Topological Jet Tagging

Dawson Thomas
Department of Physics
Yale University
New Haven, CT 06511
dawson.thomas@yale.edu

Sarah Demers
Department of Physics
Yale University
New Haven, CT 06511
sarah.demers@yale.edu

Smita Krishnaswamy
Department of Computer Science
Yale University
New Haven, CT 06511
smita.krishnaswamy@yale.edu

Bastian Rieck
Inst. of AI for Health, Helmholtz Munich
Technical University of Munich
Munich, Germany
bastian.rieck@helmholtz-munich.de

Abstract

Proton–proton collisions at the large hadron collider result in the creation of unstable particles. The decays of many of these particles produce collimated sprays of particles referred to as *jets*. To better understand the physics processes occurring in the collisions, one needs to classify the jets, a process known as *jet tagging*. Given the enormous amount of data generated during such experiments, and the subtleties between different signatures, jet tagging is of vital importance and allows us to discard events which are not of interest—a critical part of dealing with such high-throughput data. We present a new approach to jet tagging that leverages topological properties of jets to capture their inherent shape. Our method respects underlying physical symmetries, is robust to noise, and exhibits predictive performance on par with more complex, heavily-parametrized approaches.

1 Introduction

Experimental particle physics enjoys some of the largest data sets in scientific history. The High-Luminosity LHC, for example, is expected to enter the exabyte regime. Like other scientific fields, technological advances in particle physics have resulted in an increasing need to understand large high-dimensional data sets. Such high-dimensional data is not amenable to traditional statistical analysis methods, which are often unable to capture the underlying shape of the data set. However, due to correlations between features, many real-world data sets tend to lie *on* or *near* a much lower-dimensional manifold embedded in the ambient feature space. This is the so-called *manifold hypothesis*, which remains at the heart of modern geometric and topological methods for data analysis. Of particular interest in this context are methods based on the recently-emerging paradigm of *topological data analysis* [TDA]. Deeply rooted in the field of algebraic topology, TDA methods capture topological information—e.g., the number of holes, or the number of connected components—about the data manifold. Understanding the underlying topology not only leads to deeper insights about the data, but also provides a convenient, stable way to "featurize" high-dimensional data sets in a way that captures multi-scale information.

Despite advantageous robustness and stability properties [1], TDA methods have not been widely adopted for particle physics. It is our aim to show that topological techniques stand to enormously benefit the field. Importantly for this paper, as we shall see, topological features respect a set of symmetries, making them well-suited to physical applications. In particular, topological features are *provably stable* with respect to small perturbations of the underlying point cloud. We take advantage of these assets, applying TDA for jet tagging.

2 Background

2.1 Topological Data Analysis

Topological data analysis is based on the idea of capturing the shape of a data set at multiple scales. This is achieved by first approximating a data set using a *simplicial complex* K, i.e., a generalization of a graph. The connectivity of K is subsequently described using *simplicial homology*, a framework to assign a set of graded groups to K, the homology groups. The elements of the d-dimensional homology group correspond to the topological features of K, i.e., *connected components* ($d = 0$), *tunnels* ($d = 1$), and *voids* ($d = 2$). The number of d-dimensional topological features is known as the dth Betti number $\beta_d \in \mathbb{N}$; it is a characteristic property of an input data set. Extending simplicial homology to point clouds results in *persistent homology* [2], a multi-scale extension of simplicial homology, making it possible to describe homology groups at different spatial scales ϵ, thus accommodating that real-world data have features occurring at multiple scales. Multi-scale topological features are then stored in a *persistence diagram* $\mathcal{D}$, a set of points in $\mathbb{R}^2$; every point $(c, d) \in \mathcal{D}$ represents a topological feature created at scale c and destroyed at scale d. Persistence diagrams—and other topological descriptors—have shown exceptional promise in numerous tasks such as shape reconstruction [3] and graph classification [4]; we refer to a recent survey for more details on the use of topological features in machine learning [1].

