Foundations of Geometric Methods in Data Analysis: Projects

Frederic.Cazals@inria.fr, Frederic.Chazal@inria.fr

Academic year: 2018-19

Contents

Procedure to select the projects:

- Groups of two students must register on the following doodle `https://doodle.com/poll/m4znwm6vsm4qhbng`

  NB: Every option can be chosen by maximum 3 participant(s).
- When choosing a project, make sure to list to two lastnames, as they will be used to create account on a shared repository – see next page.

- **Deadline to fill the doodle (first come first serve basis): March 31st**
- **Deadline to return your work: May the 15th.**

0 Projects: general recommendations

0.1 Evaluation criteria

Each project will be evaluated on two criteria:

- A report presenting the answers to the questions. Recommended length: maximum 10 pages.
- The code (C, C++, python, shell scripts, etc) developed to answer the questions. In developing this code, you should
  - usability: replicating your experiments should be easy.
  - design: the code architecture (classes, modules, files) should be crystal clear. There is nothing worse than a big chunk of code with poor design / organization / documentation.

  See also the comments below.

0.2 Returning your work

You will upload your production (report, source code etc) using the Inria gforge at `https://gforge.inria.fr/`, within which the project fgmda-2018-19 has been created.

To interact with the gforge, each student will have to:

- register i.e. create a personal account
- send an inquiry to F. Cazals or F. Chazal be invited to the projects hosting all productions
- generate a pair of public/private ssh keys. then, once logged into one's gforge account: click on 'My page' > Account maintenance and put your ssh public key in there.
- finally, to retrieve the working directory – illustration on the first binom:

```
svn checkout svn+ssh://MYLOGIN@scm.gforge.inria.fr/svnroot/fgmda-2018-19/FAMILYNAME1-FAMILYNAM2

or

svn checkout --username MYLOGIN
https://scm.gforge.inria.fr/authscm/fcazals/svn/fgmda-2018-19/FAMILYNAME1-FAMILYNAM2
```

During the first week of March, the professors will extract from the doodle poll pairs of names and create the corresponding directories within the gforge repository:

```
FAMILYNAME1-FAMILYNAM2
...
```

The students will then upload their production into these directories.
NB: this is a mandatory procedure; that is, no files provided by email will be accepted.

0.3 Projects with coding: instructions.

Several projects require coding in C++ and / or python. The following recommendations are in order:

- Program options. Programs should have command-line options properly documents, in order for users to easily pass different arguments. In python, one can use the package OptionParser, see `https://docs.python.org/2/library/optparse.html`. In C++, boost program options are highly recommended, see `http://www.boost.org/doc/libs/1_62_0/doc/html/program_options.html`.

- Output of executions. Ad hoc output are not easily dealt with, unless one knows how to parse the output. Albeit verbose, xml files have two major advantages: (i) the tags allow one to comment on the output, and (ii) xml files are easily parsed with XML query language.

  For C++ users, boost provides serialization mechanisms making it very easy to dump XML files. For a starting point, check out `http://www.boost.org/doc/libs/1_62_0/libs/serialization/doc/index.html`.

  For python users, dictionaries are also easily serialized. See e.g. `https://docs.python.org/2/library/json.html`.

- Compilation for C++ code. Provide a CMakeLists.txt, from which the instructors will easily compile.

- Experiments. If you run several experiments, for example by varying one (or several) parameter(s), as requested in several projects, it is highly recommended to use a *batch manager* (BM). From a simple text file listing the options and their values, a batch manager handles all executions, by passing the relevant options on the command line.

  You can for example use the BL from the Structural Bioinformatics Library, see `http://sbl.inria.fr/doc/Batch_manager-user-manual.html`.

  In passing, if you have serialized your data structures, you can easily compute statistics using PALSE, see `http://sbl.inria.fr/doc/PALSE-user-manual.html`.

1 Search algorithms in metric trees

Description. The goal of this project is to carry out experiments on the performances of search procedures for metric trees [1].

Tasks. In the following, programming in C++ is highly recommended.

