u(r) \neq \emptyset \right\}.$$

Figure 10 – Union of nine disks, convex decomposition using Voronoï cells. The associated alpha complex is overlaid.

Source: Adapted from Zomorodian (2005).

Table 1 presents a summary of the simplicial complexes mentioned in this section. Here, we indicate the theoretical guarantees and the worst-case sizes of the complexes as functions of the cardinality N of the vertex set, where $O(\cdot)$ is the complexity of complex K , d is the dimension of the space, and $\lceil \cdot \rceil$ is the ceiling function.

Figure 11 – Convex decomposition of a union of disks. The weighted alpha complex is superimposed.

Source: Adapted from Edelsbrunner and Harer (2010).

Table 1 – Summary of several types of complexes that are used for persistent homology.

Complex K	Size of K
Čech	$2^{O(N)}$
Vietoris-Rips (VR)	$2^{O(N)}$
Alpha ($\mathcal{A}$)	$N^{O(\lceil d/2 \rceil)}$ (N points in $\mathbb{R}^d$)

Source: Adapted from Otter *et al.* (2017).

Finally, Table 2 shows a comparison between simplicial complexes mentioned in this section. In this way, we can see that the alpha complexes have better properties than other complexes. For this reason the alpha complexes will be used in this work.

Table 2 – Comparison between some complexes that are used for persistent homology.

Čech complex	Vietoris-Rips complex	Alpha complex
<ul style="list-style-type: none"> • Difficult to build • Calculations to check for intersections between balls are not easy • Higher cost • It has the property of being homotopic to the collection of balls 	<ul style="list-style-type: none"> • Easy to build • Calculations to check for intersections between balls are easy • Lower cost • It doesn't have the property of being homotopic to the collection of balls 	<ul style="list-style-type: none"> • Easy to build for $\mathbb{R}^2$ and $\mathbb{R}^3$ • Calculations to check for intersections between balls are easy • Lower cost • It has the property of being homotopic to the collection of balls

We now are ready to introduce the definition of complexes filtration. A *filtration* is an increasing sequence of topological spaces, each contained in the next. Next, let P be a set of linearly (affinely) independent points and K its Delaunay triangulation. For each simplex $\sigma \in K$, there is a real number α_σ such that σ belongs to the alpha complex $\mathcal{A}(\alpha)$ of P iff $\alpha_\sigma \leq \alpha$ ($\alpha \in \mathbb{R}$). More specifically, we can construct the alpha complex simply by collecting all vertices, edges, and triangles that have a value not larger than α . In this way, we index the n simplices such that every simplex is preceded by its faces, this is, $\alpha_{\sigma_1} \leq \alpha_{\sigma_2} \leq \dots \leq \alpha_{\sigma_n}$. This sequence of simplices is called a *filter*. To achieve this for the Delaunay triangulation, we only need to make sure that ties in the ordering are broken such that lower-dimensional simplices precede higher-dimensional simplices. Assuming a filter, we let K^j ($j \in \mathbb{N}$) be the collection of the

first j simplices, noting that it is a simplicial complex for every j . The increasing sequence of complexes,

$$\emptyset = K^1 \subset K^2 \subset \dots \subset K^n = K, \quad (3.1)$$

is called a *flat filtration* because any two contiguous complexes differ by only one simplex. Every alpha complex belongs to the flat filtration, but not every complex in (3.1) is an alpha complex. More specifically, the *alpha complex filtration* is a subsequence of (3.1) and it is generally not flat (EDELBRUNNER, 2014).

In the following section, we define homology group for simplicial complex and present an algorithm for computing the dimension of homology groups.

3.2 Homology group

Homology groups provide a mathematical language for the holes in a topological space. Perhaps surprisingly, they capture holes indirectly, by focusing on what surrounds them. Their main ingredients are group operations and maps that relate topologically meaningful subsets of a space with each other (EDELBRUNNER; HARER, 2010). In contrast to most other topological formalisms that capture connectivity, homology groups have associated fast algorithms.

Let K be a simplicial complex. A n -chain is a formal sum of n -simplices in K . The standard notation for this is $c = \sum_i a_i \sigma_i$, where σ_i is an oriented n -simplex from K and each a_i is a coefficient. For simplicity, we choose $a_i \in \mathbb{Z}_2 = \{\bar{0}, \bar{1}\}$. More specifically, a n -chain is a subset of the n -simplices in K . All these n -chains on K form an Abelian group which is called n -chain group and it is denoted as $C_n(K)$. To relate these groups, we define the *boundary* of a n -simplex as the sum of its $(n-1)$ -dimensional faces. Writing $\sigma = [u_0, u_1, \dots, u_n]$ for the simplex spanned by the listed vertices, its n -th boundary operator ∂_n over a n -simplex σ is defined by

$$\partial_n(\sigma) = \sum_{i=0}^n (-1)^i [u_0, u_1, \dots, \widehat{u_i}, \dots, u_n],$$

where $\widehat{u_i}$ indicates that u_i is deleted from the sequence. The n -th boundary operator induces a *boundary homomorphism* $\partial_n : C_n(K) \rightarrow C_{n-1}(K)$. However, a very important property of the boundary operator is that the composition operator $\partial_{n-1} \circ \partial_n$ is a zero map, for all n , this is,

$$\begin{aligned} \partial_{n-1} \partial_n(\sigma) &= \partial_{n-1} \sum_i (-1)^i [u_0, u_1, \dots, \widehat{u_i}, \dots, u_n] \\ &= \sum_{j < i} (-1)^i (-1)^j [u_0, \dots, \widehat{u_j}, \dots, \widehat{u_i}, \dots, u_n] + \\ &\quad \sum_{j > i} (-1)^i (-1)^{j-1} [u_0, \dots, \widehat{u_i}, \dots, \widehat{u_j}, \dots, u_n] \\ &= 0, \end{aligned}$$

as switching i and j in the second sum negates the first sum.

The *chain complex* is the sequence of chain groups connected by boundary homomorphisms,

$$0 \longrightarrow C_n(K) \xrightarrow{\partial_n} C_{n-1}(K) \xrightarrow{\partial_{n-1}} \cdots \xrightarrow{\partial_2} C_1(K) \xrightarrow{\partial_1} C_0(K) \xrightarrow{\partial_0} 0.$$

Note that the sequence is augmented on the right by a 0, with $\partial_0 = 0$. On the left, $C_{n+1} = 0$ because there aren't $(n+1)$ -simplices in K .

The *kernel* of ∂_n ($n \in \mathbb{N} \cup \{0\}$) is the collection of n -chains with zero boundary,

$$\text{Ker } \partial_n = \{\sigma \in C_n \mid \partial_n(\sigma) = 0\},$$

namely, the kernel of a map is everything in the domain that maps to 0 (See Figure 12). The *image* of ∂_n ($n \in \mathbb{N} \cup \{0\}$) is the collection of $(n-1)$ -chains that are borders from n -chains,

$$\text{Im } \partial_n = \{\sigma' \in C_{n-1} \mid \exists \sigma \in C_n : \sigma' = \partial_n(\sigma)\},$$

namely, the image of a map consists of all the elements in the range reached by elements in the domain (See Figure 12).

Notice that the equation $\partial_n \circ \partial_{n+1} = 0$ ($n \in \mathbb{N} \cup \{0\}$) is equivalent to $\text{Im } \partial_{n+1} \subseteq \text{Ker } \partial_n$. The $\text{Ker } \partial_n$ is called n -th *cycle group*, and it's denoted as $Z_n = \text{Ker } \partial_n$. Since $C_{-1} = 0$, every 0-chain is a cycle (i.e. $Z_0 = C_0$). The $\text{Im } \partial_{n+1}$ is called n -th *boundary group*, and it's denoted as $B_n = \text{Im } \partial_{n+1}$. A n -th *homology group* H_n is defined as the quotient group of Z_n and B_n (See Figure 12), that is,

$$H_n = Z_n / B_n = \text{Ker } \partial_n / \text{Im } \partial_{n+1}.$$

Figure 12 – Three consecutive groups in the chain complex. The cycle and boundary subgroups are shown as kernels and images of the boundary maps.

Source: Adapted from Edelsbrunner (2014).

The n -th *Betti number* ($n \in \mathbb{N} \cup \{0\}$) of the simplicial complex K is denoted as

$$\beta_n = \text{rank}(H_n) = \text{rank}(Z_n) - \text{rank}(B_n).$$

A n -th Betti number β_n is a finite non-negative integer, since $\text{rank}(B_n) \leq \text{rank}(Z_n) < \infty$.

In this way, given an alpha complex K we associate a collection of groups $H_n(K)$ with $n \in \mathbb{N} \cup \{0\}$ called *homology groups* of K , which provide the essential topological features of K . For the type of complexes that we consider in this work, the homology groups are of the form

$H_n(K) = \mathbb{K}^{\beta_n}$, where β_n is the n -th Betti number of K and $\mathbb{K}$ is the field of coefficient used to compute homology. More precisely, the homology groups are in fact vector spaces, and the Betti numbers are the dimensions of these vector spaces. In this way, the Betti numbers computed from a homology group are used to describe the corresponding space. Furthermore, the Betti numbers have the very important property that the n -th Betti number β_n is equal to the number of “ n -dimensional holes” in K . More specifically, for $n = 0, 1, 2$, β_0 is the number of *connected components* of K , β_1 is the number of *holes* or *tunnels* in K , and β_2 is the number of *cavities* in K . In Figure 13, some examples of complexes with their respective Betti numbers are presented.

Figure 13 – From left to right, the simplicial complex, the disc with a hole, the sphere and the torus.

Source: Elaborated by the author.

Now, the incremental algorithm for computing the Betti numbers of the last complex in the filtration is illustrated.

The Incremental Algorithm

Since consecutive complexes in (3.1) are very similar, it's not surprising that it's easy to compute the Betti numbers of K^{i+1} if we know the Betti numbers of K^i . In fact, there is only one additional simplex σ^{i+1} , and we need to determine how the addition of that simplex affects the connectivity of K^i . So, starting with the empty complex, the Betti numbers are computed by adding one simplex at a time. Since the complexes are geometrically realized in $\mathbb{R}^3$, only the first three Betti numbers are possibly non-zero (EDELSBRUNNER, 2014). Finally, Algorithm 1 returns the Betti numbers of the last complex in the filtration (3.1).

Algorithm 1 – Incremental algorithm

```

1:  $\beta_0 = \beta_1 = \beta_2 = 0$ 
2: for  $i = 0$  to  $n - 1$  do
3:    $p = \dim \sigma^{i+1}$ 
4:   if  $\sigma^{i+1} \in z \in Z_p(K^{i+1})$  then  $\beta_p = \beta_p + 1$ 
5:     else  $\beta_{p-1} = \beta_{p-1} - 1$ 
6:   end if
7: end for
8: Return  $\beta_0, \beta_1, \beta_2$ 

```

In the next section, a brief description of persistent homology is presented. For a more in-depth discussion please see [Edelsbrunner \(2014\)](#), [Kaczynski, Mischaikow and Mrozek \(2006\)](#).

3.3 Persistent Homology (PH)

Persistent Homology (PH) is a tool that provides metric information about the topological properties of an object and how robust these properties are with respect to a parameters change. More specifically, *PH* counts the number of connected components and holes of various dimensions and keeps track of how they change with parameters. Suppose that we have a space (object) X that varies as a function of a parameter. *PH* provides a way of capturing how the shape of this object changes as we vary that parameter.

3.3.1 Birth and Death

Now we address the question of birth and death of subcomplexes. The starting point is a sequence of homomorphisms connecting the homology groups of the complexes in the filtration. For $0 \leq i \leq j$, K^i is a subcomplex of K^j , which can be written as an injective map $f^{i,j} : K^i \hookrightarrow K^j$, and it is called the *inclusion map*. It carries over to an inclusion map on the n -cycles, $f_n^{i,j} : Z_n(K^i) \hookrightarrow Z_n(K^j)$ with $n \in \mathbb{N} \cup \{0\}$. This induces the following map on homology:

$$\phi_n^{i,j} : H_n(K^i) \rightarrow H_n(K^j),$$

which generally is not an inclusion map. More precisely, if γ is a class in $H_n(K^i)$, and $z \in Z_n(K^i)$ is a representative cycle, we let $\phi_n^{i,j}(\gamma)$ be the class in $H_n(K^j)$ that contains $f_n^{i,j}(z)$. It should be clear that the definition of $\phi_n^{i,j}$ does not depend on the choice of the representative. It takes a n -cycle in K^i and pushes it forward to K^j . For instance, if γ surrounds a hole in K^i that fills up at the time we reach K^j , then $\phi_n^{i,j}$ maps γ to $0 \in H_n(K^j)$. The image of $\phi_n^{i,j}$ is called a *persistent homology group* and it contains all n -dimensional homology classes that have representatives already present in K^i . The *persistent Betti number* is the rank of the persistent homology group and it counts the n -dimensional holes that exist all the way from K^i to K^j . For a particular class, we are interested in the smallest index i and the largest index j such that the class is non-trivial within the entire interval from K^i to K^j .

We say a class $\gamma \in H_n(K^i)$, with $n, i, j \in \mathbb{N} \cup \{0\}$ and $0 \leq i < j$, is *born* at K^i if γ is not in the image of $\phi_n^{i-1,i}$, and a class born at K^i *dies* entering K^{j+1} if $\phi_n^{i,j}(\gamma)$ is not in the image of $\phi_n^{i-1,j}$ but $\phi_n^{i,j+1}(\gamma)$ is in the image of $\phi_n^{i-1,j+1}$ (See Figure 14). The *index persistence* of γ is $(j - i + 1)$.

Let $K^\cdot$ be a filtration, the p -persistent n -th homology group of K^i is defined as

$$H_n^{i,p} = Z_n^i / (B_n^{i+p} \cap Z_n^i),$$

where $Z_n^i = Z_n(K^i)$ and $B_n^i = B_n(K^i)$. The p -persistent n -th Betti number is $\beta_n^{i,p} = \text{rank}(H_n^{i,p})$. A well chosen p promises reasonable elimination of topological noise.

Figure 14 – The class γ is born at K^i and dies entering K^{j+1} .

Source: Adapted from Edelsbrunner (2014).

3.3.2 Persistence diagrams

Given a finite collection of n -dimensional alpha complexes $K^1 \subset K^2 \subset \dots \subset K^n$, *persistent homology* provides information about the changes in the Betti numbers as we move from one alpha complex K^i to the next one K^{i+1} , with $i \in \mathbb{N}$. The collection of alpha complexes K^i with $i \in \mathbb{N}$ is called an *alpha complexes filtration* and it is denoted by $\mathcal{K}$. More precisely, the k -th *persistent homology* $PH_k(\mathcal{K})$ of $\mathcal{K}$ is characterized by its k -th *persistence diagrams* $PD_k(\mathcal{K})$,

$$PD_k(\mathcal{K}) = \{(b_i, d_i) \in \overline{\mathbb{R}}^2 \mid 0 \leq i \leq n+1\}$$

with $k \in \mathbb{N} \cup \{0\}$, $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$, where each $PD_k(\mathcal{K})$ is a multi-set of pairs of points of the form (b, d) in the extended plane $\overline{\mathbb{R}}^2$, called *birth-death pairs*. Each point $(b, d) \in PD_k(\mathcal{K})$ represents a k -dimensional hole γ in $\mathcal{K}$. The number $b \in \{1, 2, \dots, n\}$ is called the *birth time* (*birth index*) of γ and the number $d \in \{1, 2, \dots, +\infty\}$ is called the *death time* (*death index*) of γ . We say that γ was *born* at time b and *died* at time d . The birth time b indicates where the hole γ first appears in the filtration, and the death time d indicates where γ disappears in the filtration. Notice that to account for the cases where γ never dies.

Example 1 (Persistence diagrams). Consider the increasing sequence of α -shapes called α -shapes filtration, as illustrated in Figure 15. Each time the radius $r_i \geq 0$ ($i \in \mathbb{N}$) is increased, we get that at least two balls with center p_i and p_j are intercepted. In this way, it gives rise to the birth of alpha complexes of each time t_i . Now, consider the increasing sequence of simplicial complexes called filtration, $K^1 \subset K^2 \subset \dots \subset K^6$, as we can see in Figure 16. For each K^i complex in this filtration, the number of connected components β_0 and the number of loops β_1 are shown in Figure 16. *PH* is represented by the persistence diagrams of Figure 17, and it tells us how long each of these topological properties persist. These features are captured by the persistence diagrams $PD_0 = \{(1, 2), (1, 3), (1, 4), (1, +\infty)\}$ and $PD_1 = \{(5, 6)\}$ shown in Figure 17. For example, the point $(5, 6)$ in the persistence diagram corresponding to β_1 , tells us that a loop was created at time $t = 5$ and destroyed at time $t = 6$. In the persistence diagram corresponding to β_0 , the point $(1, +\infty)$ indicates that one of the connected components that were created at time $t = 1$ and it has never disappeared.

Figure 15 – Six different α -shapes for six values of radius increasing from t_1 to t_6 are shown. The first α -shape is the point set itself, for $r = 0$; the last α -shape is the convex hull, for $r = t_6$.

Source: Elaborated by the author.

Figure 16 – Increasing sequence of simplicial complex of Figure 15.

Source: Elaborated by the author.

Figure 17 – Persistence diagrams of the filtration of Figure 16 corresponding to the connected components β_0 (left) and the cycles β_1 (right).

Source: Elaborated by the author.

In the following section we review reduction techniques, which are heuristics that reduce the size of complexes without changing the persistent homology.