2.2 Jets

In the particle physicist's worldview, known as the Standard Model [SM], all visible matter in the universe is composed from six types (flavors) of quarks and six flavors of leptons. Each of these has an associated anti-particle, and they all interact via three fundamental forces (ignoring gravity), which are mediated by particles called bosons. This simplified explanation obscures a rich history of discovery and immense practical challenges. Only a handful of these particles are stable, and producing the more exotic ones requires high energies accessed via complex experimental endeavors such as the Large Hadron Collider [LHC]. Due to a physical process known as confinement, quarks produced in these collisions cannot exist on their own. It is energetically favorable for them to pull new particles out of the vacuum, which also subsequently decay into new particles. Additionally, the production and radiation of gluons produce particles. The result is a collimated shower of particles known as a jet. Experimentally, a jet presents as a collection of tracks and energy deposits, which are clustered together. Jet tagging refers to the task of identifying the type of particle that initiated a jet. One can directly use certain observables to tag jets. For example, *b* quark-initiated jets form short lived particles that travel a small distance before decaying, which can be identified by secondary vertices for the tracks in the jet. As another example, gluon jets tend to be wider and contain more particles [5] than quark jets. However, these selections alone achieve limited success, and as a result, jet-tagging has provided a wonderful playground for machine learning frameworks within high-energy physics. For instance, RNNs [6, 7, 8], CNNs (e.g., ATL [9]), and GNNs [10] have seen applications for this task.

2.3 Symmetries and Motivation

Symmetries in physics refer to a set of transformations under which the laws of physics are invariant. The symmetries of Einstein's special theory of relativity are described by the Poincaré group, which can be understood as the semi-direct product of the group of spacetime translations with Lorentz transformations, which relate space and time coordinates between moving frames. Lorentz transformations further decompose into spatial rotations, and *boosts*, where the direction of motion is in line with a particular coordinate axis. This structure of the underlying physics thus motivates our use of topological features as physically meaningful representations of jets: the topological features of jet point clouds are *invariant* under homeomorphisms (e.g. translations and rotations) and permutations of particles (which is a symmetry that e.g. RNNs violate). Furthermore, we claim that topological features are at least *approximately* Lorentz invariant. Boosted jets tend to squeeze in along the axis of motion. Under a boost, then, we can think of each particle as displacing by some small distance, which bounds the (Gromov–)Hausdorff distance between the original and boosted point clouds. This serves as an upper bound for the (Wasserstein) distance between topological features [11], implying that the persistence diagrams of the two jets are not too different. We are led to suspect that the topology can provide a robust representation of the jet for use in classification. Despite these advantageous properties, we are only aware of a single other work that employs topology to study jets [12]. While the authors demonstrate that topological features on average can distinguish distributions of jets, their work uses momentum information, whereas our work is the first to provide a direct classification of *individual jets* based on *coordinate* information, hoping to encourage others to follow in this novel research direction.

3 Methods and Results

We consider the quark-gluon tagging data set from Komiske et al. [13]. The goal is to distinguish the light quark (signal) jets from the gluon (background) jets. The jets in this data set are simulated with PYTHIA8 according to the processes $q\bar{q} \to Z(\to \nu\bar{\nu}) + g$ and $q\bar{q} \to Z(\to \nu\bar{\nu}) + (u, d, s)$ in $\sqrt{s} = 14$ TeV pp collisions. The final state particles, excluding neutrinos, are clustered into jets with the anti-k_T algorithm ($R = 0.4$). Only jets with transverse momentum $500 \le p_T \le 550$ GeV and rapidity $|y| < 1.7$ are kept. Each jet is stored as a collection of particles, each with transverse momentum p_T, rapidity, azimuthal angle, and ground truth particle ID $(p_T, y, \phi, \text{PID})$. We exclude PID for training purposes, as this feature is not available in experiment. We used a split of 375k, 75k, 50k training, test, and validation jets, comprising a quarter of the full 2M jet dataset.