1. Provide an implementation of metric trees, with two search strategies: the exact one which uses a pruning condition to limit the number of nodes visited, and the defeatist style strategy.
2. Generate random points in a fixed dimensional Euclidean space, and use these to build a metric tree storing them. Then, run random queries on the tree built. Compare the number of nodes visited for the two search strategies. Run various experiments by:
   - varying the dimension of the ambient space – using fully dimensional data eg data drawn according to a mixture of gaussians.
   - varying the dimension of the ambient space while keeping the intrinsic dimension data constant.
3. Repeat the previous experiment for one type of complex data of your choice. Two possible options are: images to be compared with the earth mover distance; molecular conformations to be compared with the so-called lest RMSD. For molecular conformations, on can ease the energy landscape explorer from the Structural Bioinformatics Library, see `http://sbl.inria.fr/doc/Landscape_explorer-user-manual.html`
4. In designing a metric tree, the choice of the pivot choice is critical. In class, we discussed the randomized choice. Provide a deterministic strategy aiming at minimizing the number of nodes visited during a search operation.

Contact. Frederic Cazals: frederic.cazals@inria.fr

2 kd-trees and adaptation to the intrinsic dimension

Description. We have seen in class that random projection trees adapt to the intrinsic dimension of data, with in particular properties on the diameter of point set stored in nodes of the tree. We have also seen that kd-trees can fail to reduce the diameter, or may require a large number of iterations to do so. Such pathological examples, though, use specific constructions using coordinate axis. Following [2], the goal of this project is to explore a simple idea: can kd-tree adapt to the intrinsic dimension of one randomly rotates the data stored?

Tasks. In the following, programming in C++ is highly recommended.

1. Develop a procedure to compute the diameter of a set of n points in $\mathbb{R}^d$. (NB: the diameter is the maximum distance between two points.) What is its complexity? Is it optimal?
2. Develop a procedure performing a random rotation of a point set in $\mathbb{R}^d$. NB: we have seen in class how to generate a random orthonormal matrix.
3. Implement a kd-tree data structure, with a jittered split – one splits at a random point in an interval around the median. See details in [2].
4. Run tests by:
   - varying the intrinsic dimension of the data,
   - varying the width of the interval used for the jittered split

   In running these experiments, you will provide evidence on Thm2 from [2], and stress the role of the jittered split

Contact. Frederic Cazals: frederic.cazals@inria.fr

3 The Earth Mover Distance and linear programming

Description. This project investigates selected aspects of the earth move distance (EMD, [3]). As seen in class, computing the EMD reduces to solving a linear program (LP). In the sequel, the two sets of weighted points to be compared are denoted $P = \{(p_i, w_i)\}_{i=1,\ldots,m}$ and $Q = \{(q_j, w_j^{'})\}_{j=1,\ldots,n}$.

Tasks.

1. In the class, we mentioned the property according to which the number of edges which carry flow is bounded by $n + m - 1$, with m and n the number of supply and demand vertices, respectively. Prove this claim.

2. EMD reduces to a linear program, and one easily finds a number of LP solvers, for example lp_solve. These solvers generally take as input a LP written in some standard format, e.g. MPS, see `https://en.wikipedia.org/wiki/MPS_%28format%29`.

   One of them is lp_solve. Describe the algorithm used by lp_solve, and comment on its complexity. Is this complexity optimal?

3. Given the weighted points sets P and Q, write a program, in python or C++, writing the LP to a file, amenable to processing by lp_solve.

4. Generate random point sets P and Q. Then, run experiments by varying n and m. Record the running time. Are these consistent with the complexity of lp_solve ?

5. It has been shown that the EMD can be used to compare two clusterings, [4]. Propose an implementation of this algorithm.

6. Use it to compare clusterings obtained with k-means – you may use the implementation from `http://scikit-learn.org`, or from the SBL (`http://sbl.inria.fr/doc/Cluster_engines-user-manual.html`).

7. k-means is known to incur instabilities. In the case where the number of centers used is larger than the number of clusters, propose a strategy using the previous cluster comparison method to circumvent these instabilities.

Contact. Frederic Cazals: frederic.cazals@inria.fr

4 Circumventing distance concentration phenomena

Description. Several strategies can be developed to deal with distance concentration phenomena. One of them, seen in class, consists in using suitable norms, and to project to lower dimensional spaces. Another one, proposed in [5], consists in using a biasing potential which aims at focusing on the most informative distances only. In this project, we aim at applying the procedure from [5] to a different molecular data set, namely an ensemble of conformations of a protein model known as BLN69 [6]. In a nutshell, BLN69 is a linear chain of 69 beads; since each bead has 3 cartesian coordinates, a conformation is defined by a point in dimension $d = 3 \times 69 = 207$. To each conformation, one can also associate an energy, which will be given along with the conformations. Finally, to measure the distance between two conformations, we shall use the least root mean square deviation `http://sbl.inria.fr/doc/Molecular_distances-user-manual.html`.