3.4 Algorithms for computing PH

To compute the *Persistent Homology* of a filtered simplicial complex K and obtain a persistence diagram, we need to associate K to a matrix called *boundary matrix* B , which stores information about the faces of every simplex. In this way, we lay a total ordering on the simplices of the complex that is compatible with the filtration such as:

- a face of a simplex precedes the simplex;
- a simplex in the i -th complex K^i precedes simplices in K^j for $j > i$, which are not in K^i .

Let n be the total number of simplices in the complex, and let $\sigma_1, \dots, \sigma_n$ be the simplices with respect to this ordering. A square matrix δ of dimension $n \times n$ is constructed by storing a 1 in $\delta(i, j)$ if the simplex σ_i is a face of simplex σ_j of codimension 1; otherwise, a 0 in $\delta(i, j)$ is stored.

Once one has constructed the boundary matrix, one has to reduce it using Gaussian elimination. In the following, several algorithms for reducing the boundary matrix are presented.

1. **Standard algorithm:** It is a sequential algorithm for the computation of PH. It was introduced for $\mathbb{Z}_2$ in [Edelsbrunner H. and Zomorodian \(2002\)](#) and for general fields in [Zomorodian and Carlsson \(2005\)](#). For every $j \in \{1, 2, \dots, n\}$, we define $\text{low}(j)$ to be the largest index value i ($i \in \{1, 2, \dots, n\}$) such that $\delta(i, j)$ is different from 0. If column j only contains entries equal to 0, then the value of $\text{low}(j)$ is undefined. The boundary matrix is reduced if the map low is injective on its domain of definition. In [Algorithm 2](#), the Standard algorithm for reducing the boundary matrix is illustrated. This algorithm, sometimes called the *column algorithm*, operates on columns of the matrix from left to right.

Algorithm 2 – The standard algorithm for the reduction of the boundary matrix

```

1: for  $j = 1$  to  $n$  do
2:   while there exist  $i < j$  with  $\text{low}(i) = \text{low}(j)$  do
3:     add column  $i$  to column  $j$ 
4:   end while
5: end for

```

Once the boundary matrix B is reduced, the intervals of persistence diagram can read off by pairing the simplices such as:

- If $\text{low}(j) = i$ then the simplex σ_j is paired with σ_i , and the entrance of σ_i in the filtration causes the birth of a feature that dies with the entrance of σ_j .
- If $\text{low}(j)$ is undefined then the entrance of the simplex σ_j in the filtration causes the birth of a feature. If there exists k such that $\text{low}(k) = j$ then σ_j is paired with the

In the following, an example of persistent homology computation is given.

Figure 18 – A total order on simplices (compatible with the filtration of Figure 16).

[illegible]

From the boundary matrix B , we read the following intervals:

- * σ_1 is unpaired, this gives the interval $[1, +\infty)$ in H_0 .
- * σ_2 is paired with σ_5 and σ_9 ; this gives the interval $[1, 2)$ and $[1, 4)$ in H_0 .
- * σ_4 is paired with σ_9 ; this gives the interval $[1, 4)$ in H_0 .
- * σ_9 is paired with σ_{11} ; this gives the interval $[4, 6)$ in H_1 .
- * σ_{10} is paired with σ_{11} ; this gives the interval $[5, 6)$ in H_1 .

2. **Other algorithms:** Several new algorithms have been developed with reduction strategies. Each of these algorithms gives the same output for the computation of persistent homology, so we give a brief overview and some references to these algorithms. The *Twist algorithm* is based on the Standard algorithm. It exploits the observation that a column will eventually be reduced to an empty column if its index appears as the pivot of another column. By reducing columns in decreasing order of the dimensions of the corresponding cells, we can explicitly clear the columns corresponding to pivot indices. For more detail see (CHEN; KERBER, 2011). The *Row algorithm* is a sequential algorithm. The idea behind this algorithm is to traverse the columns from right to left and, whenever the pivot of a newly inspected column A equals the pivot of column B to its right, we add A to B (SILVA; MOROZOV; VEJDEMO-JOHANSSON, 2011). The *Dual algorithm* is a dualization algorithm (SILVA; MOROZOV; VEJDEMO-JOHANSSON, 2011). This algorithm is known to give a speed-up when one computes persistent homology with the VR complex, but not necessarily for other types of complexes. Among the parallel algorithms, we include the *Spectral-sequence algorithm* (See Section VII.4 of Edelsbrunner and Harer (2010)), which is a parallel algorithm, and the *Chunk algorithm* (BAUER; KERBER; REININGHAUS, 2014a).

SOFTWARE FOR COMPUTING PERSISTENT HOMOLOGY

This chapter presents the software named CGAL. This software allows constructing filtered cell complexes using efficient geometric algorithms. In the following section, an overview of the available libraries for the computation of persistent homology is given.

4.1 CGAL

The *Computational Geometry Algorithms Library* (CGAL) (www.cgal.org) is a software library of computational geometry algorithms. The library is supported on a number of platforms, such as: GNU g++, MS Visual C++, Intel C++, Solaris, Linux, and Mac OS. CGAL can be used in various areas such as computer-aided design, geographic information systems, medical imaging, computer graphics, molecular biology, and robotics. Further, the library of CGAL covers topics like triangulations, Voronoï diagrams, Delaunay triangulation, arrangements of curves, surface and volume mesh generation, α -shapes, geometry processing, interpolation, convex hull algorithms, and shape analysis. The CGAL project was founded in 1996. For more details see [CGAL \(1995\)](#).

In the following, some packages used to obtain the filtration of the weighted α -shape are presented.

Packages Overview

- **3D Convex Hulls:** This package offers functions for computing convex hulls in three dimensions. Indeed, exists two ways for computing the convex hull of a set of points in $\mathbb{R}^3$, namely, using a static algorithm or using a triangulation to get a fully dynamic computation. Further, this package provides functions for checking if sets of points are

strongly convex or not.

- **3D Triangulation:** This package provides functions to build and handle triangulations for point sets in $\mathbb{R}^3$. Moreover, the convex hull of a set of vertices is always covered by any *CGAL* triangulation. This package permits to build incrementally the triangulations and they can be modified by insertion, displacements or removal of vertices. Another benefit this package is that provides plain triangulation (where the faces depend on the insertion order of the vertices) and also the Delaunay triangulations. Further, regular triangulations are provided for sets of weighted points. Last, the Delaunay and regular triangulations offer primitives and nearest neighbor queries to build the power diagrams and the dual Voronoi.
- **3D Triangulation Data Structure:** This package gives a data structure to store a three-dimensional triangulation with the topology of a three-dimensional sphere. Moreover, the package works as a container for the vertices and cells of the triangulation, providing basic combinatorial operations on the triangulation.
- **3D Alpha Shapes:** This package provides a data structure encoding either one alpha complex or the whole family of alpha complexes related to a given 3D Delaunay or regular triangulation. In the latter case, the data structure allows retrieving the alpha complex for some α -values. More specifically, we can obtain the whole spectrum of critical α -values, and the filtration on the triangulation faces. Moreover, this filtration is based on the first α -value for which each face is included on the alpha complex.
- **3D Point Set:** This component offers a flexible 3D point set data structure. Further, the user can define any additional property needed such as normal vectors, labels or colors. To this data structure, the *CGAL* algorithms can be easily applied.

Once one has constructed the filtration of cell complexes, in the following section, we will use this filtration as input for the software that calculates persistent homology. In this way, we give an overview of the available libraries and summarize their properties.

4.2 Software for computing *PH*

Perseus

The *Perseus* software project (<http://people.maths.ox.ac.uk/nanda/perseus/index.html>) was developed to implement Morse-theoretic reductions. *Perseus* computes the persistent homology of different types of filtered cell complexes such as simplicial complex, Vietoris-Rips complex, dense cubical grid, and sparse cubical grid. In this way, *Perseus* calculates the persistent homology these complexes after first performing certain homology-preserving Morse

theoretic reductions (NANDA, 2012). For example, for dealing with movies and images, it is recommendable to work with cubical data structures. But, if the data source is a manifold triangulation, then the appropriate representation consists of top-cell information on a simplicial complex. Moreover, point cloud data is usually handled effectively with Vietoris-Rips complexes built around those points.

PHAT

The *Persistent Homology Algorithms Toolbox (PHAT)* (<https://github.com/blazs/phat>) is a C++ library for the computation of persistent homology by matrix reduction. The purpose of *PHAT* is to provide a platform for comparative evaluation of existing and new algorithms and data structures for matrix reduction. *PHAT* is among the fastest codes for computing persistent homology currently available and it can be obtained under the GNU Lesser General Public License (BAUER *et al.*, 2014). *PHAT* contains code for several algorithmic variants such as the standard algorithm, the row algorithm, the twist algorithm, and the chunk algorithm. Further, computing persistent homology for a given dataset requires the construction of a filtered cell complex. So, a filtered cell complex is represented by its boundary matrix whose indices correspond to the ordering of the cells, and whose entries encode the boundary relation of the complex. In this way, the main goal of *PHAT* is the computation of the persistent homology of a boundary matrix in a simple and efficient way Bauer *et al.* (2014).

JavaPlex

The *JavaPlex* software package (<https://code.google.com/archive/p/javaplex>) was developed by the computational topology group at Stanford University. This software is based on the PLEX library (TAUSZ; VEJDEMO-JOHANSSON; ADAMS, 2011). The main goal of the *JavaPlex* package is to provide an extensible base to support new avenues for research in computational homology and data analysis. *JavaPlex* can be run either as a Java application, or it can be called from Matlab in jar form.

jHoles

jHoles (<https://doi.org/10.1016/j.entcs.2014.06.011>) is a Java library for computing the weight rank clique filtration for weighted undirected networks. As *jHoles* is developed in Java, it is compatible with every operating system that supports a JVM, but it requires Java 1.7. *jHoles* persistent homology engine is JavaPlex (BINCHI *et al.*, 2014). In this way, *jHoles* is designed to be easily used even by non-computer scientists. Its main point of access is *jHoles*, a class offering all the methods to process a graph. This architectural choice was made to keep it simple to use, grouping in a single class its core functions.

Dionysus

Dionysus (<http://www.mrzv.org/software/dionysus/>) is a C++ library for computing persistent homology (MOROZOV, 2012). It was the first software package to implement the dual algorithm (SILVA; MOROZOV; VEJDEMO-JOHANSSON, 2011).

DIPHA

A *Distributed Persistent Homology Algorithm* (DIPHA) (<https://github.com/DIPHA/dipha>) is a C++ software package that computes persistent homology following the algorithm proposed by Bauer, Kerber and Reininghaus (2014c). Besides supporting parallel execution on a single machine, DIPHA may also be run on a cluster of several machines using MPI (BAUER; KERBER; REININGHAUS, 2014b). To achieve good performance DIPHA supports dualized computation. This software is inclined to make use of the optimization and employs an efficient data structure developed in the PHAT project as described in Bauer *et al.* (2014).

Gudhi

The *Gudhi* library (<https://project.inria.fr/gudhi/software>) is a generic open source C++ library for Computational Topology and TDA. The *Gudhi* library intends to help the development of new algorithmic solutions in TDA and their transfer to applications. It provides efficient, robust, flexible and easy-to-use implementations of algorithms and data structures (MARIA *et al.*, 2014). The *Gudhi* project also contributes to the development of higher dimensional features in the CGAL library (e.g., Delaunay and weighted Delaunay triangulations).

SimpPers

The *SimPers* software (<http://web.cse.ohio-state.edu/dey.8/SimPers/Simpers.html>) for Topological Persistence under Simplicial Maps. *SimPers* can be used in the following case: given a sequence of simplicial maps $f_1, f_2, \dots, f_n$ between an initial simplicial complex K and a resulting simplicial complex L . *Simpers* uses the annotation-based method developed in Dey, Fan and Wang (2014) to compute the persistence of the sequence of simplicial maps.

Ripser

Ripser (<https://github.com/Ripser/ripser>) is a lean C++ code for the computation of Vietoris-Rips persistence barcodes (BAUER, 2015). The *Ripser* library is the most recently developed software. This software uses several optimizations and shortcuts to speed up the computation of persistent homology using the Vietoris-Rips complex (OTTER *et al.*, 2017).

In Table 3, we summarize the properties of the libraries used for the computation of persistent homology.

Table 3 – Overview of existing software for the computation of *Persistent Homology*.

Software	Perseus	PHAT	JavaPlex	jHoles	Dionysus	DIPHA	Gudhi	SimpPers	Ripser
• Language	C++	C++	Java	Java	C++	C++	C++	C++	C++
• Algorithms for PH	standard, Morse reductions	standard, chunk, spectral sequence, dual, twist	standard, dual	standard (Uses JavaPlex)	standard, dual, zigzag	twist, dual, distributed	dual, multifield	simplicial map	twist, dual
• Coefficient field	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Q}, \mathbb{Z}_p$	$\mathbb{Z}_2$	$\mathbb{Z}_2(\text{zigzag, standard})$ $\mathbb{Z}_p(\text{dual})$	$\mathbb{Z}_2$	$\mathbb{Z}_p$	$\mathbb{Z}_2$	$\mathbb{Z}_p$
• Homology	cubical, simplicial	cubical, simplicial	cellular, simplicial	simplicial	simplicial	cubical, simplicial	cubical, simplicial	simplicial	simplicial
• Filtrations computed	VR, lower star of cubical complex	–	$VR, \mathcal{W}, \mathcal{W}_v$	WRCF	VR, alpha complex, Čech complex	VR, lower star of cubical complex	VR, $\mathcal{W}$, lower star of cubical complex, alpha complex	–	VR
• Filtrations as input	simplicial complex, cubical complex	boundary matrix of simplicial complex	simplicial complex, zigzag, CW	–	simplicial complex, zigzag	boundary matrix of simplicial complex	–	map of simplicial complexes	–
• Visualization	persistence diagrams	–	barcodes	–	–	persistence diagrams	–	–	–

Source: Adapted from [Otter et al. \(2017\)](#).

$\mathcal{W}$: weak witness complex, $\mathcal{W}_v$: parametrized witness complexes, WRCF: weight rank clique filtration, VR: Vietoris-Rips complex. The symbol (–) signifies that the associated feature is not implemented.

MACHINE LEARNING

Machine learning is becoming one of the most active areas of research in computer science and data analysis in recent years. One of the reasons for this is its great number of successful applications in many different areas of science (GOODFELLOW *et al.*, 2016; BISHOP, 2006; ROGERS; GIROLAMI, 2016). *Machine learning* (ML) is the systematic study of algorithms and systems that improve their knowledge or performance with experience (FLACH, 2012). ML can be broadly divided into two main areas: *supervised learning* and *unsupervised learning*. In supervised learning, we have a dataset, called training dataset, for which the answers to the questions we are interested to know, and this dataset is used to “train our machine”. We then use our “trained machine” to obtain the answers to our questions for other datasets that we call “testing set”. In unsupervised learning, on the other hand, we want to extract information (such as clustering information for example) from our dataset without the aid of a training dataset.

One of the main tasks in supervised learning is the *classification*: given a dataset, each element of this set is to be classified as belonging to one of the predetermined collection of classes. This can be described more formally as follows. Let X be a vector space, the elements of which are called *feature vectors* and are meant to represent the features used to describe our objects. Let $C = \{c_1, c_2, \dots, c_d\}$ ($d \in \mathbb{N}$) be a set of *class labels*. An example of the classification problem would be the digit recognition example, in which the aim is to assign each input vector to one of a finite number of discrete categories. The goal of *supervised classification* is to classify each element of X as belonging to one of the classes given by C . To this end, assume that we are giving a set of pairs $\{(x_1, c_1), (x_2, c_2), \dots, (x_N, c_N)\} \subset X \times C$ ($N \in \mathbb{N}$) so-called the *training dataset*. Given one such pair (x_i, c_i) with $i = 1, \dots, N$, we say that the vector x_i belongs to the class labeled by c_i . Supervised machine learning classifies the elements of X by using the training dataset to “learn” (or “train”) a parameter dependent function $g: X \times \mathbb{R}^m \rightarrow C$ satisfying some given optimality conditions and such that $g(x_i, \alpha) = c_i$, for all $i = 1, \dots, N$. Learning the function g means finding a value for the parameter $\alpha = \alpha_0$ such that $f(x) := g(x, \alpha_0)$ satisfies all the required conditions. Once we have the trained function $f: X \rightarrow C$, we define the class to which

a vector $x \in X$ belong to be the class whose label is given by $f(x) \in C$.

Other of the main tasks in supervised learning is the *regression*. Generally, in regression problems one of the variables, usually called *dependent variable* is of particular interest and it is denoted by “y”. The other variables $x_1, x_2, \dots, x_n$, usually called *independent variables*, are used to predict the behaviour of “y”. In this way, if plots of the data suggest some relationship between y and the x_i ’s, then we hope to express this relationship via some function f , namely, $y \approx f(x_1, x_2, \dots, x_n)$. This relationship between independent and dependent variables is maintained, except when there exists some unknown constants or coefficient (called parameters), in this case, the relationship must be written as $y \approx f(x_1, x_2, \dots, x_n; \theta)$, where f is entirely known except for a parameter vector $\theta = (\theta_1, \theta_2, \dots, \theta_p)^t$, which needs to be estimated. For example, the model $y \approx \theta_1 x_1 + \theta_2 e^{\theta_3 x_2}$ with $\theta = (\theta_1, \theta_2, \theta_3)^t$ is nonlinear in θ_3 . In this work, specifically the models of the form $y = f(x; \theta)$ are considered, with $\theta = (\theta_1, \theta_2, \dots, \theta_p)^t$ being a vector of unknown parameters. For more details see [Wild and Seber \(2003\)](#).

In the next section, the description of some algorithms proposed for classification is presented. For a more in-depth discussion please see [Mitchell \(1997\)](#), [Bishop \(2006\)](#), [Hearst \(1998\)](#), [Barker and Rayens \(2003\)](#), [Theodoridis and Koutroumbas \(2008\)](#).