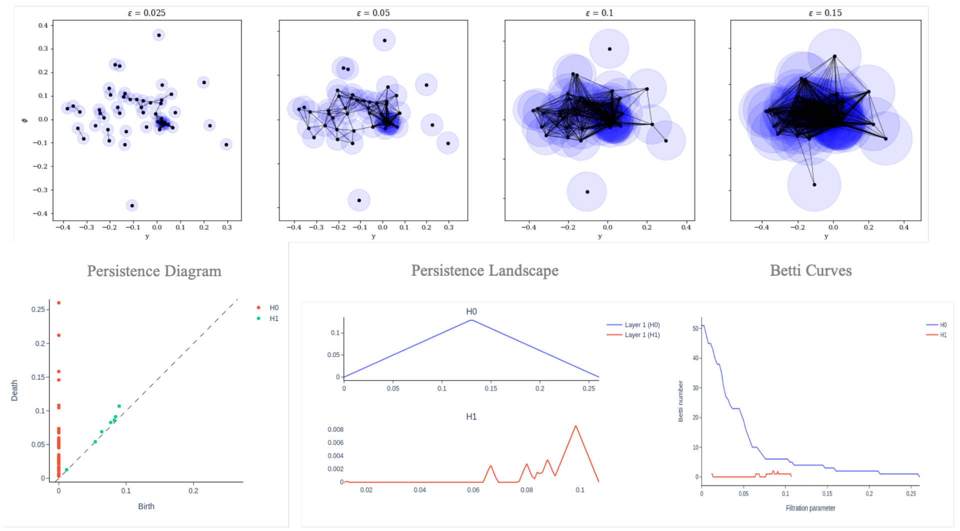

Figure 1: Illustration of the method on an example gluon jet. In the top row, growing Euclidean balls around individual data points results in an approximation of the topology of the dataset at ever-increasing scales. The birth and death scales of the topological features are encoded in a persistence diagram (bottom right). These can be represented hierarchically as a "landscape" (middle) or alternatively as a Betti curve (left), which counts the active topological features at each scale. These representations are fed to our RFs for training.

Models. Since the Lorentz invariance arguments above apply to the spatial coordinates, we computed Vietoris-Rips filtrations by Euclidean distances in (y, ϕ) and in (p_T, y, ϕ) space. From the persistence diagrams, we computed persistence landscapes and Betti curves (each with 100 bins), as shown in Figure 1. For comparison, we also computed Betti curves from momentum superlevel set filtrations over the jets' Delaunay triangulations, following the methods of Li et al. [12]. To showcase the potential of topological techniques, we trained ensemble classifiers — based on random forests (RFs) — on these topological representations of jets. RF hyperparameters were found by performing a halving grid search using 5-fold cross validation on the validation set. In all cases, the optimal parameters were found to be a maximum depth of 10, 100 minimum samples to split, and 100 trees, except for the RF trained on Vietoris-Rips Betti curves with p_T, where the optimal minimum samples to split was found to be 10. We then fit the RFs with these optimal parameters to the training

Classifier	**AUC**	$\frac{1}{\varepsilon_b}$ at $\varepsilon_s = 70\%$	$\frac{1}{\varepsilon_b}$ at $\varepsilon_s = 50\%$	$\frac{1}{\varepsilon_b}$ at $\varepsilon_s = 30\%$
DNN	0.870	7.9	23.5	75.5
EFN	0.855	7.2	23.5	76.1
PFN	0.873	8.3	23.9	69.0
Mass	0.740	2.7	7.9	32.8
Multiplicity	0.841	5.8	19.0	51.1
RF Curves	0.847	6.4	18.6	56.8
RF Curves with pt	0.846	6.3	18.4	52.3
RF Landscapes	0.779	3.6	7.6	17.9
RF Landscapes with pt	0.798	4.1	9.2	22.5
RF Superlevel pt Curves	0.832	5.3	14.3	42.1

Table 1: Performance and background rejection comparisons.

data, and computed ROC curves and rejection metrics $1/\varepsilon_b$ based on the test data. Selected code is available in Appendix A. For comparison, we trained some popular deep approaches using the same train/test split. In particular, we compared to (i) Energy Flow Networks [EFN] and Particle Flow Networks [PFN] using the `EnergyFlow` package [13], (ii) a Dense Neural Network trained on the N-subjettiness [14] with $N = 16$, and, finally, (iii) jet mass and multiplicity observables. The results are shown in Figure 2 and in Table 1.

