Geometric and topological methods in machine learning

Jean-Daniel.Boissonnat@inria.fr, Frederic.Cazals@inria.fr, Mathieu.Carriere@inria.fr

Academic year: 2021-22

Contents

Procedure to select the projects:

- Groups of two students must register on the following doodle `https://doodle.com/poll/uu2s7b52qfvrsxaq?utm_source=poll&utm_medium=link`

  NB: Every option can be chosen by maximum 3 groups.
- When choosing a project, make sure to write the two last names.
- **Deadline to fill the doodle (first come first serve basis): January the 3rd, 2022**
- **Deadline to return your work: February the 3rd, 2022**

0 Projects: general recommendations

0.1 Evaluation criteria

Each project will be evaluated on two criteria:

- A pdf 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 focus on:
  - 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

Send an email to the project supervisor (Jean-Daniel.Boissonnat@inria.fr, Frederic.Cazals@inria.fr or Mathieu.Carriere@inria.fr), with a link to a zip file containing ALL materials listed above.

0.3 Projects with coding: instructions.

Several projects require coding in C++ or Python. The following recommendations are in order:

- Program options. Programs should have command-line options properly documented, 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 RPTrees, vector quantization, and dimensionality reduction

Description. Given a set of data points in $\mathbb{R}^d$, *vector quantization* (VQ) is the problem concerned with the identification of k representative points. Wen the criterion to be minimized is the sum of squared distances between the data points and their representative, VQ boils down to solving k-means.

As always when dealing with real - high dimensional data, an interesting case is that where VQ is to be performed on a dataset living in a sub-manifold. In other words: one would like to combine two key techniques, namely (i) dimensionality reduction, and (ii) k-means. This strategy is the one explored in [1], based on *random projection trees* which have been studied in class.

Tasks.

1. Summarize the RPTree based method from [1], by stressing the role of the two types of splits used (by projection and by distance).

2. Summarize the theoretical guarantees, and compare their merits with the results yielded by k-means for vector quantization.

3. Provide a C++ implementation of this algorithm, as a C++ class parameterized by a traits class providing the types for the points and the associated distance. Indeed, the experiments below consist of processing two types of points (in the Euclidean space, and on the flat torus).

   - For the generic C++ template based design, a toy example is provided here `https://sbl.inria.fr/data-models/example-template-based-C++-programming.zip`

4. Test-case #1. In this first test-case, we consider two datasets: a d-dimensional plane embedded into $\mathbb{R}^D$, and the 2-dimensional swiss roll embedded into $\mathbb{R}^D$. The distance between two points is the standard Euclidean distance. Instantiate the algorithm with this distance, and present experiments on the quantization error, by varying d, D and the number of points n.

   Compare with the quantization error yielded by k-means or k-means++ [2].

5. Test-case #2. In this second test-case, we consider molecular data. Recall that a protein is a polymer of small molecules called amino-acids. Given a protein molecule, the so-called *internal coordinates* comprise three types of terms: bond lengths, valence angles, and dihedral angles, see `https://sbl.inria.fr/doc/Molecular_coordinates-user-manual.html`. In the sequel, we are concerned with dihedral angles in proteins, see also `https://en.wikipedia.org/wiki/Dihedral_angle`.

   In dealing with these angular data, note that the distance is measured on the flat torus; that is, for each angular coordinate, one takes the shortest distance on S^1.

   Dihedral angles collected on amino-acids are used to answer a key question: how many clusters can one use to represent all conformations of a given a.a. ? When this question admists a clear answer, one may indeed discretize the conformations of the a.a..

   Let us consider two datasets for three different amino acids, namely lysine, arginine, and methionine, see `https://sbl.inria.fr/data-models/dihedrals-from-dunbrack-2011--clustering/` :

   - χ dataset: dihedral angles on the side chain only.
   - ϕ, ψ, χ dataset: dihedral angles on the side chain + the two dihedral angles of the backbone ϕ and ψ.

   Instantiate the RPTree construction with the angular distance. Then, run it on each datasets. Document the type of slit used. By plotting the quantization error as a function of the number of leaves in the RPTree, investigate the following question: for each dataset, what is a sound number of clusters?

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

2 Intrinsic dimension estimation via 2 nearest neighbors

Description. The dimensionality reduction (DR) methods studied in class do not explicitly use an analytical model for the distribution of points. A method doing so is presented in [3]. In short, this method uses a homogeneous Poisson process to represent the distribution of neighbors about a particular point, and to derive a statistic whose cumulated distribution function can be used to estimate the intrinsic dimension (D). Naturally, the ID can be used in a second step to perform DR.

Tasks.

1. Briefly explain the algorithm from [3]. In reviewing the algorithm, detail the role of the *constant density* assumption used in the derivations.