5.1 Classification

Supervised classification of data is one of the main tasks in Machine Learning. There are several algorithms to do classification, in this work we used only *Naive Bayes*, *Support Vector Machine* (SVM), and *Partial Least Squares-Discriminant Analysis* (PLS-DA).

5.1.1 Naive Bayes

Naive Bayes is a probabilistic learning algorithm based on the Bayes’ rule, taking its name from Equation (5.1) on which it is based. It is useful in tasks where the information to be classified is incomplete or imprecise ([BISHOP, 2006](#)). Bayes’ theorem provides a way to calculate the probability of a hypothesis based on its prior probability, the probabilities of observing various data given the hypothesis, and the observed data itself.

To define Bayes’ theorem, first some notations are introduced. Let $P(h)$ be the initial probability that the hypothesis h holds before the training data is observed. $P(h)$ is called the *prior probability* of h , and it may reflect any background knowledge that we have about the chance that h is a correct hypothesis. In this way, if we have no such prior knowledge then we might simply assign the same prior probability to each candidate hypothesis. Similarly, $P(D)$ denotes the prior probability that training data D will be observed (i.e., the probability of D given no knowledge about which hypothesis holds). $P(D|h)$ denotes the probability of observing data D given some world in which hypothesis h holds. So we are interested in the probability $P(h|D)$ that h holds given the observed training data D , where $P(h|D)$ is called the *posterior probability*

of h because it reflects our confidence that h holds after we have seen the training data D . Notice the posterior probability $P(h|D)$ reflects the influence of the training data D , in contrast to the prior probability $P(h)$, which is independent of D .

Bayes' theorem is the cornerstone of Bayesian learning methods because it provides a way to calculate the posterior probability $P(h|D)$, from the prior probability $P(h)$ together with $P(D)$ and $P(D|h)$ (MITCHELL, 1997). Bayes' theorem is stated mathematically as the following equation:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}, \quad (5.1)$$

where $P(D) \neq 0$.

In many learning scenarios, first some set of candidate hypotheses H is considered, and then finding the most probable hypothesis $h \in H$ given the observed data D (or at least one of the maximally probable if there are several). Any such maximally probable hypothesis is called a *Maximum a posteriori* (MAP) hypothesis. It's possible to determine the MAP hypothesis by using Bayes' theorem to calculate the posterior probability of each candidate hypothesis. Then, h_{MAP} is a MAP hypothesis provided, that is,

$$\begin{aligned} h_{\text{MAP}} &\equiv \arg \max_{h \in H} P(h|D) \\ &= \arg \max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \arg \max_{h \in H} P(D|h)P(h). \end{aligned} \quad (5.2)$$

Notice in the final step above the term $P(D)$ is dropped because it is a constant independent of h .

In some cases, if it's assumed that every hypothesis in H is equally probable a priori ($P(h_i) = P(h_j)$ for all h_i and h_j in H) then Equation (5.2) could be simplified in Equation (5.3). In this case, the term $P(D|h)$ is only considered to find the most probable hypothesis. Also, $P(D|h)$ is called the *likelihood* of the data D given h , and any hypothesis that maximizes $P(D|h)$ is called a *maximum likelihood (ml)* hypothesis h_{ml} , that is,

$$h_{\text{ml}} = \arg \max_{h \in H} P(D|h). \quad (5.3)$$

Next, we will deal specifically with Bayes classifier. In this sense, the output of Naive Bayes classifier is the probability of a new object belonging to a particular class (ROGERS; GIROLAMI, 2016).

Let X be a set of n training objects $x_1, x_2, \dots, x_n$, where each x_i is a vector with dimension d , and a set of m labels S . For each object x_i , a label $c \in S$ is provided that it will describe which class the object x_i belongs to. Each label $c \in S$ could be taken an positive integer value, this is, if there are m classes then $S = \{1, 2, \dots, m\}$. In this way, given a training set $X = \{x_1, x_2, \dots, x_n\}$ from m classes, the aim is to be able to compute the predictive probabilities (Equation (5.4)) for each of potential class $c \in S$. More specifically, our task is to predict the class T_{new} for an unseen

object x_{new} , and this probability for the class $c \in S$ is given by,

$$P(T_{\text{new}} = c | x_{\text{new}}, X, S). \quad (5.4)$$

As (5.4) is a probability, it must satisfy the following conditions:

$$\begin{aligned} 0 &\leq P(T_{\text{new}} = c | x_{\text{new}}, X, S) \leq 1 \\ \sum_{c=1}^m P(T_{\text{new}} = c | x_{\text{new}}, X, S) &= 1. \end{aligned}$$

From Bayes' rule (5.1), the following expression for the predictive probability is obtained:

$$P(T_{\text{new}} = c | x_{\text{new}}, X, S) = \frac{p(x_{\text{new}} | T_{\text{new}} = c, X, S) \cdot P(T_{\text{new}} = c | X, S)}{p(x_{\text{new}} | X, S)}, \quad (5.5)$$

where $p(x_{\text{new}} | T_{\text{new}} = c, X, S)$ is a distribution specific to the class c (it's conditioned on $T_{\text{new}} = c$) evaluated at x_{new} , $P(T_{\text{new}} = c | X, S)$ is the prior probability of the class c conditioned on just the training data X , and last $p(x_{\text{new}} | X, S)$ is the *marginal likelihood*. For more details see [Rogers and Girolami \(2016, p. 171\)](#).

Also, in (5.5) the marginal likelihood $p(x_{\text{new}} | X, S)$ can be expanded to a sum over the m possible classes. In this way, the Bayes classifier can be defined by:

$$P(T_{\text{new}} = c | x_{\text{new}}, X, S) = \frac{p(x_{\text{new}} | T_{\text{new}} = c, X, S) \cdot P(T_{\text{new}} = c | X, S)}{\sum_{c'=1}^m p(x_{\text{new}} | T_{\text{new}} = c', X, S) \cdot P(T_{\text{new}} = c' | X, S)}. \quad (5.6)$$

5.1.2 Support Vector Machine

Support vector machine (SVM) is a statistical learning algorithm that is very successful and popular in the literature because it obtains results equivalent or superior to other algorithms in different applications.

The basic idea of the SVM algorithm is to find a nonlinear mapping of the training data into a higher-dimensional feature space, and construct a separating hyperplane with maximum margin. This will yield a nonlinear decision boundary in the input space. In this way, by the use of a kernel function, it is possible to compute the separating hyperplane without explicitly carrying out the map into a feature space ([HEARST, 1998](#)). Next, we will deal specifically with the SVM algorithm.

Let $\mathcal{F}$ be a feature space, and Φ is a nonlinear map $\Phi : \mathbb{R}^n \rightarrow \mathcal{F}$. To construct a hyperplane in a feature space $\mathcal{F}$, one first has to transform the n -dimensional input vector x into an m -dimensional feature vector through a choice of an m -dimensional vector *function* $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ($\mathbb{R}^m \subset \mathcal{F}$), this is,

$$x \in \mathbb{R}^n \rightarrow \Phi(x) = (\phi_1(x), \phi_2(x), \dots, \phi_m(x)) \in \mathbb{R}^m.$$

A mapping $\Phi(x)$ is chosen in advance, i.e., it is a fixed function. An input space $\mathbb{R}^n$ is spanned by components x_i of an input vector x , and a feature space $\mathcal{F}$ is spanned by components $\phi_i(x)$ of a vector $\Phi(x)$. By performing such a mapping, we hope that in a feature space $\mathcal{F}$, the learning algorithm will be able to linearly separate images of x by applying the linear SVM formulation presented above. In this way, if a projection $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is used, the dot product can be represented by a *kernel* function $k : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{K}$ (field $\mathbb{K}$) as being,

$$k(x, y) = \Phi^t(x) \cdot \Phi(y), \quad (5.7)$$

for all $x, y \in \mathbb{R}^n$. Thus, by using the chosen kernel $k(x, y)$, we can construct an SVM that operates in an infinite dimensional space. In addition, by applying the kernels we don't even have to know what the actual mapping $\Phi(x)$. For more details see [Wang \(2005\)](#).

There are many possible kernels used with SVM. In Table 4, the most popular ones are presented.

Table 4 – Popular admissible Kernels.

Kernel functions	Type of classifier
$k(x, y) = \exp(-\ x - y\ ^2 / (2\sigma^2))$	Gaussian RBF
$k(x, y) = x^t \cdot y$	Linear kernel
$k(x, y) = (x^t \cdot y + 1)^d$	Polynomial of degree d
$k(x, y) = \tanh(x^t \cdot y + b)$ (*)	Multilayer perceptron
$k(x, y) = \frac{1}{\sqrt{\ x - y\ ^2 + \beta}}$	Inverse multiquadric function

(*) only for certain values of b , RBF = Radial basis function.

Source: Adapted from [Wang \(2005\)](#).

When the classes are not always linearly separable in a feature space, the solution is to build a decision function that is not linear. This is done by using the kernel trick that can be seen as creating an decision energy by positioning kernels on observations. In this work, the Gaussian radial basis function (RBF) kernels is used, this is,

$$k(x, y) = \exp(-\|x - y\|^2 / (2\sigma^2)),$$

for all $x, y \in \mathbb{R}^n$ and σ is a free parameter.

SVM is based on the class of hyperplane $w^t \cdot \Phi(x) + b = 0$ (w is a vector in $\mathbb{R}^m$, $b \in \mathbb{R}$), corresponding to the decision function $f(x)$ given by

$$f(x) = \text{sign}(w^t \cdot \Phi(x) + b). \quad (5.8)$$

The distance of the hyperplanes to the origin is given by $\rho = \frac{b}{\|w\|}$. Then, if the objective is to maximize the margin between the hyperplanes then we have to minimize $\|w\|$.

Thereby, we have all the tools to construct the nonlinear classifier. In this way, $\Phi(x_i)$ is substituted for each training sample $x_i \in \mathbb{R}^n$, and the optimal hyperplane algorithm in $\mathcal{F}$ is performed. Due to the use of kernels, Equation (5.8) ends up with nonlinear decision function of the form:

$$f(x) = \text{sign}\left(\sum_{i=1}^l v_i \cdot k(x, x_i) + b\right),$$

where the parameters v_i are computed as the solution of a quadratic programming problem in terms of the kernels.

Finally, there are two main categories for Support Vector Machines: Support Vector Classification (SVC) and Support Vector Regression (SVR). The model produced by SVC only depends on a subset of the training data, because the cost function for building the model doesn't care about training points that lie beyond the margin. The model produced by SVR only depends on a subset of the training data, because the cost function for building the model ignores any training data that is close to the model prediction. For more details see Subsection 5.2.1.

5.1.3 Partial least squares-discriminant analysis

Partial Least Squares-Discriminant Analysis (PLS-DA) is a classification method that combines the properties of Partial Least Squares (PLS) regression with a discrimination technique. In this sense, PLS-DA is based on the PLS-regression algorithm. Also, PLS-DA is a multivariate dimensionality-reduction tool popular in the field of chemometrics (WOLD; SJÖSTRÖM; ERIKSSON, 2001) and omics analyses (BALLABIO; CONSONNI, 2013; BARKER; RAYENS, 2003; STÄHLE; WOLD, 1987; WORLEY; HALOUSKA; POWERS, 2013).

PLS-regression is a tool adapted to predict a dependent variable from an unlimited number of possibly correlated explanatory features. In the case of PLS-DA, it tries to find the sharpen separation between the classes of samples, understanding which variables more carry the labels information and obtaining the maximum separation among classes (WOLD; SJÖSTRÖM; ERIKSSON, 2001). Next, we will deal specifically with the PLS-DA algorithm.

Using the continuous predictor variables $X_1, X_2, \dots, X_p$, we want to predict q continuous response variables $Y_1, Y_2, \dots, Y_q$. Considering the available data sample $(\tilde{x}_i, \tilde{y}_i)_{i=1, \dots, n}$ of n observations, where $\tilde{x}_i$ denotes the i -th observation of the predictor and $\tilde{y}_i$ denotes the i -th observation of response variables. As each i -th observation is an uncentered sample, we need to centralize the samples, this is,

$$x_i = \tilde{x}_i - \frac{1}{n} \sum_{s=1}^n \tilde{x}_s \quad \text{and} \quad y_i = \tilde{y}_i - \frac{1}{n} \sum_{s=1}^n \tilde{y}_s.$$

In this sense, (x_i, y_i) is called the i -th *centered sample*, with $i = 1, \dots, n$.

The *data matrix* and *response matrix* are denoted by

$$\mathbf{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad \text{and} \quad \mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix},$$

respectively. The $n \times p$ matrix $\mathbf{X}$ is formed by the row vectors $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})$ with $i = 1, \dots, n$. Analogous, the $n \times q$ matrix $\mathbf{Y}$ is formed by the row vectors $y_i = (y_{i1}, y_{i2}, \dots, y_{iq})$ with $i = 1, \dots, n$.

PLS-DA is based on the basic latent component decomposition:

$$\begin{aligned} \mathbf{X} &= \mathbf{T} \cdot \mathbf{P}^t + \mathbf{E}, \\ \mathbf{Y} &= \mathbf{T} \cdot \mathbf{Q}^t + \mathbf{F}, \end{aligned} \tag{5.9}$$

where $\mathbf{T}$ is a $n \times c$ matrix that giving the latent components for the n observations, $\mathbf{P}$ is a $p \times c$ matrix of coefficients, $\mathbf{Q}$ is a $q \times c$ matrix of coefficients, $\mathbf{E}$ is a $n \times p$ matrix of random errors, and $\mathbf{F}$ is a $n \times q$ matrix of random errors (BRERETON; LLOYD, 2014; KUHN, 2016).

PLS-DA can be seen as a method to construct a matrix of *latent components* $\mathbf{T}$ as a linear transformation of $\mathbf{X}$, this is,

$$\mathbf{T} = \mathbf{X} \cdot \mathbf{W},$$

where $\mathbf{W}$ is a $p \times c$ matrix of weights formed by the column vectors $w_j = (w_{1j}, w_{2j}, \dots, w_{pj})^t$ with $j = 1, \dots, c$. The columns of $\mathbf{T}$ are often denoted as *latent variables* or *scores*. In this way, for a fixed matrix of weights $\mathbf{W}$, the random variables obtained by forming the corresponding linear transformations of the predictor variables $X_1, X_2, \dots, X_p$ are denoted as being:

$$\begin{aligned} T_1 &= w_{11}X_1 + \dots + w_{p1}X_p, \\ \vdots &= \vdots \\ T_c &= w_{1c}X_1 + \dots + w_{pc}X_p. \end{aligned}$$

In place of the original variables be used the latent components for prediction. More specifically, once $\mathbf{T}$ is constructed, $\mathbf{Q}^t$ is obtained as the least squares solution of Equation (5.9), this is, $\mathbf{Q}^t = (\mathbf{T}^t \mathbf{T})^{-1} \mathbf{T}^t \mathbf{Y}$.

Now, let $\mathbf{B}$ be the matrix of regression coefficients for model $\mathbf{Y} = \mathbf{X}\mathbf{B} + \mathbf{F}$ where $\mathbf{F}$ is the matrix of random errors, this is, $\mathbf{B} = \mathbf{W}\mathbf{Q}^t = \mathbf{W}(\mathbf{T}^t \mathbf{T})^{-1} \mathbf{T}^t \mathbf{Y}$. Hence, the fitted response matrix $\mathbf{Y}$ may be written as:

$$\hat{\mathbf{Y}} = \mathbf{T}(\mathbf{T}^t \mathbf{T})^{-1} \mathbf{T}^t \mathbf{Y}. \tag{5.10}$$

In this way, if we have a new uncentered observation $\tilde{x}_0$ then the prediction $\hat{y}_0$ of the response will be writing as:

$$\hat{y}_0 = \frac{1}{n} \sum_{i=1}^n \tilde{y}_i + \mathbf{B}^t \left(\tilde{x}_0 - \frac{1}{n} \sum_{i=1}^n \tilde{x}_i \right).$$

The basic idea of the PLS-DA classifier is that the response matrix $\mathbf{Y}$ should be taken into account for the construction of the matrix components $\mathbf{T}$. More specifically, the components of $\mathbf{T}$ are defined such that they have high covariance with the response $\mathbf{Y}$.

In summary, PLS-DA looks for the variables that best correlate with the classifier. These variables would have a high weight for the more significant PLS-components and they form a model for which appears that the classes are separated. So, when the number of variables exceeds the number of samples, the predictions appear to be very good. In this sense, the samples classified correctly into their respective groups is observed. On the other hand, with numerous correlated variables, there is a substantial risk for over-fitting, more specifically, getting a well-fitting model without predictive power. Finally, a strict test of the predictive significance of each PLS-component is necessary, and then it is stopping when components start to be non-significant (WOLD; SJÖSTRÖM; ERIKSSON, 2001).

5.2 The Nonlinear Regression

A nonlinear regression model can be written as

$$Y_n = f(x_n, \theta) + L_n, \quad (5.11)$$

with $n = 1, 2, \dots, N$, where f is the expectation function, x_n is a vector of associated regressor variables or independent variables, θ is a set of parameters, and L_n is a random variable called *disturbance*. In the nonlinear model, one of the derivatives of the expectation function with respect to the parameters depends on at least one the parameters. In Equation (5.11), the random variable Y_n represents the *response* for the n -th case.