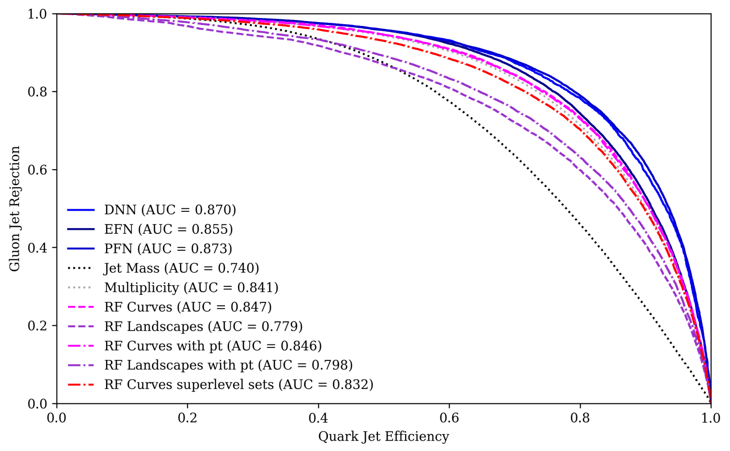

Figure 2: A comparison of ROC curves. Persistence landscapes perform slightly worse than Betti curves, whose performance approaches that of highly-parametrized models.

4 Conclusion

In terms of AUC, we see that the topological methods all beat the jet mass observable, with the RFs trained on Betti curves performing particularly well — comparable to the deep approaches. For the reasons given in Ketchum and Beretvas [5], we expect multiplicity to distinguish qg jets particularly well, but in terms of AUC, and for the 30% and 70% signal efficiency working points, the Betti curve RFs outperformed multiplicity. We emphasize the promising fact that these sparse models perform almost as well as the deeper models. The EFN, PFN, and DNN models have 56,630, 56,730, and 12,002 trainable parameters, respectively, far more than the random forests. We foresee many more possibilities using more sophisticated architectures for classification, and numerous applications for our method.

Our results, though promising, are still in their early stages. Perhaps most notably, we only applied our methods to the case of quark-gluon tagging. We attribute this to a general lack of open-source jet tagging data sets, especially those which include coordinate information in addition to kinematic information. It is possible that we would notice a difference in performance for other jet-tagging tasks. For example, our Lorentz stability argument hints at success for situations such as top-tagging. These jets are also richer in substructure, which further motivates the application of TDA methods. Nevertheless, we hope our work motivates the inclusion of topological techniques into jet tagging algorithms.

Broader Impacts

Our work is unique in that it is the first to implement a tagger based on jet topology. Though relatively simple in its current form, our results suggest that incorporating topological features of jets stands to improve tagging algorithms. Moreover, our work shows that this can be achieved with relatively simple classification architectures. In experiments such as those at the LHC, events occur at frequencies on the order of MHz, and a system of triggers selects the interesting events at a more reasonable rate. Clearly, time is of the essence in this process. If a topological method can be made faster than, for instance, a deep network forward pass, even small improvements can drastically increase the size of the statistics that are kept. This is crucial for all subsequent physics analysis. Moreover, topological methods are inherently robust to noise, which we suspect makes them resistant to effects such as pileup, which will be a critical obstacle in the High Luminosity LHC. We hope to explore these angles more precisely in our future work. The potential of topological methods in particle physics remains largely untapped, but our work represents one step in the right direction.

References

[1] Felix Hensel, Michael Moor, and Bastian Rieck. A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, 4, 2021. ISSN 2624-8212. doi: 10.3389/frai.2021.681108.