Tasks.

1. An ensemble of $N \sim 10^6$ local minima of the BLN69 protein model can be found at `http://sbl.inria.fr/data-models`. This set is denoted $\mathcal{S}$ in the sequel. To get familiar with this data set, select a *reasonable* number of local minima with low energy, and display them in 2D using multi-dimensional scaling (MDS). For example, you may focus on the 10 lowest local minina. This set is denoted $\mathcal{T}$ in the sequel.

2. We wish to analyze pairwise distances between selected conformations. Since N precludes using all pairs, propose two procedures to:

   - select a subset $\mathcal{S}_1$ of n conformations by retaining the low energy conformations only. Hint: you may use topological persistence, see e.g. [7].
   - select a subset $\mathcal{S}_2$ of n conformations maximizing the distances between the conformations selected. Hint: you may use the smart seeding procedure used in k-means.

   Practically, you may take n in the range $[10^3, 10^4]$.

3. Using functionalities from the Molecular distances package from the SBL (`http://sbl.inria.fr/doc/Molecular_distances-user-manual.html`), produce a plot identical to [5, Fig 1 (C)] for the sets $\mathcal{S}_1$ and $\mathcal{S}_2$.

4. Analyse the distributions of pairwise distances for the sets $\mathcal{S}_1$ and $\mathcal{S}_2$. You may proceed in 2 directions:

   - As in [5], check whether portions of the distribution correspond to distances between random points drawn according to a Gaussian distribution.
   - Following the distance concentration phenomenon studied in class, you may check whether some concentration phenomenon is observed.

5. Finally, following [5], use a sigmoid function function to *tone down* non informative distances. Using MDS, compute the 2D embedding defined by the distance matrix. Use this embedding to project in 2D the local minima from the set $\mathcal{T}$. Compare to the embedding obtained for the first question.

Contact. Frederic Cazals: frederic.cazals@inria.fr

5 Multivariate two-sample tests

Description. The maximum mean discrepancy (MMD) is a powerful test statistic to compare two distributions p and q [8]. However, it critically depends on the choice of a *kernel width*, and this choice is non trivial in particular when multiple scales are present in the data [9]. The goal of this project is to explore alternative ways to compute suitable kernel widths.

Tasks.

1. Consider a 2D point cloud consisting of a grid of Gaussians blobs, as in [9]. Using the `ToMATo` algorithm seen in class, and whose code can be downloaded from `http://geometrica.saclay.inria.fr/data/Steve.Oudot/clustering/ToMATo_code.tgz` or from `http://sbl.inria.fr/doc/Morse_theory_based_analyzer-user-manual.html`, automate the tasks of clustering the data set. Note that using several persistence thresholds will yields nested clusters, for which the kernel width will be estimated as the median distance within points in a cluster.

2. Use the previous analysis to carry out an automatic kernel width calculation. Then, run tests on Gaussian blobs. You will report tables / plots for the type I and type II errors, and the experiments will be run by varying:

   - the eigenvalue ratio used to generate Gaussians blobs for the distribution q (when testing the type II error),
   - the persistence threshold used to define the clustering,
   - the dimension D of the data.

3. In class, we have studied a simple non parametric two sample test, namely the Wilcoxon Mann Whithney (U) test. Propose a novel multivariate TST performing first a random projection of the two point clouds along a random direction, and then computing the U test.

4. Test your procedure on the data used for MMD, and compare both.

5. As an improvement, one may apply the U test on a fixed (pre-defined) number of random projection – call it L. Test this idea by reporting the type I and type II errors, for various values of L, e.g. $L = 5, 20$. To control the type I error: is a Bonferroni-like correction needed?

Contact. Frederic Cazals: frederic.cazals@inria.fr

6 Detection of differences via feedback analysis

Description. In this project, we shall compare two methods studied during the class: two-sampled testing and feedback analysis.

The first method is the multivariate two-sample test (TST) MMD [8, 9], used to compare two distributions, that is $H_0 : p = q$ versus $H_1 : p \neq q$. Matlab implementations of different MMD estimators and tests are provided by the authors[1]. Two versions of MMD are provided in the R package `kernlab` in the function `kmmd`.