When analyzing a particular set of data we consider the vectors x_n , $n = 1, 2, \dots, N$, as fixed and concentrate on the dependence of the expected responses on θ . The N -vector $\eta(\theta)$ was created, this is,

$$\eta_n(\theta) = f(x_n, \theta), \quad n = 1, \dots, N,$$

and the nonlinear regression model for N case can be written as

$$Y = \eta(\theta) + L, \quad (5.12)$$

where we assumed that L has a spherical normal distribution. That is,

$$\begin{aligned} E[Z] &= 0, \\ \text{Var}(Z) &= E[ZZ^t] = \sigma^2 I, \end{aligned}$$

where I is the $N \times N$ identity matrix and σ^2 is the variance. For more details see Bates and Watts (2007, p. 32).

There are several algorithms to do regression analysis, in this work we used only *Support Vector Regression*.

5.2.1 Support Vector Regression

A version of SVM for regression has been proposed by [Vapnik, Golowich and Smola \(1996\)](#). This version is called *Support Vector Regression* (SVR). An overview of the basic ideas underlying SVR has been given in [Basak, Pal and Patranabis \(2007\)](#). We are interested in the nonlinear case.

Considering the training data $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \subset X \times \mathbb{R}$, where X is the space of the input patterns and $Y = \{y_1, \dots, y_n\}$ is the vector of labels. The next step consists in to make a preprocessing the training patterns x_i by a map $\Phi : X \rightarrow \mathcal{F}$, into some feature space $\mathcal{F}$ and then applying the standard SV algorithm of [Smola and Schölkopf \(2004\)](#). The SV algorithm only depends on the linear dot product between nonlinear mapping Φ , this is, $k(x_i, x_j) := \langle \Phi(x_i), \Phi(x_j) \rangle$. In this sense, it suffices to know the kernel $k(x_i, x)$ rather than Φ explicitly which allow us to restate the SV optimization problem.

The Support Vector expansion w can be described as combination of $\Phi(x_i)$, this is,

$$w = \sum_{i=1}^n (\alpha_i - \alpha_i^*) \Phi(x_i), \quad (5.13)$$

with α_i^* and α_i are Lagrangian multipliers in the dual problem. The difference with the linear case is that w is no longer explicitly given. Then, the standard SVR to solve the approximation problem is given by

$$f(x) = \sum_{i=1}^n (\alpha_i - \alpha_i^*) k(x_i, x) + b. \quad (5.14)$$

For evaluating $f(x)$, it isn't needed to compute w explicitly. For calculating b , it's necessary to exploit Karush-Kuhn-Tucker (KKT) conditions giving in [Smola and Schölkopf \(2004\)](#). In summary for the nonlinear case, the optimization problem consists to find the flattest function in the feature space and not in input space.

The coefficients α_i and α_i^* in (5.14) was obtained by minimizing the following regularized risk functional:

$$R_{reg}[f] = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n L_\varepsilon(y), \quad (5.15)$$

where the term $\|w\|^2$ has been characterized as model complexity, C as a constant determining the trade-off, and the ε -intensive loss function $L_\varepsilon(y)$ is given by

$$L_\varepsilon(y) = \begin{cases} 0 & , \text{ for } |f(x) - y| < \varepsilon, \\ |f(x) - y| - \varepsilon & , \text{ otherwise.} \end{cases}$$

Last, in ε -SV regression, the goal of [Smola and Schölkopf \(2004\)](#) is to find a function $f(x)$ that will be flat and at the same time, it has minor errors that ε . In classical SVR is difficult to determine in advance the proper value for the parameter ε . But this problem is partially solved in a new algorithm called ν -Support Vector Regression (ν -SVR), in which ε is a variable in the

optimization process and it's controlled by another new parameter $\nu \in (0, 1)$. For more details see [Smola and Schölkopf \(2004\)](#).

5.3 Some statistical measures

The algorithms of supervised classification try to induce a predictive model based on a set of labeled data. A *classification model* (or *classifier*) is a mapping from samples to predicted classes. Some classification models produce a continuous output to which different thresholds may be applied to predict class membership. Other classifiers produce a discrete class label indicating only the predicted class of the sample. To distinguish between the actual class and the predicted class we use the labels $\{Y, N\}$ for the class predictions produced by a classifier. For more details see [Fawcett \(2006\)](#).

Given a classifier and set of samples (testing dataset), a *confusion matrix* (M), also called *contingency table*, can be constructed representing the disposition of the set of samples. The confusion matrix collects the outputs of the classification models and it is the initial step for evaluating the classification performance. More specifically, M is a square matrix that collects the actual and predicted sample class with dimensions $c \times c$, where c is the number of classes. The matrix entries on the main diagonal represent the number of samples assigned to the correct class, while off-diagonal entries represent classification errors, i. e., the columns of M represent the sample in actual classes (true condition) whereas the rows represent the samples in predicted classes.

In Figure 19 we have the two-by-two confusion matrix, where the *True Positive* (TP) is the number of actual class samples correctly predicted in the class, *True Negative* (TN) is the number of samples that are not part of the class and were correctly predicted as no-class samples, *False Negative* (FN) the number of actual class samples incorrectly predicted as no-class samples, and *False Positive* (FP) the number of actual no-class samples incorrectly predicted as belonging to the class.

Figure 19 – Confusion matrix for a disjoint two-class problem.

		True condition	
		Class	
Predicted condition	A	TP	FP
	B	FN	TN

Source: Elaborated by the author.

Now we presented several common metrics calculated from the two-by-two confusion matrix given in Figure 19. For a discussion of those measures see [Fawcett \(2006\)](#) .

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

$$F_1\text{-score} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Accuracy measures the overall amount of correct identifications from all predictions made by the classifier. The best accuracy is given by the value 1. *Precision* describes the model ability to correctly recognize samples belonging to the class. In contrast, *Recall* describes the model ability to retrieve samples that truly belong to the class, i.e., rejecting samples of all other classes. *F₁-score* is a measure of a test's accuracy. More specifically, *F₁-score* is the harmonic mean of the Precision and Recall measure for a classifier. In most problems, *F₁-score* represents a trade-off between Precision and Recall where increasing one measure will disfavour the other, and *F₁-score* quickly decreases. However, *F₁-score* reach greater values when both Precision and Recall are higher and similar. In this way, the optimal classifier will have higher *F₁-score* more precise (correctly classified samples) and robust (capture of all significant samples). For instance, with high precision but low recall, the classifier is extremely accurate, but it misses a considerable number of significant instances.

For a binary classification problem is easy to compute Precision and Recall but it can be quite confusing to compute these measures for a multi-class classification problem. Now let's look at how to compute Precision and Recall for a three-class problem. In Figure 20, we have the three-by-three confusion matrix where TP_k is the number of actual class samples correctly predicted in the class k with $k = \{A, B, C\}$, and each E_{ij} corresponds to the number of items with true class j that were classified as belonging to the class i with $i, j = \{A, B, C\}$ and $i \neq j$.

Figure 20 – Confusion matrix for a disjoint three-class problem.

		True condition			
		Class	A	B	C
Predicted condition	A	TP_A	E_{BA}	E_{CA}	
	B	E_{AB}	TP_B	E_{CB}	
	C	E_{AC}	E_{BC}	TP_C	

Source: Elaborated by the author.

Now we present several common metrics calculated from the three-by-three confusion matrix given in Figure 20.

$$Accuracy = \frac{TP_A + TP_B + TP_C}{TP_A + E_{AB} + E_{AC} + TP_B + E_{BA} + E_{BC} + TP_C + E_{CA} + E_{CB}}$$

$$Precision_A = \frac{TP_A}{TP_A + E_{BA} + E_{CA}}$$

$$Precision_B = \frac{TP_B}{TP_B + E_{AB} + E_{CB}}$$

$$Precision_C = \frac{TP_C}{TP_C + E_{AC} + E_{BC}}$$

$$Precision = \frac{1}{3}(Precision_A + Precision_B + Precision_C)$$

$$Recall_A = \frac{TP_A}{TP_A + E_{AB} + E_{AC}}$$

$$Recall_B = \frac{TP_B}{TP_B + E_{BA} + E_{BC}}$$

$$Recall_C = \frac{TP_C}{TP_C + E_{CA} + E_{CB}}$$

$$Recall = \frac{1}{3}(Recall_A + Recall_B + Recall_C)$$

$$F_1\text{-score} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Now we will describe some measures used to evaluate a regression model. After one has fit a model using regression analysis, it is necessary to determine how well the model fits the data.

The *coefficient of determination* or the *coefficient of multiple determination for multiple regression*, denoted by *R*-squared (R^2), is a statistical measure of how well the regression predictions approximate the real data points. In the case of simple regression analysis, R^2 measures the proportion of the variance in the dependent variable explained by the independent variable (ALLEN, 1997). This coefficient is computed using either the variance of the errors of prediction or the variance of the predicted values in relation to the variance of the observed values on the dependent variable as follows:

$$R^2 = \frac{Var(\hat{y})}{Var(y)} = 1 - \frac{Var(e)}{Var(y)} \quad (5.16)$$

where $\hat{y}$ are the predicted values, y is the dependent variable, and $e = y - \hat{y}$ is the error of prediction. R^2 ranges from 0 to 1, where the best R^2 is 1.

Equation (5.16) can easily be extended to the case of multiple regression analysis because the variances of the predicted values and the errors of prediction in simple regression have direct counterparts in multiple regression (CAMERON; WINDMEIJER, 1997). In short, the addition of independent variables to the regression model does not affect the equations for computing either the predicted values or the errors of prediction. More precisely, the fundamental relationship between the variance of the dependent variable y , the variance of the predicted values $\hat{y}$, and the variance of the errors of prediction e , remains the same, such that:

$$Var(y) = Var(\hat{y}) + Var(e)$$

Further, R^2 in multiple regression analysis has exactly the same definition as it does in simple regression given in (5.16). More specifically, the interpretation of the coefficient of determination remains the same regardless of how many variables there are in the regression equation. Application of this measure to nonlinear models generally leads to a measure that can lie outside the interval $[0, 1]$ and decrease as regressors are added. Moreover, the desirable properties of an R^2 include interpretation in terms of the information content of the data, and sufficient generality to cover a reasonably broad class of models (CAMERON; WINDMEIJER, 1997).

Another measure to evaluate a regression model is the *Root mean square error* (RMSE) given in (5.17). It is given by the root mean square error of the difference between the values (samples) predicted $\hat{y}$ by a model and the real value y . RMSE is the most commonly used error measure for measuring the quality of a model. This measure can consider it as a measure analogous to the Standard deviation. RMSE is always non-negative and a value of 0 (almost never achieved in practice) would indicate a perfect fit to the data.

$$RMSE = \sqrt{\frac{\sum_i^n (\hat{y}_i - y_i)^2}{n}} \quad (5.17)$$

PROPOSED METHOD

The goal of this work is to apply techniques of *Topological Data Analysis* (TDA), more specifically, we used *Persistent Homology* (PH) to calculate topological features more persistent in the cell complex of an object. In this way, the corresponding *persistence diagrams* (PD) of the object are processed as features for applying the *Machine Learning* (ML) algorithms.

In pre-processing stage, the α -shape filtration of the cell complex is obtained. More, precisely, once we have the alpha complexes filtration, we compute the persistent homology of this filtration. Recall from Section 3.3 that the k -dimensional persistence diagram $PD_k(\mathcal{X})$, $k \in \mathbb{N} \cup \{0\}$ of filtration $\mathcal{X}$ is a multi-set of pairs of points of the form (b, d) , where each pair correspond to the birth and death values of a give k -dimensional hole γ for which γ appears and disappears in the filtration $\mathcal{X}$. So, we have persistent homology intervals containing the information about the birth and death of connected components β_0 , tunnels β_1 , and cavities β_2 .

In processing stage, the numerical attributes (the birth and death values) was extracted of each persistence diagram. More specifically, to extract a feature vector from the persistence diagram $PD_k(\mathcal{X})$, $k = \{0, 1, 2\}$, we fixed $\alpha_{\min} < \alpha_{\max}$ with $\alpha_{\min}, \alpha_{\max} \in \mathbb{R}$, and considered only the persistence points whose birth values are in the interval $[\alpha_{\min}, \alpha_{\max}]$. Now considered a uniform interval $\alpha_{\min} = \alpha_0 < \alpha_1 < \dots < \alpha_m = \alpha_{\max}$ consisting of m subintervals of $[\alpha_{\min}, \alpha_{\max}]$, and let v_j be the number of pairs in $PD_k(\mathcal{X})$, whose birth value b is in the interval $[\alpha_{j-1}, \alpha_j]$. Then the k -dimensional *persistence feature vector* of size m is given by the vector $v^k(\mathcal{X}) = (v_1, v_2, \dots, v_m) \in \mathbb{R}^m$. Consequently, we concatenated these k -vectors and define the new persistence feature vector, $w(\mathcal{X}) := [v^0(\mathcal{X}), \dots, v^k(\mathcal{X})] \in \mathbb{R}^{km}$. Last, the *general matrix* $W(\mathcal{X})$ be constructed using the n new persistence feature vectors.

In the last stage, for classification of dataset and parameter identification in a Predator-Prey System, we used the following machine learning algorithms: *Partial least squares-discriminant analysis* (PLS-DA), *Support vector machine* (SVM), and *Naive Bayes*. For the parameter estimation, we used the machine learning regression methods: *SVR* and *KNeighbors*.

The entire proposed procedure is summarized in the next pipeline (See Figure 21)

Figure 21 – Pipeline about entire proposed method.

Source: Elaborated by the author.

PROTEINS CLASSIFICATION

Topology is the field of mathematics that studies the notion of shape. More specifically, topology covers the study of two main tasks, measurement and representation of the shape (ZOMORODIAN, 2005). Both tasks are relevant in the context of complex and high dimensional datasets because they allow measuring significant properties of the shape related to the data. For these tasks, the α -shape model is employed because it provides a compressed representation of the datasets, it maintains the original features and relationships between the data. The α -shape models were originally introduced for the study of points in the plane (EDELSBRUNNER; G.; SEIDEL, 1983) but later were generalized to points in higher dimensions and weighted points (EDELSBRUNNER; MUCKE, 1994).

With the need of new algebraic topology tools, computational topology (EDELSBRUNNER; HARER, 2010) has recently realized a significant development toward data analysis, giving birth to the field of *Topological Data Analysis* (TDA) (CARLSSON, 2009; EPSTEIN; CARLSSON; EDELSBRUNNER, 2011). TDA provides a framework for analyzing the topological characteristics extracted from the data and it gives a way to understand the overall organization of the data directly. In this sense, TDA has been successfully applied in a great variety of areas, including biology (XIA; WEI, 2014; KASSON *et al.*, 2007), brain science (LEE *et al.*, 2011; SINGH *et al.*, 2008), biochemistry (GAMEIRO *et al.*, 2015), material science (HIRAOKA *et al.*, 2016; NAKAMURA *et al.*, 2015), and information science (CARLSSON *et al.*, 2008; SILVA; GHRIST, 2007). One of the goals of TDA is to detect significant topological properties from the dataset, in order to characterize relevant metric information. More specifically, TDA through *Persistent Homology* (PH) (EDELSBRUNNER, 2014) provides metric information about the topological properties of an object, such as the number of connected components, loops, and cavities.

In this chapter, we propose to apply techniques from TDA, specifically persistent homology, combined with *Machine Learning* (ML) (BISHOP, 2006; GOODFELLOW *et al.*, 2016; ROGERS; GIROLAMI, 2016) to classify proteins sets. More precisely, we compute *PH* of a

filtered simplicial complex (EDELSBRUNNER, 1995) that representing to each protein and use the corresponding *Persistence Diagrams* (PD) as features for ML algorithms.

In the next section, we describe how to use persistent homology of weighted alpha complex to extract features to be used in the machine learning methods (classifiers).

7.1 Proposed Method

Our goal is to apply techniques from Topological Data Analysis, more precisely, persistent Homology to detect topological properties more persistent from a protein set. For this purpose, we use the weighted α -shapes model of each protein. For constructing the weighted α -shapes of a protein, the three-dimensional coordinates and Van der Waals radii (See Table 5) were extracted of each atom. To obtain the α -shapes filtration, one gradually increases the radius of balls. More precisely, the α -shapes can be constructed by the intersection of balls, and so it gives rise to the birth of alpha complexes. Once we have the alpha complexes filtration, we use persistent homology, specifically, persistence diagrams.

Recall from Section 3.3 that the k -dimensional persistence diagram of filtration $\mathcal{X}$ is a multi-set $PD_k(\mathcal{X})$ ($k \in \mathbb{N} \cup \{0\}$) of pairs of points of the form (b, d) , where each pair correspond to the birth and death values of a given k -dimensional hole γ for which γ appears and disappears in the filtration $\mathcal{X}$. Here, we only selected the persistence diagrams of tunnels β_1 and cavities β_2 of each protein, i. e., $k = 1, 2$, because we consider that they present relevant topological information. To extract a feature vector from the persistence diagram $PD_k(\mathcal{X})$, we fixed $\alpha_{\min} < \alpha_{\max}$ with $\alpha_{\min}, \alpha_{\max} \in \mathbb{R}^+$, and considered only the persistence points whose birth values are in the interval $[\alpha_{\min}, \alpha_{\max}]$ (See Chapter 6). Now considered a uniform interval $\alpha_{\min} = \alpha_0 < \dots < \alpha_m = \alpha_{\max}$ consisting of m subintervals of $[\alpha_{\min}, \alpha_{\max}]$, and let v_i be the number of pairs in the persistence diagram $PD_k(\mathcal{X})$, whose birth value b is in the interval $[\alpha_{j-1}, \alpha_j]$. We define the k -dimensional *persistence feature vector* of size m to be the vector $v_i^k(\mathcal{X}) = (v_{(i)1}^k, v_{(i)2}^k, \dots, v_{(i)m}^k) \in \mathbb{R}^m$ with $k = 1, 2$. Namely, we define the new persistence feature vector $w_i(\mathcal{X}) := (v_i^1(\mathcal{X}), v_i^2(\mathcal{X})) \in \mathbb{R}^{2m}$ with $i = 1, \dots, n$. Last, the *general matrix* $W(\mathcal{X})$ be constructed using these n new persistence feature vectors.