[2] Herbert Edelsbrunner and John Harer. *Computational topology: An introduction*. American Mathematical Society, Providence, RI, USA, 2010.

[3] Dominik J. E. Waibel, Scott Atwell, Matthias Meier, Carsten Marr, and Bastian Rieck. Capturing shape information with multi-scale topological loss terms for 3d reconstruction. In Linwei Wang, Qi Dou, P. Thomas Fletcher, Stefanie Speidel, and Shuo Li, editors, *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 150–159, Cham, Switzerland, 2022. Springer. doi: 10.1007/978-3-031-16440-8_15.

[4] Max Horn, Edward De Brouwer, Michael Moor, Yves Moreau, Bastian Rieck, and Karsten Borgwardt. Topological graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2022.

[5] Wesley Ketchum and Andy Beretvas. Gluon jets contain more color than quark jets, Jan 2014. URL `https://news.fnal.gov/2014/01/gluon-jets-contain-more-color-than-quark-jets/`.

[6] Gilles Louppe, Kyunghyun Cho, Cyril Becot, and Kyle Cranmer. QCD-aware recursive neural networks for jet physics. *Journal of High Energy Physics*, 2019(1), jan 2019. doi: 10.1007/jhep01(2019)057. URL `https://doi.org/10.1007%2Fjhep01%282019%29057`.

[7] Taoli Cheng. Recursive neural networks in quark/gluon tagging. *Computing and Software for Big Science*, 2(1), jun 2018. doi: 10.1007/s41781-018-0007-y. URL `https://doi.org/10.1007%2Fs41781-018-0007-y`.

[8] Shannon Egan, Wojciech Fedorko, Alison Lister, Jannicke Pearkes, and Colin Gay. Long short-term memory (lstm) networks with jet constituents for boosted top tagging at the lhc, 2017. URL `https://arxiv.org/abs/1711.09059`.

[9] Quark versus Gluon Jet Tagging Using Jet Images with the ATLAS Detector. Technical report, CERN, Geneva, 2017. URL `http://cds.cern.ch/record/2275641`. All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2017-017.

[10] Huilin Qu and Loukas Gouskos. Jet tagging via particle clouds. *Physical Review D*, 101(5), mar 2020. doi: 10.1103/physrevd.101.056019. URL `https://doi.org/10.1103%2Fphysrevd.101.056019`.

[11] Frédéric Chazal and Bertrand Michel. An introduction to topological data analysis: fundamental and practical aspects for data scientists, 2017. URL `https://arxiv.org/abs/1710.04019`.

[12] Lingfeng Li, Tao Liu, and Si-Jun Xu. Jet topology, 2020. URL `https://arxiv.org/abs/2006.12446`.

[13] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. Energy Flow Networks: Deep Sets for Particle Jets. *JHEP*, 01:121, 2019. doi: 10.1007/JHEP01(2019)121.

[14] Kaustuv Datta and Andrew Larkoski. How much information is in a jet? *Journal of High Energy Physics*, 2017(6), jun 2017. doi: 10.1007/jhep06(2017)073. URL `https://doi.org/10.1007%2Fjhep06%282017%29073`.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] Guillaume Tauzin, Umberto Lupo, Lewis Tunstall, Julian Burella Pérez, Matteo Caorsi, Anibal Medina-Mardones, Alberto Dassatti, and Kathryn Hess. giotto-tda: A topological data analysis toolkit for machine learning and data exploration, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] , See Section 1.
   (b) Did you describe the limitations of your work? [Yes] , See Section 4
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] , See section 3 and Appendix A.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] , See Section 3
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] , still to do.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] , See Appendix A.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] , See Section 3 and Appendix A.
   (b) Did you mention the license of the assets? [Yes] , See Appendix A.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] , Code is provided in Appendix A.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix A Selected Code and Implementation

In this appendix, we provide some python code for computing the topological features and training the random forests, which are implemented in scikit-learn [15] (BSD license). For the deep models used, we ran the examples available in the `EnergyFlow` Package of Komiske et al. [13] out-of-the-box. For topological computations, we used giotto-tda [16] (GNU AGPLv3 license). All code was run in Google Colab, with GPU enabled for training the deep models.