The second method is the discrepancy localization strategy based on the Jensen-Shannon divergence (JSD) [10], whose implementation is provided in the SBL at `http://sbl.inria.fr/doc/Density_difference_based_clustering-user-manual.html`. In short, this feedback method allows one to localize the discrepancy between two distributions, based on the following divergence:

$$\delta(z) \equiv D_{\mathrm{KL}}\left(P(\cdot|z) \| P(\cdot)\right). \tag{1}$$

Summarizing, the TST delivers a binary information (accept/reject the null), while the feedback analysis delivers clusters contributing to the JS divergence. The goal of this project is to study the relationship between both pieces of information.

Tasks.

1. In [10], the conditional probability estimation is carried out using a k-nearest neighbors based regressor. What is the rationale for doing so? What is the meaning of adaptation to the intrinsic dimension in this context [11]?
2. Data generation. In the following, we consider distributions p and q defined by a mixture of gaussians. We assume that a given distribution (and the associated sample) is parameterized by
   - the ambient dimension d,
   - the number of gaussians N,
   - a translation vector τ used to translate the mean of each gaussian.

   Write a procedure, e.g. using numpy, to generate such samples.
3. MMD and type I error. Propose a strategy to check whether MMD is a test of level α. Then, present experiments on the gaussian data. Discuss the results.
4. TST versus feedback under the null hypothesis. Consider two distributions p and q with $p = q$. Let us assume that one is willing to define a TST using the JS divergence as test statistic.

   Present a method to do so, and run experiments on the data used for question 3. NB: a permutation test may be used.
5. TST versus feedback under the alternative. Consider now distributions p and q with $p \neq q$. Note that the *magnitude* of difference between p and q may be changed by playing with the aforementioned parameter τ.
   - Run experiments to compare the variation of the test statistic of MMD, and that of the JS divergence returned by the feedback.
   - Comment on the respective variations observed, keeping in mind two key parameters of the methods, namely the kernel width used by MMD, and the number of neighbors for knn in the feedback.

Contact. Frederic Cazals: frederic.cazals@inria.fr

[1] Available at `www.gatsby.ucl.ac.uk/~gretton/mmd/mmd.htm` and `www.gatsby.ucl.ac.uk/~gretton/adaptMMD/adaptMMD.htm`.

7 Mode-seeking for detecting metastable states in protein conformations

Description. The goal of this project is to analyze protein conformations using mode-seeking techniques, in order to detect metastable states and their proximity relations.

Relevant protein conformations can be generated in various ways by exploiting the molecular dynamics. For instance, one can simulate the protein folding process at small timescales. Each conformation then gives rise to a vector with 3n coordinates, 3 per atom on the backbone (n atoms in total). One of the challenges is to understand how the conformations regroup themselves into clusters called matastable states, within which the probability of transition is high whereas it is low in-between. These states can then be fed to some stochastic process (such as a Markov chain) for efficient large timescale simulation. See [12] for more background.

The difficulty of recovering the metastable states stems from the fact that the clustering occurs in fairly high dimension (n can be of ther order of the hundreds or thousands), with data that are not sampled along linear structures and clusters that are nonconvex. This is where mode-seeking techniques can help. Assuming the data points have been sampled iid from some unknown probability distribution, the principe of mode-seeking is to use an approximation of the gradient flow of the probability density function to push the data points towards the density maxima. These maxima then serve as cluster centers, and their preimages through the gradient flow are their corresponding clusters.

In this project we will use the topology-based method `ToMATo` [13] to cluster the conformations. The goal is to get the same kind of results as in [12] and [13].

Tasks.