The machine learning methods used in this section are the *Support vector machine* (SVM), *Partial least squares-discriminant analysis* (PLS-DA), and *Naive Bayes* classifier. The performance of a machine learning method on a dataset is measured by the *accuracy*, which measures the overall amount of correct identification from all predictions that were made on the dataset. The entire procedure is summarized in the next pipeline (See Figure 22).

Figure 22 – Pipeline about entire proposed procedure.

Source: Elaborated by the author.

In the following section, we describe the proteins datasets and the procedures that we use to validate the proposed method. We examine the accuracy, F_1 -score, and explore the utility of the proposed method.

7.2 Experiments and Results

In this experiment we have chosen the datasets used in [Cang et al. \(2015\)](#) and publicly available in [Fox, Brenner and Chandonia \(2013\)](#), [Berman et al. \(2000\)](#). We use [Edelsbrunner \(2014\)](#)'s proposal to represent a protein by the space-filling model where each atom (element) is represented by a ball with Van Der Waals radii (See Table 5).

Table 5 – List of Van Der Waals radii (Å) of some chemical elements.

Element	symbol	radius (Å)	Element	symbol	radius(Å)
Hydrogen	H	1.20	Sulfur	S	1.80
Zinc	Zn	1.39	Lithium	Li	1.82
Helium	He	1.40	Arsenic	As	1.85
Copper	Cu	1.40	Bromine	Br	1.85
Fluorine	F	1.47	Uranium	U	1.86
Oxygen	O	1.52	Gallium	Ga	1.87
Neon	Ne	1.54	Argon	Ar	1.88
Nitrogen	N	1.55	Selenium	Se	1.90
Mercury	Hg	1.55	Indium	In	1.93
Cadmium	Cd	1.58	Thallium	Tl	1.96
Nickel	Ni	1.63	Iodine	I	1.98
Palladium	Pd	1.63	Krypton	Kr	2.02
Gold	Au	1.66	Lead	Pb	2.02
Carbon	C	1.70	Tellurium	Te	2.06
Silver	Ag	1.72	Silicon	Si	2.10
Magnesium	Mg	1.73	Xenon	Xe	2.16
Chlorine	Cl	1.75	Tin	Sn	2.17
Platinum	Pt	1.75	Sodium	Na	2.27
Phosphorus	P	1.80	Potassium	K	2.75

Source: Adapted from [Huheey et al. \(2006\)](#).

For our experiments, we used the following proteins datasets:

- a) 19 **proteins**. It consists of 19 samples organized into 2 classes, as follow: 9 structures of hemoglobin in *Relaxed form* denoted as *R-form*, and 10 structures of hemoglobin in *Taut form* denoted as *T-form* (See Table 6).

Table 6 – Protein molecules used for the Hemoglobin classification.

R-form	1aj9, 1hbr, 1hho, 1ibe, 1lfq, 1rvw, 2d5x, 2w6v, 3a0g
T-form	1gzx, 1lfl, 1kd2, 1olj, 2d5z, 2dhb, 2dxm, 2hbs, 2hhb, 4rol

Source: [Cang et al. \(2015\)](#).

- b) 900 **proteins**. This dataset contains 900 samples organized into 3 classes: the *Alpha* class formed by 300 proteins and denoted as G_1 , the *Beta* class formed by 300 and denoted as G_2 , and the mixed *Alpha and Beta* class formed by 300 proteins and denoted as G_3 (See Appendix in [Cang et al. \(2015\)](#)).

Our implementation was done in *Matlab* and *R* software. Specifically, the construction of the filtration of alpha complexes was done in *Matlab*, for the computation of persistent homology

was used *PHAT*, and for the classification of the dataset was used *R* software.

7.2.1 Classifiers evaluation

In the case of 19 proteins dataset, we have two groups that we denote by *R-form* and *T-form*, each one consisting of 9 and 10 samples of size $2m$, respectively. For each run, we randomly selected two samples of the dataset to be the test set (one sample of each class) and the remaining for the training set. Now, in the case of 900 proteins dataset, we have three groups that we denote by G_1 , G_2 , and G_3 . Each group consist of 300 features vectors of size $2m$.

For each value of m we apply the methods SVM, PLS-DA, and Naive Bayes to classify the dataset. For each run, we randomly selected 80% of the dataset as the training set and the remaining 20% as the test set. For both datasets, we run each computation 30 times and computed the average accuracy among these 30 computations. We varied the m parameter for evaluating the performance of the used classifiers.

For the 19 proteins dataset, Figure 23 shows the plot of average accuracy as a function of m . We observe that the SVM and Naive Bayes method obtained the best result. On the other hand, the PLS-DA method presented more oscillations and obtained the lowest result because this classifier had problems of overfitting or misleading classification results due to the lower number of samples in relation to the number of features (attributes).

Figure 23 – Average accuracy values according to m for SVM, PLS-DA, and Naive Bayes classifiers for the 19 proteins dataset (*R-form* and *T-form*).

Source: Research data.

For the 900 proteins dataset, we observe in Figure 24(a) that the growth of the accuracy curves are more stable, and with minimum oscillations in the SVM classifier. In general, the combinations of G_2 and G_3 (red), and G_1 and G_2 (blue) curves produce better results. On the other hand, the Naive Bayes method presented more oscillations and obtained the lowest results. Last, we observe that the SVM classifier obtained the best results.

Figure 24 – Average accuracy values according to m for (a) SVM, (b) PLS-DA, and (c) Naive Bayes classifiers for the 900 proteins dataset.

Source: Research data.

7.2.2 Visualization of classifiers for the 900 proteins

For all the classifiers the chosen parameter is $m = 100$. The best results of the classification are reported through the validate table. For PLS-DA classifier we visualize its behavior using the projections onto their PLS components C_i , with $i = 1 : 3$ (See Subsection 5.1.3). For SVM and Naive Bayes classifiers we used the principal components $\mathbf{pc}_i$, with $i = 1 : 3$ (WOLD; ESBENSEN; GELADI, 1987; ABDI; WILLIAMS, 2010). In addition, the confusion matrix show the number of correct and incorrect predictions made by the classifiers compared to the actual.

The methods PLS-DA, SVM, and Naive Bayes are used to classify all possible pairs G_i and G_j , and also the three groups G_1 , G_2 , and G_3 . Figures 25, 26, 27 (left side) show the projections. Comparing classifiers, we observe in Figures 25 (a), (c), (e), (g), that there is an excellent separation of classes when using the PLS-DA classifier. However, we observe in Figures 26, 27, that there is a small number of incorrectly clustered instances for the three groups, although that there is a clearer distinction between pairs combinations G_i and G_j .

The confusion matrix for the PLS-DA, SVM, and Naive Bayes classifier are presented in Figures 25, 26, and 27 (right side). For instance, we observed in Figure 25 (b) that only one protein of G_3 is classified as belonging to G_1 . Also in Figure 26 (d), one protein of the G_1 is classified as belonging to G_2 . In Figure 27 (f), two proteins of G_3 is classified as belonging to G_1 .

Figure 25 – Projections and the validation tables of the proteins classification of G_1 (black), G_2 (green), and G_3 (red) group using PLS-DA classifier. (a), (b), G_1 , G_2 , and G_3 ; (c), (d), G_1 and G_2 ; (e), (f), G_1 and G_3 ; (g), (h), G_2 and G_3 . (a), (c), (e), and (g) are projections, the remaining are the respective confusion matrix.

Source: Research data.

Figure 26 – Projections and the validation tables of the proteins classification of G_1 (black), G_2 (green), and G_3 (red) group using SVM classifier. (a), (b), G_1 , G_2 , and G_3 ; (c), (d), G_1 and G_2 ; (e), (f), G_1 and G_3 ; (g), (h), G_2 and G_3 . (a), (c), (e), and (g) are projections, the remaining are the respective confusion matrix.

Source: Research data.

Figure 27 – Projections and the validation tables of the proteins classification of G_1 (black), G_2 (green), and G_3 (red) group using Naive Bayes classifier. (a), (b), G_1 , G_2 , and G_3 ; (c), (d), G_1 and G_2 ; (e), (f), G_1 and G_3 ; (g), (h), G_2 and G_3 . (a), (c), (e), and (g) are projections, the remaining are the respective confusion matrix.

Source: Research data.

7.2.3 Comparing classifiers

Now we compared the performance of the three adopted classifiers taking into account the accuracy and F_1 -score indices, obtaining a comparative overview of our proposed approach.

For instance, in Figure 28 we present the performance of the classifiers as a function of m and the average F_1 -scores. We observe that the classifiers achieved higher values of F_1 -scores (above 0.7), which was similar with the accuracy. This result indicates that the proposed method considering the persistence diagram information, for proteins classification leads to very precise and robust results.

Figure 28 – Classifiers comparison of the F_1 -score performance in function of m , for the 900 **proteins** in the cases: (a) G_1 and G_2 ; (b) G_2 and G_3 ; (c) G_1 and G_3 ; (d) G_1 , G_2 , and G_3 ; and (e) the 19 **proteins** in the case: R -form and T -form.

Source: Research data.

Furthermore, we can observe in Figure 28 that the topological information and separation of the classes benefited the classification process regardless of the classifier vies (probabilistic, optimization or dimensional regression). In the 900 **proteins** results (See Figures 28 (a),(b),(c), and (d)), the SVM, and PLS-DA classifiers achieved very similar F_1 -scores, with insignificant differences from $m \geq 50$. On the other hand, the Naive Bayes classifier achieved the lowest F_1 -scores values in all the cases, especially in the regime of $10 \leq m \leq 50$. In the case of the 19 **proteins** dataset (See Figure 28 (e)), we observe that the variation of m perturbs the F_1 -scores achieved by the classifiers. In particular, the SVM and Naive Bayes classifiers achieved the best results in most of the m values. Notice that the F_1 -score results for the PLS-DA were on average equal to 0.7. The before indicates that even that the PLS-DA obtained low accuracy values, the precision of the classifier is higher with a higher ratio of correctly predicted proteins to all samples in the actual class. This means, lower False Positives predictions, which are usually considered more critical and crucial than False Negatives.

Now we highlight the performance of the best results reached for some values of the m parameter of each classifier. In the case of 900 **proteins** dataset, Tables 7, 8, and 9 present the best classification results of all possible group of protein separation problems, i.e., (G_i, G_j) and the three groups (G_1, G_2, G_3) , using the SVM, PLS-DA, and Naive Bayes classifiers. Notice that in terms of ranking results, SVM was the best classifier in three of four protein separation problems. Although, PLS-DA obtained one first place (in G_2, G_3 Groups) according to its accuracy and F_1 -score results. Moreover, in the 19 **proteins** results, Table 10 shows the best classification results, using the SVM, PLS-DA, and Naive Bayes classifier. Observe that Naive Bayes tied in first place, along with SVM.

Table 7 – Comparative results for the performance of SVM classifier in the case of 900 proteins dataset

Groups	Measure Parameter	Average accuracy	Average F_1 -score
G_1, G_2	$m = 69$	0.995556	0.995526
G_1, G_3	$m = 55$	0.995833	0.995812
G_2, G_3	$m = 94$	0.993611	0.993524
G_1, G_2, G_3	$m = 89$	0.988703	0.988879

Source: Research data.

Table 8 – Comparative results for the performance of PLS-DA classifier in the case of 900 proteins dataset.

Groups	Measure Parameter	Average accuracy	Average F_1 -score
G_1, G_2	m = 83	0.994444	0.994401
G_1, G_3	m = 95	0.992500	0.992511
G_2, G_3	m = 75	0.995833	0.995839
G_1, G_2, G_3	m = 56	0.984259	0.984508

Source: Research data.

Table 9 – Comparative results for the performance of Naive Bayes classifier in the case of 900 proteins dataset.

Groups	Measure Parameter	Average accuracy	Average F_1 -score
G_1, G_2	m = 96	0.981944	0.982360
G_1, G_3	m = 92	0.976388	0.976902
G_2, G_3	m = 72	0.995556	0.995489
G_1, G_2, G_3	m = 91	0.961667	0.962598

Source: Research data.

Table 10 – Comparative results for the performance of classifiers in the case of 19 proteins dataset.

Classifiers	Measure Parameter	Average accuracy	Average F_1 -score
SVM	m = 24	1.000000	1.000000
PLS-DA	m = 44	0.916666	0.944444
Naive Bayes	m = 11	1.000000	1.000000

Source: Research data.

7.2.4 Discussion

The resulting of our method was validated in two datasets cited in [Cang et al. \(2015\)](#). First, we explored the performance of our method for distinguishing three classes of proteins in 900 samples. Using SVM classifier, we found an average accuracy of 98.87% (See Table 11) for three protein classes. In comparison with the MTF-SVM method of [Cang et al. \(2015\)](#), our proposed method does an excellent job of classifying this dataset.

Table 11 – CV classification rates (%) of SVM with MTF-SVM (cited from [Cang et al. \(2015\)](#)) and our method.

Classification rates		900 proteins dataset
Method		
MTF-SVM		84.93
Our method		98.83

Source: [Cang et al. \(2015\)](#).

In our last test, the discrimination of hemoglobin molecules in their relaxed and taut forms was considered for comparing our method with the MTF-SVM method of [Cang et al. \(2015\)](#), and the PWGK-RKHS method proposed by [Kusano, Fukumizu and Hiraoka \(2016\)](#), [Kusano, Fukumizu and Hiraoka \(2017\)](#). Again, our method works very well with an average accuracy of 100% (See Table 12) using SVM classifier.

Table 12 – CV classification rates (%) of SVM with MTF-SVM, PWGK-RKHS (cited from [Cang et al. \(2015\)](#), [Kusano, Fukumizu and Hiraoka \(2017\)](#), [Kusano, Fukumizu and Hiraoka \(2016\)](#)), and our method.

Classification rates		19 proteins dataset
Method		
MTF-SVM		84.50
PWGK-RKHS		88.90
Our method		100

Source: [Cang et al. \(2015\)](#).

Finally, using the SVM classifier, we can see that our method achieves better performance than the results of [Cang et al. \(2015\)](#), [Kusano, Fukumizu and Hiraoka \(2016\)](#), [Kusano, Fukumizu and Hiraoka \(2017\)](#). The detailed comparisons were verified experimentally in Subsection 7.2.3.

7.3 Conclusions and Future Works

We proposed a method for proteins classification based on the topology of data (persistence diagrams) and combined it with machine learning. As a significant advantage, we conclude that our method worked very well because it got a better accuracy and higher values of F_1 -scores in comparison with other works. In addition, the results indicate that our method was very precise and robust.

For future work, we plan to apply the method to other datasets, including experimental data where the method has the potential of being very useful to match parameters of experimental and simulated data.

PARAMETER IDENTIFICATION IN A PREDATOR-PREY SYSTEM

Mathematical models, in particular, differential equations, are extensively used to study problems in sciences and engineering, and hence, it is extremely important to develop methods to design and analyze such models. Designing mathematical equations to model problems in sciences and engineering is referred to as *mathematical modeling* (OBERKAMPF; ROY, 2010). When modeling a problem, it is necessary to decide what type of model to use, i.e., whether to use differential equations or statistical models for example, and then to choose the appropriate equations to describe the problem. Once a suitable model has been chosen, it is necessary to ensure that it is solved correctly and that it solves the correct problem. In scientific computing, this step is referred to as *verification* and *validation* of the model (OBERKAMPF; ROY, 2010; CUESTA; ABREU; ALVEAR, 2015). More specifically, *verification* can be defined as the process of ensuring that the model is correctly implemented and the solution is accurate (“solving the equations right”) and *validation* can be defined as the process of determining that the model provides an accurate description of the problem it is intended to describe (“solving the right equations”). The validation process often involves comparing the results from the model to experimental data (IVES *et al.*, 2008; OBERKAMPF; ROY, 2010).

Mathematical models often include parameters that need to be determined during the validation process. The process of determining the model parameters, called *parameter identification*, is a very important and challenging problem is often addressed by comparing the solutions of the model to experimental data (KRISHAN *et al.*, 2007; IVES *et al.*, 2008; SARGENT, 2013; XUN *et al.*, 2013).

In this chapter, we propose to apply techniques from *Topological Data Analysis* (TDA) (CARLSSON, 2009), more specifically *Persistent Homology* (PH) (EDELSBRUNNER, 2014; GHRIST, 2008), combined with *Machine Learning* (ML) (BISHOP, 2006; MITCHELL, 1997; ROGERS; GIROLAMI, 2016) to study the parameter identification problem in models pro-

ducing complex spatio-temporal patterns (GARVIE, 2007; HEARST, 1998; IVES *et al.*, 2008; KACZYNSKI; MISCHAIKOW; MROZEK, 2006). More precisely, we compute persistent homology of the level sets of the patterns produced by the system and use the corresponding *Persistence Diagrams* (PD) as features for machine learning algorithms.

8.1 Persistent Homology of Level Sets

Persistent homology is a tool that provides metric information about the topological properties of an object and how robust these properties are with respect to change in parameters. More precisely, persistent homology counts the number of connected components and holes of various dimensions and keeps track of how they change with parameters.

Given a space (object) X that varies as a function of a parameter. Persistent homology provides a way of capturing how the shape of this object changes as we vary this parameter. To make this more precise, we need to describe the type of spaces X we will consider and how X changes with the parameter (KACZYNSKI; MISCHAIKOW; MROZEK, 2006).