2. Provide a C++ implementation of this algorithm as a C++ class parameterized by a traits class providing the types for (i) the points, (ii) the associated distance, and (iii) the algorithm returning the two nearest neighbors. Indeed, the experiments below consist of processing two types of points (in the Euclidean space, and on the flat torus).

   - For the generic C++ template based design, a toy example is provided here `https://sbl.inria.fr/data-models/example-template-based-C++-programming.zip`
   - You are allowed to incorporate an external class providing a nearest neighbor search algorithm.

3. Test-case #1: original experiments. Intantiate the algorithm using the Euclidean distance, possibly with boundary conditions. Then, replicate the experiments from [3] using the same distributions: uniform in a cube, the swiss roll, and the Cauchy distribution.

4. Test-case number #2: mixtures. The method from [3] assumes a single distribution. In practice, it is often the case that the intrinsic dimension varies locally in the point cloud. For example, using the point sets from test-case #1, we can consider a point cloud mixing two sub-samples from 2 or 3 distributions, or from the same distribution but with different dimensions. We call such a point cloud a *mixture*, and the individual point clouds its *components*.

   Present two additional experiments: a mixture based on two different distributions, and a mixture with one distribution but using two different dimensions (e.g. a cube with periodic boundary conditions in dimensions d_1 and d_2 with $d_1 \neq d_2$.)

5. Generalizing the algorithm. Propose a generalization of the algorithm from [3], able to cope with a mixture composed of a fixed number of components.

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

3 Detecting metastable states in protein conformations with ToMATo

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 metastable 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 [4] 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 i.i.d. 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 to cluster the conformations. The goal is to get the same kind of results as in [4] and [5].

Tasks.

- Collect the data (see Data below):
  - the set of alanine dipeptide conformations: 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.
- Compute the RMSD distance matrix between the 30-dimensional conformations (RMSD = Root Mean Square Deviation), using you own code. See `https://en.wikipedia.org/wiki/Root-mean-square_deviation_of_atomic_positions` for the definition.
- Retrieve the code for ToMATo (see Code below) and get familiar with it, e.g. try it out on the toy examples provided in the archive then play around with the parameters.
- Try applying ToMATo to the computed RMSD distance matrix. Beware that the data is huge so 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.
- Hopefully, you will be able to recover the same kind of result as in [4] and [5]. Read these articles and compare your results to theirs.

Data. The data sets can be downloaded at `http://www-sop.inria.fr/abs/teaching/uca-master-data-science-GTML/exam_mathieu/TDA_projects.html`.

Code. For ToMATo computations, the use of the GUDHI library (`https://gudhi.inria.fr/python/latest/`) is strongly recommended.

Contact. Mathieu Carrière: `mathieu.carriere@inria.fr`

4 Analyzing contact maps with Mapper

Description. The goal of this project is to analyze a data set of single-cell Hi-C contact maps. An Hi-C contact map is a pairwise distance matrix (computed from a given human cell) that encodes how chromatin is folded in the nucleus: each row and column of the matrix represents a small DNA window, and each entry in the matrix is the spatial distance between these windows in the nucleus. If chromatin is straight, the contact map will have very few extra diagonal terms, while folded chromatin induces a denser matrix (since DNA windows at large genomic distances can be spatially close). Since the cell cycle strongly influences how chromatin folds, it can be efficiently characterized with contact maps. Moreover, the cell cycle is one of the several biological phenomena that induces strong biases in single cell data sets, and its detection is thus critically needed. This project aims at recovering the topological structure of the cell cycle (a loop) using Mappers computed on contact maps, processed with Stratum-adjusted Correlation Coefficients (SCC) [6].

Tasks.

- Download the data set (see Data below). Take a look at the SCC article [6], in particular "2D mean filter smoothing", "Stratification by distance" and "Stratum-adjusted correlation coefficient (SCC)" in "Methods" section, and implement a function that takes as inputs two contact maps, and outputs the SCC.
- Implement your own Mapper algorithm (bonus), or use existing code (see Code below), and compute the Mapper using the eigenfunctions of a Kernel PCA run on the pairwise SCC matrix as filters. Try various Mapper and SCC parameters, and interpret their influence on the Mapper shape. Beware that the data is huge so 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.
- Find a set of parameters for which the cell cycle can be detected as a big loop in the Mapper. Prove this loop indeed represents the cell cycle by coloring the Mapper nodes with various markers correlated with the cell cycle, such as "mean insu", "f near band", "f mitotic band", "repli score" in provided feature file (see Data below).
- Quantify the cell cycle statistical robustness (whether it is an artifact of computation or not) by bootstrapping the data: subsample the data set (with replacement) many times, and count the number of runs in which the cell cycle is detected.
- Compare and discuss your results with directly running dimensionality reduction on the raw contact maps.