1. Collect the data:
   - the set of alanine dipeptide conformations (`http://geometrica.saclay.inria.fr/team/Fred.Chazal/Centrale2017.html`): 3 coordinates per atom, 10 atoms per conformation, 1 atom per line (so 10 lines per conformation, seen as a 30-dimensional point)
   - the set of conformations projected down to 2 dimensions for visualization (`http://geometrica.saclay.inria.fr/team/Fred.Chazal/Centrale2017.html`)
2. Compute the RMSD distance matrix between the 30-dimensional conformations (RMSD = Root Mean Square Deviation, see `https://en.wikipedia.org/wiki/Root-mean-square_deviation_of_atomic_positions` for the definition, and `https://github.com/pandegroup/IRMSD` for some code to compute the RMSD).
3. Retrieve the code for `ToMATo` at `http://geometrica.saclay.inria.fr/team/Fred.Chazal/Centrale2017.html` and get familiar with it, e.g. try it out on the toy examples provided in the archive then play around with the parameters.
4. Try applying ToMaTo to the computed RMSD distance matrix. Beware that:
   - the code takes in a matrix of point coordinates and uses the Euclidean distance by default -¿ you should tweak the file Distance.h to your needs
   - the data are huge so the method will need some degree of optimization to scale up. Alternatively, you may want to consider applying it to subsamples of your data, although in that case you will need to find a mechanism to ascertain your results.
5. Hopefully you will be able to recover the same kind of result as in [12] and [13]. Don't forget to read these papers to get some insight into the data and their interpretation!

Contact. Frédéric Chazal: frederic.chazal@inria.fr

8 TDA for financial time series: persistent homology and landscapes

Description. The goal of this project is to analyze the evolution of daily returns of four major US stock markets indices (DowJones, Nasdaq, Russell2000, SP500) over the period $1989 - 2016$ using persistent homology following the approach proposed in [14]. A classical approach in TDA to extract topological features from multivariate time-series taking its values in $\mathbb{R}^d$ (here, since we are considering the evolution of four indices $d = 4$) consists in using a sliding window of fixed length w to generate a sequence of w points in $\mathbb{R}^d$. Using the Vietoris-Rips filtration, the persistence diagram of each of these point cloud is then computed and used as a topological feature for further analysis or processing of the initial data.

This project aims at reproducing the experiments of [14] and explore and discuss a few variants.

Tasks.

1. Download the paper [14] and the data from the following address: `http://geometrica.saclay.inria.fr/team/Fred.Chazal/Centrale2017.html`. Have a quick look at the whole paper [14] to get used to the considered problem and proposed approach, and a more careful reading of Sections 3.1 and 4.

2. Write a function to compute persistence landscapes. This function should take as input a persistence diagram dgm (in the Gudhi format), a dimension k, the endpoints x_{min}, x_{max} of an interval, the number nb_{nodes} of nodes of a regular grid on the interval $[x_{min}, x_{max}]$ and a number of landscapes nb_{ld}, and output a $nb_{ld} \times nb_{nodes}$ array storing the values of the first nb_{ld} landscapes of dgm on the node of the grid.

3. Use the landscape function to run the experiments done in Section 4 of [14] but taking windows of length $w = 40$ and $w = 80$ and $w = 120$. Compare your results to the ones provided in the paper (are they very similar).

4. Propose and experiment another method, than just computing the norm of landscapes (e.g. consecutive bottleneck distances between persistence diagrams, norm of the difference between consecutive landscapes,...). Briefly discuss and compare your results to the ones of Section 4 in [14].

Software. For persistent homology computations, the use of the GUDHI library (`http://gudhi.gforge.inria.fr/`) is strongly recommended (C++ or Python version). Alternately, you can use the R package TDA : `https://cran.r-project.org/web/packages/TDA/index.html`

Contact. Frédéric Chazal: frederic.chazal@inria.fr

9 Persistent homology for smartphone data analysis (pedestrian recognition)

Description. The goal of this is project to illustrate, on a toy example, the benefit of "coordinate invariance" of persistent homology. The walk of 3 pedestrians A, B and C, has been recorded using the accelerometer sensor of a smartphone carried in the their pocket, giving rise to 3 multivariate time series in $\mathbb{R}^3$: each time series represents the 3 coordinates of the acceleration of the corresponding pedestrian in a coordinate system attached to the sensor. As, the smartphone was carried in unknown different positions and was not fixed, these time series cannot be compared coordinates by coordinates. Using a sliding window, each series has been splitted in a list of 100 times series made of 200 consecutive points, that are stored in data_A, data_B and data_C. To each set of 200 points is associated a label A, B or C stored in label (see the data set and the Python script to load the data). The objective is to compute the persistence diagrams of these 3D point clouds and use them to achieve a pedestrian recognition task (supervised setting).

Note: This project requires some (basic) knowledge of learning (random forests).

Tasks.