Let $h \in \mathbb{R}$ be a fixed grid size. Given $j \in \mathbb{Z}$ we denote by $I_j = [jh, (j+1)h]$ the interval of length h with end-points jh and $(j+1)h$. An n -dimensional cube (or a cube of dimension n) is a set of the form $I_{j_1} \times I_{j_2} \times \cdots \times I_{j_n}$, where $j_1, j_2, \dots, j_n \in \mathbb{Z}$. An n -dimensional cubical complex is a finite collection X of n -dimensional cubes.

Observation 1. We can also choose different values of h in each dimension.

To a cubical complex X , we associate a collection of groups $H_k(X)$ ($k = 0, 1, \dots$) called *homology groups* of X , that provide the essential topological features of X . For the type of complexes that we consider in this section, the homology groups are of the form $H_k(X) = \mathbb{R}^{\beta_k}$, $k = 0, 1, \dots$, where β_k is a non-negative integer called the k -th *Betti number* of X . More specifically, for the cubical complexes that we consider in this section, the homology groups are in fact vector spaces and the Betti numbers are the dimensions of these vector spaces. In this section we consider only 2-dimensional cubical complexes, hence we are concerned only with the number of components β_0 and the number of holes β_1 .

Given a finite collection of n -dimensional cubical complexes $X^1 \subset X^2 \subset \cdots \subset X^r$, with $r = 1, 2, \dots$, *persistent homology* provides information about the changes in the Betti numbers as we move from one cubical complex X^j to the next one X^{j+1} , $1 \leq j < r$. The collection of cubical complexes X^i ($i = 0, 1, \dots$) is called a *filtration* and we denote it by $\mathcal{X}$. More precisely, the *persistent homology* $PH_k(\mathcal{X})$ ($k = 0, 1, \dots$) of $\mathcal{X}$ is characterized by its *persistence diagrams* $PD_k(\mathcal{X})$, $k = 0, 1, \dots$, where each $PD_k(\mathcal{X})$ is a multi-set of pairs of points of the form (b, d) called *birth-death pairs*. Each point $(b, d) \in PD_k(\mathcal{X})$ represents an k -dimensional hole γ in $\mathcal{X}$ (EDELSBRUNNER, 2014; OTTER *et al.*, 2017; MISCHAIKOW; NANDA, 2013).

Example 3 (Persistence diagrams). Consider the increasing sequence of cubical complexes called cubical filtration, $X^1 \subset X^2 \subset \dots \subset X^6$, as we can see in Figure 29. In this example, we have four connected components in X^1 , three in X^2 , two in X^3 , and one component in X^4 to X^6 . Also, there are zero holes in X^1 to X^3 , one hole in X^4 , three holes in X^5 , and zero holes in X^6 . These features are captured by the persistence diagrams $PD_0 = \{(1,2), (1,3), (1,4), (1,+\infty)\}$ and $PD_1 = \{(4,6), (5,6), (5,6)\}$ shown in Figure 30.

Figure 29 – Filtration of cubical complexes $X^1 \subset X^2 \subset \dots \subset X^6$, and their Betti numbers β_0 and β_1 .

Source: Elaborated by the author.

Figure 30 – Persistence diagrams PD_0 (left) and PD_1 (right) of the filtration in Figure 29. Notice that the fact that the point $(5,6)$ appears twice in PD_1 is not visible in the plot.

Source: Elaborated by the author.

Given a function $u: \Omega \rightarrow \mathbb{R}$ defined on a rectangle $\Omega := [a, b] \times [c, d]$, we can construct a cubical complex filtration by making a grid on the domain Ω and considering the sub-level sets of u given by

$$U_r := \{(x, y) \in \Omega \mid u(x, y) \leq r\},$$

with threshold $r \in \mathbb{R}^+$.

We then define the cubical complex X^r to be the set of grid elements that intersect U_r . Since there is only a finite number of grid elements, we get a finite filtration of cubical complexes $X^{r_0} \subset X^{r_1} \subset \dots \subset X^{r_N}$, with $r_0 < r_1 < \dots < r_N$, $r_i \in \mathbb{R}^+$, $i = 0, \dots, N$, where $X^{r_0} = \emptyset$ and X^{r_N} is the full cubical grid. Using this filtration we can compute the persistent homology of the sub-level sets of u . Figure 32 presents an example of some level sets and the persistence diagrams of the corresponding filtration.

Let consider the following reaction-diffusion predator-prey system (8.1) defined on a rectangular domain Ω with no-flux (Neumann) boundary conditions (GARVIE, 2007; HOLLING, 1965; WARBURTON, 1974; MURRAY, 2002).

$$\begin{cases} \frac{\partial u}{\partial t} = \Delta u + u(1 - u) - \frac{uv}{\alpha + u} \\ \frac{\partial v}{\partial t} = \delta \Delta v + \beta \frac{uv}{\alpha + u} - \gamma v \end{cases} \quad (8.1)$$

Here $u(x, y, t)$ and $v(x, y, t)$ represent the population densities of prey and predators, respectively, at time t and vector position (x, y) , Δ is the usual Laplacian operator in $d \leq 3$ space dimensions, and the parameters α, δ, β , and γ are strictly positive. The choice of boundary conditions is equivalent to the assumption that both species cannot leave the domain.

Solving the predator-prey system (8.1) numerically on a uniform grid in space and time using a semi-implicit (in time) finite-differences method given in Garvie (2007). The initial approximations $u(x, y, 0)$ and $v(x, y, 0)$ to the solutions u and v of the system (8.1) in two-dimensions are given by

$$\begin{aligned} u_0(x, y) &= 6/35 - 2 \times 10^{-7}(x - 0.1y - 225)(x - 0.1y - 675), \\ v_0(x, y) &= 116/245 - 3 \times 10^{-5}(x - 450) - 1.2 \times 10^{-4}(y - 150), \end{aligned}$$

respectively.

We denote the grid sizes in space by h and in time by Δt . For our experiments we fix the domain size and the parameter values as follows: $\Omega = [0, 400] \times [0, 400]$, $h = 1$, $\Delta t = 1/3$, $\alpha = 0.4$, $\gamma = 0.6$, $\delta = 1$, and vary the parameter β . Figure 31 shows some level sets of the solutions $u(x, y, t)$ of the system (8.1) for different values of the parameter β .

Figure 32 (first row) presents some cubical complexes on the filtration of level sets of one of the solutions on the top-right corner of Figure 31. Also, Figure 32 (second row) shows the persistence diagrams of the corresponding filtration to the connected components β_0 (bottom-left) and the cycles β_1 (bottom-right).

Figure 31 – Level sets of solutions $u(x,y,t)$ of the predator-prey system (8.1). The solution on the first row correspond the $\beta = 2.0$, on the second row to $\beta = 2.1$, and on the third row to $\beta = 2.2$. The solutions on the first column correspond to $t = 100$, and the second column to $t = 200$, and on the third column to $t = 300$.

Source: Research data.

Figure 32 – Some complexes on the filtration of the level sets of the solution corresponding to $\beta = 2.0$ on Figure 31 (top) and the corresponding persistence diagrams (bottom).

Source: Research data.

In the next section, we describe how to use persistent homology of level sets to extract features to be used in the machine learning methods (classifiers).

8.2 Proposed Method

Our goal is to apply machine learning to identify parameters of solutions in the reaction-diffusion predator-prey system (8.1). More specifically, we use the persistence diagrams of the level sets of the solutions of (8.1) to extract features from these solutions and apply machine learning to these features (CALCINA SABRINA S., 2018).

Recall from Section 8.1 that the k -dimensional persistence diagram of a level set filtration $\mathcal{X}$ is a multi-set $PD_k(\mathcal{X})$ ($k = 0, 1, \dots$) of pairs of points of the form (b, d) , where each pair correspond to the birth and death values of a given k -dimensional hole γ in terms of the level set values (threshold) r with $r \in \mathbb{R}^+$ for which γ appears and disappears in the filtration $\mathcal{X}$, compute its persistence diagram using the Perseus software (See Section 4.2). Here, we only selected the persistence diagrams of connected components β_0 and tunnels β_1 of a level set filtration, i. e., $k = 0, 1$. To extract a feature vector from the persistence diagram $PD_k(\mathcal{X})$, we fix $r_{\min} < r_{\max}$ with $r_{\min}, r_{\max} \in \mathbb{R}^+$, and consider only the persistence points whose birth values are in the interval $[r_{\min}, r_{\max}]$ (see Figure 33). Now consider a uniform grid $r_{\min} = r_0 < r_1 < \dots < r_m = r_{\max}$ consisting of m ($m \in \mathbb{N}$) subintervals of $[r_{\min}, r_{\max}]$, and let v_j ($j = 1, \dots, m$) be the number of pairs in the persistence diagram $PD_k(\mathcal{X})$ with $k = 0, 1$, whose birth value b is in the interval $[r_{j-1}, r_j)$ with $j = 1, \dots, m$. We define the k -dimensional *persistence feature vector* of size m to be the vector $v^k(\mathcal{X}) = (v_1, v_2, \dots, v_m) \in \mathbb{R}^m$, $k = 0, 1$. Then, we concatenated these vectors and define the *persistence feature vector* $w(\mathcal{X}) \in \mathbb{R}^{2m}$. Finally, we constructed the *general matrix* $W(\mathcal{X})$ using this new persistence feature vectors (for more details see Chapter 6). Figure 33 presents the procedure for the extraction of a feature vector.

Figure 33 – Pipeline about the procedure for feature vector extraction.

Source: Elaborated by the author.

In the next section, we describe the datasets that we use to classify the solutions according to their parameter values by applying machine learning methods (classifiers) to the persistence feature vectors.

8.3 Experiments and Results

As described in Section 8.1 we solve the predator-prey system (8.1) numerically on the domain $\Omega = [0, 400] \times [0, 400]$ with spatial grid size $h = 1$, time step $\Delta t = 1/3$, and the following parameters values: $\alpha = 0.4$, $\gamma = 0.6$, and $\delta = 1$. For the computations in this work, we consider three values of the parameter β , namely, $\beta_1 = 2.0$, $\beta_2 = 2.1$, and $\beta_3 = 2.2$. For each value of the parameter β , we solve system (8.1) up to $t = 300$, and consider the solutions $u(x, y, t)$ for t varying from $t = 100$ to $t = 300$ to form our dataset. Hence, we have three groups of solutions corresponding to β_1 , β_2 , and β_3 that we denote by S_1 , S_2 , and S_3 , respectively. Since $\Delta t = 1/3$, each group consists of 600 solutions of (8.1).

Now as described in Section 8.2, given values of $r_{min}, r_{max} \in \mathbb{R}^+$ and $m \in \mathbb{N}$, for each solution $u(x, y, t)$ in the groups S_1 , S_2 , and S_3 , we construct a level set filtration $\mathcal{X}$, compute its persistence and constructed the 0-dimensional and the 1-dimensional persistence feature vectors $v^0(\mathcal{X})$ and $v^1(\mathcal{X})$. Last, we concatenated these two vectors and define the *persistence feature vector* $w(\mathcal{X}) := (v^0(\mathcal{X}), v^1(\mathcal{X})) \in \mathbb{R}^{2m}$. Therefore, we have three groups of feature vectors, that we denote by $P_1(m)$, $P_2(m)$, and $P_3(m)$, each one consisting of 600 feature vectors of size $2m$. In total, the **dataset** consists of 1800 samples organized into 3 classes: the P_1 class formed by 600 samples, the P_2 class formed by 600 samples, and the P_3 class formed by 600 samples.

We fix the values of $r_{min} = 0$ and $r_{max} = 0.792$ for the 0-dimensional and the 1-dimensional persistence diagrams, and compute the groups $P_1(m)$, $P_2(m)$, and $P_3(m)$, for several values of m . For each value of m we apply the methods SVM, PLS-DA, and Naive Bayes to classify all possible pairs $P_i(m)$ and $P_j(m)$ ($i \neq j$ and $i, j \in \{1, 2, 3\}$), and also to classify the three groups $P_1(m)$, $P_2(m)$, and $P_3(m)$. For each run, we randomly selected 80% of the dataset as the training set and the remaining 20% as the test set. We run each computation 30 times and computed the average accuracy among these 30 computations.

Figure 34 shows the plots of the average accuracy as a function of m . As we can see from these results, the classification is successful in all the cases. Hence, the method is effective in identifying the parameter values corresponding to each group.

Figure 34 – Average accuracy values versus the parameter m for (a) SVM, (b) PLS-DA, and (c) Naive Bayes classifiers.

Source: Research data.

8.3.1 Comparing classifiers

We compared the performance of the three used classifiers taking into account the accuracy and F_1 -score indices, obtaining a comparative overview of our proposed approach. In Figure 35, for example, we present the performance of the classifiers according to m parameter and the average F_1 -scores. We observe that the classifiers achieved higher values of F_1 -scores (above 0.998), which was similar with the accuracy. This result indicates that the proposed method considering the persistence diagram information, for classify the three groups (P_1 , P_2 , and P_3) leads to very precise and robust results.

Now we present the performance of the best results achieved in most of the m values of each classifier. Table 13 presents the best classification results of all possible group combinations, i.e., (P_i and P_j , with $i \neq j$ and $i, j \in \{1, 2, 3\}$) and the three groups (P_1 , P_2 , and P_3), using the SVM, PLS-DA, and Naive Bayes classifiers. Notice that in terms of ranking results, the three were good classifiers in all separation problems. Although, SVM obtained one first place according to its accuracy and F_1 -score results how we can observe in the curves of the Figures 34 and 35.

Figure 35 – Classifiers comparison of the F_1 -score performance in function of m , for (a) P_1 and P_2 ; (b) P_2 and P_3 ; (c) P_1 and P_3 ; (d) P_1 , P_2 , and P_3 groups.

Source: Research data.

Table 13 – Comparative results for the performance of SVM, PLS-DA, and Naive Bayes classifier.

Classifiers	SVM		PLS-DA		Naive Bayes	
	Average Accuracy	Average F_1 -score	Average Accuracy	Average F_1 -score	Average Accuracy	Average F_1 -score
P_1, P_2	1.0	1.0	1.0	1.0	1.0	1.0
P_1, P_3	1.0	1.0	1.0	1.0	1.0	1.0
P_2, P_3	1.0	1.0	1.0	1.0	1.0	1.0
P_1, P_2, P_3	1.0	1.0	1.0	1.0	1.0	1.0

Source: Research data.

8.4 Conclusions and Future Works

We use persistent homology as a feature extractor for machine learning methods (classifiers) to identify parameters in systems of differential equations exhibiting complex spatio-temporal patterns. The method is applied to the patterns generated by a system of differential equations, hence it can be applied directly to experimental (image) data where we have only images representing the state of a system such as experiments in fluid dynamics for example. The method presents excellent results on the dataset considered in our experiment.

For future works, we plan to apply the method to other datasets, including experimental data where the method has the potential of being very useful to match parameters of experimental and simulated data.

PARAMETER ESTIMATION IN SYSTEMS EXHIBITING SPATIALLY COMPLEX SOLUTIONS

Differential equations and other types of mathematical models are extensively used to model problems in sciences and engineering. One key step in the development of a *mathematical model* (OBERKAMPF; ROY, 2010) to describe a problem is to ensure that one has the right equations and that they are being solved correctly. This step is referred to as *model verification and validation* in scientific computing (CUESTA; ABREU; ALVEAR, 2015; OBERKAMPF; ROY, 2010), where *verification* is the process by which one ensures that the model is implemented (solved) correctly and that the solution is accurate, and *validation* is the process of determining if the model provides an accurate description of the problem. This last step often involves comparing the results of the model with experimental data (IVES *et al.*, 2008; OBERKAMPF; ROY, 2010) and determining the correct parameter for the model (IVES *et al.*, 2008; KRISHAN *et al.*, 2007; SARGENT, 2013; XUN *et al.*, 2013).

In this chapter, we propose to apply techniques from *Topological Data Analysis* (TDA) (CARLSSON, 2009), more precisely *Persistent Homology* (PH) (EDELSBRUNNER, 2014; GHRIST, 2008; WEINBERGER, 2011), combined with *Machine Learning Regression* models (BISHOP, 2006; SMOLA; SCHÖLKOPF, 2004; VAPNIK; GOLOWICH; SMOLA, 1996) to estimate the parameters of models producing complex spatio-temporal patterns (GARVIE, 2007; GAMEIRO; MISCHAIKOW; KALIES, 2004; GAMEIRO; MISCHAIKOW; WANNER, 2005). More specifically, we apply machine learning regression models to a vectorization of the *Persistence Diagrams* (PD) of the patterns. In this sense, our goal is to use persistent homology of level sets to estimate parameters in systems producing complicated spatio-temporal patterns.

9.1 Persistent Homology of Level Sets

In this section we present the compute of persistent homology of level sets. For a more details please see Chapter 8 (Section 8.1).

Given a function $u: \Omega \rightarrow \mathbb{R}$ defined on a domain $\Omega := [a, b] \times [c, d]$, we can construct a cubical complex filtration by making a grid on the domain Ω and considering the sub-level sets of u given by

$$U_r := \{(x, y) \in \Omega \mid u(x, y) \leq r\},$$

with threshold $r \in \mathbb{R}$. We then define the cubical complex X^r to be the set of grid elements that intersect U_r . Since there is only a finite number of grid elements, we get a finite filtration of cubical sets $X^{r_0} \subset X^{r_1} \subset \dots \subset X^{r_N}$, with $r_0 < r_1 < \dots < r_N$. So $X^{r_0} = \emptyset$ and X^{r_N} is the full cubical grid. Using this filtration we can compute the persistent homology of the sub-level sets of u . An example of some level sets and the persistence diagrams of the corresponding filtration is presented in Figure 38.

In the next subsections, we describe the datasets that we use to estimate their parameters values by applying machine learning methods (regression) to the persistence feature vectors.