Data. The data set of contact maps can be downloaded at: `http://www-sop.inria.fr/abs/teaching/uca-master-data-science-GTML/exam_mathieu/TDA_projects.html`. This folder contains the contact maps encoded in sparse COO matrices from the SciPy package. The rows and columns are small DNA windows, and the chromosome they belong to is encoded in the file "chromosomes.txt". The feature file containing the cell info is "features.txt".

Code. Mapper can be computed with Scikit-TDA: `https://scikit-tda.org/` or Giotto: `https://giotto-ai.github.io/gtda-docs/0.4.0/library.html`.

Contact. Mathieu Carrière: `mathieu.carriere@inria.fr`

5 Analyzing financial time series with persistent homology

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 [7]. A classical approach in TDA to extract topological features from multivariate time-series with values in $\mathbb{R}^d$ ($d = 4$ here, since we are considering the evolution of four indices) 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 clouds 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 [7] and explore and discuss a few variants.

Tasks.

- Download the article [7] and the data set (see Data below). Have a quick look at the whole article [7] to get used to the considered problem and proposed approach. Have a careful reading of Sections 3.1 and 4.
- Write a function to compute persistence landscapes WITHOUT using the GUDHI library. This function should take as input a persistence diagram D (in the GUDHI format), a dimension k, the endpoints $x_{\min}$, $x_{\max}$ of an interval, the number n of nodes of a regular grid on the interval $[x_{\min}, x_{\max}]$ and a number of landscapes m, and should output an $m \times n$ array storing the values of the first m landscapes of the persistence diagram D on the nodes of the grid. Check on some simple examples that your code is correct.
- Use the landscape function to run the experiments done in Section 4 of [7], using windows of length $w = 40$ and $w = 80$ and $w = 120$. Compare your results to the ones provided in the article: are they similar or not?
- Propose and experiment other methods, than just computing the norm of landscapes[1]. Briefly discuss and compare your results to the ones in Section 4 of [7].

Data. The data set can be downloaded at the following address: `http://www-sop.inria.fr/abs/teaching/uca-master-data-science-GTML/exam_mathieu/TDA_projects.html`.

Code. For persistent homology computations, the use of the GUDHI library (`https://gudhi.inria.fr/python/latest/`) is strongly recommended.

Contact. Mathieu Carrière: `mathieu.carriere@inria.fr`

[1] For instance, you can try computing the consecutive bottleneck distances between persistence diagrams, the norm of the differences between consecutive landscapes, etc

6 Dimensionality reduction for persistent homology

Description. Due to the curse of dimensionality, Topological Data Analysis is difficult to apply on data living in very high dimension. Various techniques have been proposed to walk around the curse of dimensionality. One of them uses random projections in lower affine subspaces. The central result is the Johnson-Lindenstrauss lemma that states that any subset P of n points of Euclidean space $\mathbb{R}^D$ can be embedded via random projection in a subspace of (lower) dimension $d = O(\log n/\epsilon^2)$ without modifying the interpoint distances by more than a multiplicative factor of $1 \pm \epsilon$, i.e., for any two points $p_i, p_j \in P$ with images p'_i and p'_j , we have $(1-\epsilon)\|p_i - p_j\| \leq \|p'_i - p'_j\| \leq (1+\epsilon)\|p_i - p_j\|$.

Tasks.

- Write a short survey of the main results about dimensionality reduction using random projections and their applications in Data Analysis (nearest neighbour search, clustering, persistent homolgy, etc.).
- Read the two papers [8, 9]. The goal of the two papers is to show that the Čech filtration can be $(1+\epsilon)$-approximated using random projections. The second paper uses the Euclidean distance while the other one uses the k-distance which is more robust to noise and outliers.
- Implement the algorithms in those papers for the Čech filtration using respectively the Euclidean distance and the k-distance.

Contact. Jean-Daniel Boissonnat: `jean-daniel.boissonnat@inria.fr`

References

[1] S. Dasgupta and Y. Freund. Random projection trees for vector quantization. *Information Theory, IEEE Transactions on*, 55(7):3229–3242, 2009.

[2] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *ACM-SODA*, page 1035. Society for Industrial and Applied Mathematics, 2007.

[3] Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):1–8, 2017.

[4] 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.

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

[6] T. Yang, F. Zhang, G. G. Yardımcı, F. Song, R. C. Hardison, W. S. Noble, F. Yue, and Q. Li. Hicrep: assessing the reproducibility of hi-c data using a stratum-adjusted correlation coefficient. *Genome Research*, 27(11):1939–1949, 2017.

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

[8] Shreya Arya, Jean-Daniel Boissonnat, Kunal Dutta, and Martin Lotz. Dimensionality Reduction for k-Distance Applied to Persistent Homology. In *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164, pages 10:1–10:15. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

[9] Donald Sheehy. The persistent homology of distance functions under random projection. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, pages 328–334. Association for Computing Machinery, 2014.