1. Download the data together with a Python script to load it at the following address: `http://geometrica.saclay.inria.fr/team/Fred.Chazal/Centrale2017.html`.
2. Compute and save the 0-dimensional and 1-dimensional persistence diagrams of the Rips filtrations (or alternately the alpha-shape filtrations) built on top of each of the 300 point clouds in $\mathbb{R}^3$.
3. Compute the matrices of pairwise bottleneck distances between diagrams and use a dimensionality reduction algorithm to visualize them in 2D and 3D (e.g. Multidimensional Scaling).
4. Write a function to compute persistence landscapes. This function should take as input a persistence diagram dgm (in the Gudhi format), a dimension k, the endpoints x_{min}, x_{max} of an interval, the number nb_{nodes} of nodes of a regular grid on the interval $[x_{min}, x_{max}]$ and a number of landscapes nb_{ld}, and output a $nb_{ld} \times nb_{nodes}$ array storing the values of the first nb_{ld} landscapes of dgm on the node of the grid. Check, on some simple examples that your code is correct.
5. For each 0-dimensional and 1-dimensional persistence diagrams, compute the first 5 landscapes on a relevant interval with a few hundred of nodes. Splitting randomly the data set into a 80/20 learning/test data, use a random forest to explore the performances of the 0-dimensional or 1-dimensional landscapes to classify pedestrians. An example of code to realize such an experiment can be downloaded at `http://geometrica.saclay.inria.fr/team/Fred.Chazal/Centrale2017.html`. Compare the results you obtain using 0-dimensional landscapes, 1-dimensional landscapes or both.
6. Do the same experiment as previously, but using the raw data (3×200 array of acceleration coordinates). Compare the obtained classification results to the previous one.

Software. For persistent homology computations, the use of the GUDHI library (`http://gudhi.gforge.inria.fr/`) is strongly recommended (C++ or Python version).
The use of the Scikit Lear library (Python) is highly recommended for the dimensionality reduction and classification experiments.

Contact. Frédéric Chazal: frederic.chazal@inria.fr

References

[1] Peter N Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *ACM SODA*, volume 93, pages 311–321, 1993.

[2] S. Vempala. Randomly-oriented kd trees adapt to intrinsic dimension. In *FSTTCS*, pages 48–57, 2012.

[3] Y. Rubner, C. Tomasi, and L.J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[4] D. Zhou, J. Li, and H. Zha. A new mallows distance based metric for comparing clusterings. In *ICML*, pages 1028–1035. ACM, 2005.

[5] M. Ceriotti, G. Tribello, and M. Parrinello. Simplifying the representation of complex free-energy landscapes using sketch-map. *PNAS*, 108(32):13023–13028, 2011.

[6] A. Roth, T. Dreyfus, C.H. Robert, and F. Cazals. Hybridizing rapidly growing random trees and basin hopping yields an improved exploration of energy landscapes. *J. of Computational Chemistry*, 37(8):739–752, 2016.

[7] F. Cazals, T. Dreyfus, D. Mazauric, A. Roth, and C.H. Robert. Conformational ensembles and sampled energy landscapes: Analysis and comparison. *J. of Computational Chemistry*, 36(16):1213–1231, 2015.

[8] A. Gretton, K.M. Borgwardt, J.R. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[9] A. Gretton, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, K. Fukumizu, and B.K. Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *Advances in Neural Information Processing Systems*, pages 1205–1213, 2012.

[10] F. Cazals and A. Lhéritier. Beyond two-sample-tests: Localizing data discrepancies in high-dimensional spaces. In P. Gallinari, J. Kwok, G. Pasi, and O. Zaiane, editors, *IEEE/ACM International Conference on Data Science and Advanced Analytics*, Paris, 2015. Preprint: Inria tech report 8734.

[11] Samory Kpotufe. k-nn regression adapts to local intrinsic dimension. In *Advances in Neural Information Processing Systems*, pages 729–737, 2011.

[12] J. Chodera, W. Swope, J. Pitera, and K. Dill. Long-time protein folding dynamics from short-time molecular dynamics simulations. *Multiscale Modeling & Simulation*, 5(4):1214–1226, 2006.

[13] F. Chazal, L. Guibas, S. Oudot, and P. Skraba. Persistence-based clustering in riemannian manifolds. *J. ACM*, 60(6):1–38, 2013.

[14] Marian Gidea and Yuri A Katz. Topological data analysis of financial time series: Landscapes of crashes. *Physica A*, 491:820–834, 2018.