9.1.1 Predator-Prey System

Our goal is to use persistent homology of level sets to estimate parameters in systems producing complicated spatio-temporal patterns. For this purpose consider the following reaction-diffusion predator-prey system (9.1) defined on a rectangular domain Ω with no-flux (Neumann) boundary conditions (GARVIE, 2007; HOLLING, 1965; MURRAY, 2002; Warburton, 1974).

$$\begin{cases} \frac{\partial u}{\partial t} = \Delta u + u(1 - u) - \frac{uv}{\alpha + u} \\ \frac{\partial v}{\partial t} = \delta \Delta v + \beta \frac{uv}{\alpha + u} - \gamma v \end{cases} \quad (9.1)$$

Here, $u(x, y, t)$ and $v(x, y, t)$ represent the population densities of prey and predators, respectively, at time t and vector position (x, y) , Δ is the usual Laplacian operator, and the parameters α, δ, β , and γ are strictly positive. The choice of boundary conditions is equivalent to the assumption that both species cannot leave the domain.

Solving the predator-prey system (9.1) numerically on a uniform grid in space and time using a semi-implicit (in time) finite-differences method given in Garvie (2007). The initial approximations $u(x, y, 0)$ and $v(x, y, 0)$ to the solutions of u and v of the system (9.1) in two-dimensions are given by

$$\begin{aligned} u_0(x, y) &= 6/35 - 2 \times 10^{-7}(x - 180)(x - 720) - 6 \times 10^{-7}(y - 90)(y - 210), \\ v_0(x, y) &= 116/245 - 3 \times 10^{-5}(x - 450) - 6 \times 10^{-5}(y - 135), \end{aligned}$$

respectively.

In Figures 36 and 37, we present some level sets of the solutions $u(x, y, t)$ of the predator-prey system (9.1) in the domain $\Omega = [0, 400] \times [0, 400]$ for parameter values: $h = 1$, $\Delta t = 1/3$, $\alpha = 0.4$, $\gamma = 0.6$, and $\delta = 1$, and several values of the parameter β .

Figure 36 – Level sets of solutions $u(x, y, t)$ of the predator-prey system (9.1). The solution on the first row correspond the $\beta = 1.75$, on the second row to $\beta = 1.8$, on the third row to $\beta = 1.85$, on the fourth row to $\beta = 1.9$, on the fifth row to $\beta = 1.95$. The solutions on the first column correspond to $t = 301$, and the second column to $t = 350$, and on the third column to $t = 400$.

Source: Research data.

Figure 37 – Level sets of solutions $u(x, y, t)$ of the predator-prey system (9.1). The solution on the first row correspond the $\beta = 2.0$, on the second row to $\beta = 2.05$, on the third row to $\beta = 2.1$, on the fourth row to $\beta = 2.15$, on the fifth row to $\beta = 2.2$. The solutions on the first column correspond to $t = 301$, and the second column to $t = 350$, and on the third column to $t = 400$.

Source: Research data.

Figure 38 (first row) presents some cubical complexes on the filtration of the level sets of one of the solutions on the first column-bottom of Figure 36. Also, Figure 38 (second row) presents the persistence diagrams of the corresponding filtration to the connected components β_0 (bottom-left) and the cycles β_1 (bottom-right).

Figure 38 – Some complexes on the filtration of the level sets of the solution corresponding to $\beta = 1.95$ on Figure 36 (the first column-bottom) and the corresponding persistence diagrams (bottom).

Source: Research data.

9.1.2 Ginzburg-Landau

Consider the complex Ginzburg-Landau equation ([KURAMOTO, 2012](#))

$$\frac{\partial u}{\partial t} = \Delta u + u - (1 + \beta i)u|u|^2, \quad (9.2)$$

with periodic boundary condition on a two-dimensional domain Ω . Solving Equation (9.2) numerically on the domain $\Omega = [0, 200] \times [0, 200]$ with time-step $\Delta t = 1/2$ and consider the solutions from $t = 100$ to $t = 300$ for the following values of the parameter: $\beta = 1$, $\beta = 1.2$, $\beta = 1.4$, $\beta = 1.6$, and $\beta = 1.8$.

The initial condition $u(x, y, 0)$ for the solution of Equation (9.2) in two-dimensions is given by a random initial condition with amplitude 0.1.

In Figure 39, we show some level sets of the solutions $u(x, y, t)$ of the Ginzburg-Landau Equation (9.2) for the parameter value $\Delta t = 1/2$, and several values of the parameter β .

Figure 39 – Level sets of solutions $u(x, y, t)$ of the Ginzburg-Landau Equation (9.2). The solution on the first row correspond the $\beta = 1.0$, on the second row to $\beta = 1.2$, on the third row to $\beta = 1.4$, on the fourth row to $\beta = 1.6$, on the fifth row to $\beta = 1.8$. The solutions on the first column correspond to $t = 100$, and the second column to $t = 200$, and on the third column to $t = 300$.

Source: Research data.

Figure 40 (first row) shows some cubical complexes on the filtration of level sets of one of the solutions on the third column-top of Figure 39. Also, Figure 40 (second row) presents the corresponding persistence diagrams of the connected components β_0 (bottom-left) and the cycles β_1 (bottom-right).

Figure 40 – Some complexes on the filtration of the level sets of the solution corresponding to $\beta = 1.0$ on Figure 39 (the third column-top) and the corresponding persistence diagrams (bottom).

Source: Research data.

9.2 Proposed Method

Our goal is to apply machine learning to estimate the parameters of models producing complex spatio-temporal patterns. More specifically, using the persistence diagrams of the level sets of the solutions to extract features from these solutions and apply machine learning to these features.

Recall from Chapter 8 (Section 8.2), we must select the persistence diagram of connected components β_0 and cycles β_1 of a level set filtrations $\mathcal{X}$. To extract a feature vector from persistence diagram $PD_k(\mathcal{X})$, we fix $r_{min} < r_{max}$ with $r_{min}, r_{max} \in \mathbb{R}$, and consider only the persistent points whose birth are in the interval $[r_{min}, r_{max}]$. Now consider a uniform grid $r_{min} = r_0 < r_1 < \dots < r_m = r_{max}$ consisting of m ($m \in \mathbb{N}$) subintervals of $[r_{min}, r_{max}]$. Let v_j be the number of pairs in the persistence diagram whose birth value b is in the interval $[r_{j-1}, r_j]$. We then define the k -dimensional persistence feature vector of size m to be the vector $v_k(\mathcal{X}) = (v_1, \dots, v_m) \in \mathbb{R}^m$. Last, the general matrix $W(\mathcal{X})$ is constructed using the new persistence feature vector $w(\mathcal{X}) \in \mathbb{R}^{2m}$ that was obtained by concatenating the vectors $v^0(\mathcal{X})$ and $v^1(\mathcal{X})$ (for more details see Chapter 6).

Therefore, we have groups of feature vectors (datasets), each one of size $2m$. In this way, we compute the datasets for several values of m . For each value of m , using these datasets we apply machine learning regression methods *SVR* (BASAK; PAL; PATRANABIS, 2007; SMOLA; SCHÖLKOPF, 2004) and *KNeighbors* (ALTMAN, 1992; COVER; HART, 1967). For each run, we randomly selected 80% of the dataset as training set and 20% as test set. Now we define the estimated parameter value of the test data as the average of all the values, as estimated by the regression model, in the test data set corresponding to the same parameter. Since the datasets are

time series of solutions, we know which solutions belong to the sets corresponding to the same parameters (even if we don't know the value of the parameter they correspond to). For this reason it is reasonable to take the estimated parameter value as the average of all the estimated values corresponding to the same parameter. We apply cross-validation by running each computation 30 times and computing the average accuracy among these 30 computations.

In the next section, we present the experiments and results to applying machine learning methods (regression) to the persistence feature vectors.

9.3 Experiments and Results

9.3.1 Predator-Prey System

As described in Subsection 9.1.1, we solve the predator-prey system (9.1) on the domain $\Omega = [0, 400] \times [0, 400]$ with spatial grid size $h = 1$, time step $\Delta t = 1/3$, and the following parameters values: $\alpha = 0.4$, $\gamma = 0.6$, and $\delta = 1$. For the computations in this work, we consider several values of the parameter β , namely, $\beta = 1.75$, $\beta = 1.80$, $\beta = 1.85$, $\beta = 1.90$, $\beta = 1.95$, $\beta = 2.00$, $\beta = 2.05$, $\beta = 2.10$, $\beta = 2.15$, and $\beta = 2.20$. For each value of the parameter β , we solve system (9.1) and consider the solutions $u(x, y, t)$ for t varying from $t = 301$ to $t = 400$ to form our dataset. Hence we have 10 datasets of solutions corresponding to 10 values of the parameter β above that we denoted by S_i with $i = 1, \dots, 10$. Since $\Delta t = 1/3$, each dataset consists of 300 solutions of (9.1).

We then fix the values of $r_{min} = 0.0010$ and $r_{max} = 0.8230$ for both the 0-dimensional and the 1-dimensional persistence diagrams, and compute the datasets of feature vectors that we denoted by $P_i(m)$ with $i = 1, \dots, 10$, each one consisting of 300 feature vectors of size $2m$. In total, this **dataset** consists of 3000 samples organized into 10 classes: the $P_i(m)$ class formed by 300 samples with $i = 1, \dots, 10$.

In Figure 41 we plot the average of estimated parameter values versus the actual parameter values for each one of the values of the parameter β , using the feature vectors of size $m = 10$. Table 14 presents the average of the R^2 and RMSE measure for $m = 10$ using KNeighbors and SVR regressor.

Table 14 – R^2 and RMSE measure using KNeighbors and SVR regressor for $m = 10$ for the predator-prey system (9.1).

Regressor \ Measure	Measure	
	R^2	RMSE
KNeighbors	0.9956	9.004×10^{-5}
SVR	0.9945	1.137×10^{-4}

Source: Research data.

Figure 41 – Average prediction values (triangles) with standard deviation error bar versus the actual value of the parameter β (first column), and average prediction (triangles) plus all the predicted values (red dots) versus the actual value of the parameter β (second column) for $m = 10$. The regressor used was KNeighbors (first row) and SVR (second row).

Source: Research data.

Figure 42 shows the plot of the average R^2 measure as a function of m . As we can see from these results, the estimated parameter values are very accurate for both regressors. In particular, the best accuracy is in the regime of $10 < m < 15$.

Figure 42 – Average R^2 values with RMSE error bars as a function of the parameter m for KNeighbors and SVR regressor.

Source: Research data.

9.3.2 Ginzburg-Landau

As described in Subsection 9.1.2, we solve the complex Ginzburg-Landau Equation (9.2) on the domain $\Omega = [0, 200] \times [0, 200]$ with time step $\Delta t = 1/2$, and considering several values of the parameter β , namely, $\beta = 1.0$, $\beta = 1.2$, $\beta = 1.4$, $\beta = 1.6$, and $\beta = 1.8$. For each value of the parameter β , we solve the Equation (9.2) and considering the solutions $u(x, y, t)$ for t varying from $t = 100$ to $t = 300$ to form our dataset. Hence we have 5 datasets of solutions corresponding to 5 values of the parameter β above that we denoted by S_i with $i = 1, \dots, 5$. Since $\Delta t = 1/2$, each dataset consists of 401 solutions of (9.2).

We fix the values of $r_{min} = -1.0227$ and $r_{max} = 0.9970$ for both the 0-dimensional and the 1-dimensional persistence diagrams, and compute the datasets of feature vectors that we denoted by $P_i(m)$ with $i = 1, \dots, 5$, each one consisting of 401 feature vectors of size $2m$. In total, this **dataset** consists of 2005 samples organized into 5 classes: the $P_i(m)$ class formed by 401 samples with $i = 1, \dots, 5$.

In Figure 43, we plot the average of estimated parameter values versus the actual parameter values for each one the values of the parameter β , using the feature vectors of size $m = 10$.

Figure 43 – Average prediction values (triangles) with standard deviation error bar versus the actual value of the parameter β (first column), and average prediction (triangles) plus all the predicted values (red dots) versus the actual value of the parameter β (second column) for $m = 10$. The regressor used was KNeighbors (first row) and SVR (second row).

Source: Research data.

Table 15 shows the average of the R^2 and RMSE measure for $m = 10$ using KNeighbors and SVR regressor. As we can see from the results in Table 15, the parameters can be estimated with very good accuracy for this equation as well.

Table 15 – R^2 and RMSE measure using KNeighbors and SVR regressor for $m = 10$ for the complex Ginzburg-Landau Equation (9.2).

Regressor \ Measure	Measure	
	R^2	RMSE
KNeighbors	1.00	0
SVR	0.9996	3.1207×10^{-5}

Source: Research data.

In Figure 44, we show the plot of the average R^2 as a function of m . As we can see from these results, the estimated parameter values are very accurate for the KNeighbors regressor in all values of m , and for the SVR regressor, the best accuracy is in the regime of $15 < m < 19$.

Figure 44 – Average R^2 values with RMSE error bars as a function of the parameter m for KNeighbors and SVR regressor.

Source: Research data.

9.4 Conclusions

We use persistent homology as a feature extractor for machine learning methods to estimate parameters in systems of equations exhibiting spatially complex patterns. One important characteristic of the method is that it is applied directly to the patterns generated by the system, and hence it can also be applied to experimental (image) data. The method presents excellent results on the datasets considered in the experiments.

CONCLUSION AND FUTURE WORKS

Topological Data Analysis (TDA) was used as a feature extractor for machine learning methods. More specifically, we used the persistent homology of datasets combined with machine learning for classifying dataset of proteins, for identifying parameters and estimate parameters in partial differential equations that exhibiting complex spatio-temporal patterns. Last, we found that proposed method is very precise and robust, this is, it presents excellent results in all dataset used.

For future works, we plan to (1) develop other techniques to vectorize the persistence diagrams; (2) use other machine learning techniques such as deep neural networks (with multiple layers between the input and output layers) and convolutional neural network (CNN) most commonly applied to analyzing visual imagery; (3) apply TDA combined with machine learning to clustering of data with rich spacial geometry; (4) apply these techniques to medical imaging.

BIBLIOGRAPHY

ABDI, H.; WILLIAMS, L. Principal component analysis. **Wiley Interdisciplinary Reviews: Computational Statistics**, Wiley Online Library, v. 2, n. 4, p. 433–459, 2010. Citation on page 82.

ALLEN, M. P. The coefficient of determination in multiple regression. **Understanding Regression Analysis**, Springer, p. 91–95, 1997. Citation on page 72.

ALTMAN, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. **The American Statistician**, Taylor & Francis Group, v. 46, n. 3, p. 175–185, 1992. Citation on page 107.

BALLABIO, D.; CONSONNI, V. Classification tools in chemistry. part 1: linear models. pls-da. **Anal. Methods**, The Royal Society of Chemistry, v. 5, p. 3790–3798, 2013. Citation on page 66.

BARKER, M.; RAYENS, W. Partial least squares for discrimination. **Journal of Chemometrics**, John Wiley & Sons, Ltd., v. 17, n. 3, p. 166–173, 2003. ISSN 1099-128X. Available: <<http://dx.doi.org/10.1002/cem.785>>. Citations on pages 62 and 66.

BASAK, D.; PAL, S.; PATRANABIS, D. C. Support vector regression. **Neural Information Processing-Letters and Reviews**, v. 11, n. 10, p. 203–224, 2007. Citations on pages 69 and 107.

BATES, D. M.; WATTS, D. G. Book; Book/Illustrated. **Nonlinear regression analysis and its applications**. [S.l.]: New York ; Chichester : Wiley, 2007. Citation on page 68.

BAUER, U. **Ripser**. 2015. Available: <<https://github.com/Ripser/ripser>>. Accessed: 30/03/2011. Citation on page 58.

BAUER, U.; KERBER, M.; REININGHAUS, J. Clear and compress: Computing persistent homology in chunks. In: **Topological methods in data analysis and visualization III**. [S.l.]: Springer, 2014. p. 103–117. Citation on page 53.

_____. Dipha (a distributed persistent homology algorithm). **Software available at <https://code.google.com/p/dipha>**, 2014. Citation on page 58.

_____. Distributed computation of persistent homology. In: SIAM. **2014 proceedings of the sixteenth workshop on algorithm engineering and experiments (ALENEX)**. [S.l.], 2014. p. 31–38. Citation on page 58.

BAUER, U.; KERBER, M.; REININGHAUS, J.; WAGNER, H. Phat-persistent homology algorithms toolbox. **Mathematical Software-ICMS 2014:4th International Congress. Proceedings,chapter**, p. 137–143, 2014. Citations on pages 57 and 58.

BERMAN, H.; WESTBROOK, J.; FENG, Z.; GILLILAND, G.; BHAT, T.; WEISSIG, H.; SHINDYALOV, I.; BOURNE, P. **The Protein Data Bank**. 2000. Available: <<http://www.rcsb.org/>>. Accessed: 10/05/2014. Citation on page 79.

- BINCHI, J.; MERELLI, E.; RUCCO, M.; PETRI, G.; VACCARINO, F. jholes: A tool for understanding biological complex networks via clique weight rank persistent homology. **Electronic Notes in Theoretical Computer Science**, Elsevier, v. 306, p. 5–18, 2014. Citation on page 57.
- BISHOP, C. M. **Pattern Recognition and Machine Learning (Information Science and Statistics)**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN 0387310738. Citations on pages 61, 62, 77, 91, and 101.
- BRERETON, R. G.; LLOYD, G. R. Partial least squares discriminant analysis: taking the magic away. **Journal of Chemometrics**, Wiley Online Library, v. 28, n. 4, p. 213–225, 2014. Citation on page 67.
- BUBENIK, P. Statistical topological data analysis using persistence landscapes. **The Journal of Machine Learning Research**, JMLR. org, v. 16, n. 1, p. 77–102, 2015. Citation on page 35.
- CALCINA SABRINA S., D. A. V.-O. M. G. F. A. R. Proteins classification through persistent homology. In preparation, 2018. Citation on page 96.
- CAMERON, A. C.; WINDMEIJER, F. A. An r-squared measure of goodness of fit for some common nonlinear regression models. **Journal of econometrics**, Elsevier, v. 77, n. 2, p. 329–342, 1997. Citation on page 73.
- CANG, Z.; MU, L.; WU, K.; OPRON, K.; XIA, K.; WEI, G. W. A topological approach for protein classification. **Molecular Based Mathematical Biology**, v. 3, n. 1, p. 140–162, 2015. Citations on pages 21, 32, 36, 79, 80, 88, and 89.
- CARLSSON, G. Topology and data. **Bulletin of the American Mathematical Society**, v. 46, n. 2, p. 255–308, 2009. Citations on pages 31, 77, 91, and 101.
- CARLSSON, G.; ISHKHANOV, T.; SILVA, V. D.; ZOMORODIAN, A. On the local behavior of spaces of natural images. **International journal of computer vision**, Springer, v. 76, n. 1, p. 1–12, 2008. Citations on pages 32 and 77.
- CGAL. **Computational Geometry Algorithms Library**. 1995. Available: <<http://www.cgal.org/>>. Accessed: 12/07/2014. Citation on page 55.
- CHAZAL, F.; GLISSE, M.; LABRUÈRE, C.; MICHEL, B. Convergence rates for persistence diagram estimation in topological data analysis. **The Journal of Machine Learning Research**, JMLR. org, v. 16, n. 1, p. 3603–3635, 2015. Citation on page 32.
- CHEN, C.; KERBER, M. Persistent homology computation with a twist. In: **Proceedings 27th European Workshop on Computational Geometry**. [S.l.: s.n.], 2011. v. 11. Citation on page 53.
- COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE transactions on information theory**, IEEE, v. 13, n. 1, p. 21–27, 1967. Citation on page 107.
- CUESTA, A.; ABREU, O.; ALVEAR, D. **Evacuation Modeling Trends**. [S.l.]: Springer, 2015. Citations on pages 91 and 101.
- DEY, T. K.; FAN, F.; WANG, Y. Computing topological persistence for simplicial maps. In: **ACM. Proceedings of the thirtieth annual symposium on Computational geometry**. [S.l.], 2014. p. 345. Citation on page 58.

EDELSBRUNNER, H. The union of balls and its dual shape. **Discrete and Computational Geometry**, v. 13, n. 1, p. 415–440, 1995. Citation on page [78](#).

_____. **Geometry and Topology for Mesh Generation**. New York, NY: Cambridge University Press, 2001. Citations on pages [31](#) and [39](#).

_____. **A Short Course in Computational Geometry and Topology**. 4. ed. London: Springer, 2014. Citations on pages [31](#), [39](#), [41](#), [43](#), [45](#), [46](#), [47](#), [48](#), [49](#), [77](#), [79](#), [91](#), [92](#), and [101](#).

EDELSBRUNNER, H.; G., H. D.; SEIDEL, R. On the shape of a set of points in the plane. **IEEE Trans. Inform Theory**, v. 29, p. 379–400, 1983. Citation on page [77](#).

EDELSBRUNNER, H.; HARER, J. **Computational Topology: an introduction**. [S.l.]: American Mathematical Soc., 2010. Citations on pages [31](#), [44](#), [45](#), [53](#), and [77](#).

EDELSBRUNNER, H.; MUCKE, E. Three dimensional alpha shapes. **ACM Trans. Graphics**, v. 13, p. 43–72, 1994. Citation on page [77](#).

EDELSBRUNNER H., L. D.; ZOMORODIAN, A. Topological persistence and simplification. **Discrete and Computational Geometry**, v. 28, n. 4, p. 511–533, 2002. Citations on pages [31](#) and [51](#).

EPSTEIN, C.; CARLSSON, G.; EDELSBRUNNER, H. Topological data analysis. **Inverse Problems**, IOP Publishing, v. 27, n. 12, p. 120201, 2011. Citations on pages [31](#) and [77](#).

FAWCETT, T. An introduction to roc analysis. **Pattern Recognition Letters**, v. 27, n. 8, p. 861–874, 2006. Citations on pages [70](#) and [71](#).

FLACH, P. **Machine learning: the art and science of algorithms that make sense of data**. [S.l.]: Cambridge University Press, 2012. Citation on page [61](#).

FOX, N. K.; BRENNER, S. E.; CHANDONIA, J. Scope: Structural classification of proteins—extended, integrating scop and astral data and classification of new structures. **Nucleic acids research**, Oxford University Press, v. 42, n. D1, p. D304–D309, 2013. Citation on page [79](#).

GAMEIRO, M.; HIRAOKA, Y.; IZUMI, S.; KRAMAR, M.; MISCHAIKOW, K.; NANDA, V. A topological measurement of protein compressibility. **Japan Journal of Industrial and Applied Mathematics**, Springer, v. 32, n. 1, p. 1–17, 2015. Citations on pages [32](#) and [77](#).

GAMEIRO, M.; MISCHAIKOW, K.; KALIES, W. Topological characterization of spatial-temporal chaos. **Physical Review E**, APS, v. 70, n. 3, p. 035203, 2004. Citations on pages [37](#) and [101](#).

GAMEIRO, M.; MISCHAIKOW, K.; WANNER, T. Evolution of pattern complexity in the cahn–hilliard theory of phase separation. **Acta Materialia**, Elsevier, v. 53, n. 3, p. 693–704, 2005. Citations on pages [37](#) and [101](#).

GARVIE, M. R. Finite-difference schemes for reaction–diffusion equations modeling predator–prey interactions in matlab. **Bulletin of mathematical biology**, Springer, v. 69, n. 3, p. 931–956, 2007. Citations on pages [32](#), [37](#), [92](#), [94](#), [101](#), and [102](#).

GHRIST, R. Barcodes: the persistent topology of data. **Bulletin of the American Mathematical Society**, v. 45, n. 1, p. 61–75, 2008. Citations on pages [91](#) and [101](#).

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A.; BENGIO, Y. **Deep learning**. [S.l.]: MIT press Cambridge, 2016. Citations on pages 61 and 77.

HEARST, M. A. Support vector machines. **IEEE Intelligent Systems**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 13, n. 4, p. 18–28, Jul. 1998. ISSN 1541-1672. Available: <<http://dx.doi.org/10.1109/5254.708428>>. Citations on pages 62, 64, and 92.

HIRAOKA, Y.; NAKAMURA, T.; HIRATA, A.; ESCOLAR, E. G.; MATSUE, K.; NISHIURA, Y. Hierarchical structures of amorphous solids characterized by persistent homology. **Proceedings of the National Academy of Sciences**, National Acad Sciences, v. 113, n. 26, p. 7035–7040, 2016. Citations on pages 32 and 77.

HOLLING, C. S. The functional response of predators to prey density and its role in mimicry and population regulation. **The Memoirs of the Entomological Society of Canada**, Cambridge University Press, v. 97, n. S45, p. 5–60, 1965. Citations on pages 32, 94, and 102.

HUHEEY, J. E.; KEITER, E. A.; KEITER, R. L.; MEDHI, O. K. **Inorganic chemistry: principles of structure and reactivity**. [S.l.]: Pearson Education India, 2006. Citation on page 80.

IVES, A. R.; EINARSSON, Á.; JANSEN, V. A.; GARDARSSON, A. High-amplitude fluctuations and alternative dynamical states of midges in lake myvatn. **Nature**, Nature Publishing Group, v. 452, n. 7183, p. 84, 2008. Citations on pages 91, 92, and 101.

KACZYNSKI, T.; MISCHAIKOW, K.; MROZEK, M. **Computational homology**. [S.l.]: Springer Science & Business Media, 2006. Citations on pages 39, 48, and 92.

KASSON, P. M.; ZOMORODIAN, A.; PARK, S.; SINGHAL, N.; GUIBAS, L. J.; PANDE, V. S. Persistent voids: a new structural metric for membrane fusion. **Bioinformatics**, Oxford University Press, v. 23, n. 14, p. 1753–1759, 2007. Citations on pages 32 and 77.

KRISHAN, K.; KURTULDU, H.; SCHATZ, M. F.; GAMEIRO, M.; MISCHAIKOW, K.; MADRUGA, S. Homology and symmetry breaking in rayleigh-bénard convection: Experiments and simulations. **Physics of Fluids**, AIP, v. 19, n. 11, p. 117105, 2007. Citations on pages 91 and 101.

KUHN, M. A short introduction to the caret package. URL: <https://cran.r-project.org/web/packages/caret/vignettes/caret.pdf>, 2016. Citation on page 67.

KURAMOTO, Y. **Chemical oscillations, waves, and turbulence**. [S.l.]: Springer Science & Business Media, 2012. Citation on page 105.

KUSANO, G.; FUKUMIZU, K.; HIRAOKA, Y. Persistence weighted gaussian kernel for topological data analysis. **International Conference on Machine Learning**, p. 2004–2013, 2016. Citations on pages 21, 32, 36, and 89.

_____. Kernel method for persistence diagrams via kernel embedding and weight factor. **arXiv preprint arXiv:1706.03472**, 2017. Citations on pages 21, 36, and 89.

LEE, H.; CHUNG, M. K.; KANG, H.; KIM, B. N.; LEE, D. S. Discriminative persistent homology of brain networks. In: IEEE. **Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on**. [S.l.], 2011. p. 841–844. Citations on pages 32 and 77.

LI, C.; OVSJANIKOV, M.; CHAZAL, F. Persistence-based structural recognition. **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**, p. 1995–2002, 2014. Citation on page 35.

MARIA, C.; BOISSONNAT, J.-D.; GLISSE, M.; YVINEC, M. The gudhi library: Simplicial complexes and persistent homology. In: SPRINGER. **International Congress on Mathematical Software**. 2014. p. 167–174. Available: <<https://project.inria.fr/gudhi/software/>>. Citation on page 58.

MISCHAIKOW, K.; NANDA, V. Morse theory for filtrations and efficient computation of persistent homology. **Discrete & Computational Geometry**, Springer, v. 50, n. 2, p. 330–353, 2013. Citation on page 92.

MITCHELL, T. M. **Machine Learning**. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072. Citations on pages 62, 63, and 91.

MOROZOV, D. Dionysus library for computing persistent homology. **Software available at <http://www.mrzv.org/software/dionysus>**, v. 2, 2012. Citation on page 58.

MURRAY, J. D. **Mathematical biology. I, volume 17 of Interdisciplinary Applied Mathematics**. [S.l.]: Springer-Verlag, New York., 2002. Citations on pages 94 and 102.

NAKAMURA, T.; HIRAOKA, Y.; HIRATA, A.; ESCOLAR, E. G.; NISHIURA, Y. Persistent homology and many-body atomic structure for medium-range order in the glass. **Nanotechnology**, IOP Publishing, v. 26, n. 30, p. 304001, 2015. Citations on pages 32 and 77.

NANDA, V. Perseus: the persistent homology software. **Software available at <http://people.maths.ox.ac.uk/nanda/perseus/index.html>**, 2012. Citation on page 57.

OBERKAMPF, W. L.; ROY, C. J. **Verification and validation in scientific computing**. [S.l.]: Cambridge University Press, 2010. Citations on pages 91 and 101.

OTTER, N.; PORTER, M. A.; TILLMANN, U.; GRINDROD, P.; HARRINGTON, H. A. A roadmap for the computation of persistent homology. **EPJ Data Science**, Springer, v. 6, n. 1, p. 17, 2017. Citations on pages 32, 44, 58, 59, and 92.

PEREIRA, C. M.; MELLO, R. F. de. Persistent homology for time series and spatial data clustering. **Expert Systems with Applications**, v. 42, n. 15, p. 6026–6038, 2015. Citation on page 35.

ROBINS, V.; TURNER, K. Principal component analysis of persistent homology rank functions with case studies of spatial point patterns, sphere packing and colloids. **Physica D: Nonlinear Phenomena**, Elsevier, v. 334, p. 99–117, 2016. Citation on page 35.

ROGERS, S.; GIROLAMI, M. **A first course in machine learning**. [S.l.]: CRC Press, 2016. Citations on pages 61, 63, 64, 77, and 91.

SARGENT, R. G. Verification and validation of simulation models. **Journal of simulation**, Springer, v. 7, n. 1, p. 12–24, 2013. Citations on pages 91 and 101.

SILVA, V. D.; GHRIST, R. Coverage in sensor networks via persistent homology. **Algebraic & Geometric Topology**, Mathematical Sciences Publishers, v. 7, n. 1, p. 339–358, 2007. Citations on pages 32 and 77.

SILVA, V. D.; MOROZOV, D.; VEJDEMO-JOHANSSON, M. Dualities in persistent (co)homology. **Inverse Problems**, IOP Publishing, v. 27, n. 12, p. 124003, 2011. Citations on pages 53 and 58.

SINGH, G.; MEMOLI, F.; ISHKHANOV, T.; SAPIRO, G.; CARLSSON, G.; RINGACH, D. L. Topological analysis of population activity in visual cortex. **Journal of vision**, The Association for Research in Vision and Ophthalmology, v. 8, n. 8, p. 11–11, 2008. Citations on pages 32 and 77.

SMOLA, A. J.; SCHÖLKOPF, B. A tutorial on support vector regression. **Statistics and computing**, Springer, v. 14, n. 3, p. 199–222, 2004. Citations on pages 69, 70, 101, and 107.

STÄHLE, L.; WOLD, S. Partial least squares analysis with cross-validation for the two-class problem: A monte carlo study. **Journal of Chemometrics**, John Wiley & Sons, Ltd., v. 1, n. 3, p. 185–196, 1987. ISSN 1099-128X. Available: <<http://dx.doi.org/10.1002/cem.1180010306>>. Citation on page 66.

TAUSZ, A.; VEJDEMO-JOHANSSON, M.; ADAMS, H. Javaplex: A research software package for persistent (co) homology. **Software available at <http://code.google.com/javaplex>**, v. 2, 2011. Citation on page 57.

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern Recognition, Fourth Edition**. 4th. ed. [S.l.]: Academic Press, 2008. ISBN 1597492728, 9781597492720. Citation on page 62.

VAPNIK, V.; GOLOWICH, S.; SMOLA, A. **Support vector method for function approximation, regression estimation and signal processing. Advanced neural information processing system**. Denver, CO. [S.l.]: USA: MIT Press, 1996. Citations on pages 69 and 101.

WANG, B.; WEI, G. W. Object-oriented persistent homology. **Journal of Computational Physics**, v. 305, p. 276–299, 2016. Citation on page 32.

WANG, L. **Support vector machines: theory and applications**. [S.l.]: Springer Science & Business Media, 2005. Citation on page 65.

WARBURTON, F. E. **Stability and Complexity in Model Ecosystems**. [S.l.]: JSTOR, 1974. Citations on pages 94 and 102.

WEINBERGER, S. What is... persistent homology? **Notices of the AMS**, v. 58, n. 1, p. 36–39, 2011. Citation on page 101.

WILD, C.; SEBER, G. **Nonlinear regression**. New Jersey: Jhon Wiley & sons, Inc., 2003. Citation on page 62.

WOLD, S.; ESBENSEN, K.; GELADI, P. Principal component analysis. **Chemometrics and intelligent laboratory systems**, Elsevier, v. 2, n. 1, p. 37–52, 1987. Citation on page 82.

WOLD, S.; SJÖSTRÖM, M.; ERIKSSON, L. Pls-regression: a basic tool of chemometrics. **Chemometrics and intelligent laboratory systems**, Elsevier, v. 58, n. 2, p. 109–130, 2001. Citations on pages 66 and 68.

WORLEY, B.; HALOUSKA, S.; POWERS, R. Utilities for quantifying separation in pca/pls-da scores plots. **Analytical Biochemistry**, v. 433, n. 2, p. 102 – 104, 2013. ISSN 0003-2697. Citation on page 66.

XIA, K.; LI, Z.; MU, L. Multiscale persistent functions for biomolecular structure characterization. **arXiv preprint arXiv:1612.08311**, 2016. Citation on page [36](#).

XIA, K. L.; WEI, G. W. Persistent homology analysis of protein structure, flexibility and folding. **International journal for Numerical Methods in Biomedical Engineerings**, v. 30, p. 814–844, 2014. Citations on pages [32](#), [36](#), and [77](#).

_____. Multidimensional persistence in biomolecular data. **Journal Computational Chemistry**, v. 36, p. 1502–1520, 2015. Citation on page [32](#).

XUN, X.; CAO, J.; MALLICK, B.; MAITY, A.; CARROLL, R. J. Parameter estimation of partial differential equation models. **Journal of the American Statistical Association**, Taylor & Francis Group, v. 108, n. 503, p. 1009–1020, 2013. Citations on pages [91](#) and [101](#).

ZHOU, W.; YAN, H. Alpha shape and delaunay triangulation in studies of protein-related interactions. **Briefings in bioinformatics**, Oxford University Press, v. 15, n. 1, p. 54–64, 2012. Citations on pages [41](#) and [42](#).

ZOMORODIAN, A.; CARLSSON, G. Computing persistent homology. **Discrete and Computational Geometry**, v. 33, n. 2, p. 249–274, 2005. Citations on pages [32](#) and [51](#).

ZOMORODIAN, A. J. **Topology for Computing**. 1. ed. New York: Cambridge University, 2005. Citations on pages [31](#), [39](#), [40](#), [43](#), and [77](#).