Appendix A.1 Computing Persistence Diagrams

```
from energyflow.datasets import qg_jets

from energyflow.utils import data_split
from energyflow.utils import to_categorical

from gtda.diagrams import BettiCurve, PersistenceLandscape
from gtda.homology import VietorisRipsPersistence

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, roc_auc_score, roc_curve

from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import HalvingGridSearchCV

import numpy as np
import matplotlib.pyplot as plt

for i in range(20):
    filepath = './.energyflow/datasets/QG_jets_{}.npz'.format(i)
    print("Loaded file ", filepath)

    if i==0:
      data = np.load('./.energyflow/datasets/QG_jets.npz')
    else:
      data = np.load(filepath)

    X = data['X']
    y = data['y']

    X = X[:, :, :3] #:3, drop PDG ID

    for x in X: # Center and normalize jets
        mask = x[:, 0] > 0
        yphi_avg = np.average(x[mask, 1:3], weights=x[mask, 0], axis=0)
        x[mask, 1:3] -= yphi_avg
        x[mask, 0] /= x[:, 0].sum()
    X = X[:, :, 1:3] #just y and phi
    X_drop_zeros = [x[~np.all(x == 0, axis=1)] for x in X] # drop zero padding
    del X
    X_drop_zeros_and_empty = []
    for x in X_drop_zeros:
      if x.shape[0] >= 1:
        X_drop_zeros_and_empty.append(x)
      else:
        # To avoid an error in the rare case the jet is empty
        X_drop_zeros_and_empty.append(np.array([[0., 0.]]))
    del X_drop_zeros

    print("computing diagrams for file ", filepath, "...")
    vr = VietorisRipsPersistence()
    diagrams = vr.fit_transform(X_drop_zeros_and_empty)
    print("done computing diagrams.")
    print("saving...")

    np.savez('<name_your_file_{}>'.format(i), diagrams=diagrams, y=y)
    print("saved diagrams to ./diagrams/diagrams_{}.npz".format(i))

    del diagrams
    del vr
```

Appendix A.2 Computing Representations of Diagrams

We include the code to process saved diagrams into Betti curves. The process is identical for persistence landscapes.

```
for i in range(20):
  filepath = '<path_to_dgms_{}>'.format(i)
  data = np.load(filepath)
  diagrams, y = data['diagrams'], data['y']
  print("Loaded file ", filepath)

  bc = BettiCurve(n_bins=100) # or landscapes
  print("Calculating Betti curves...")
  curves = bc.fit_transform(diagrams)
  print("done computing Betti curves.")

  del diagrams

  print('re-shaping...')
  curves = curves.reshape(curves.shape[0], -1)
  print('done re-shaping.')

  new_fname = '/path/curves_{}'.format(i)
  print("saving as ", new_fname, ".npz ...")
  np.savez(new_fname, curves=curves, y=y)
  print('done\n')

  del curves
```

Appendix A.3 Random Forest Training

First we optimize hyperparameters on the validation set, which we have loaded into curves_train, curves_val, and curves_test as described in Section 3.

```
# Set up possible values of parameters to optimize over
```

```
p_grid = {'max_depth': [3, 5, 10], 'min_samples_split': [10, 100, 1000],
             'n_estimators' : [1, 10, 50, 100]}


rf = RandomForestClassifier(n_jobs=-1)

# Score according to auc
clf = HalvingGridSearchCV(estimator=rf, param_grid=p_grid, cv=5,
             scoring='roc_auc', n_jobs=-1)
clf.fit(curves_val, y_val)

best_params = clf.best_params_
print(best_params)

## Returned {'max_depth': 10, 'min_samples_split': 100, 'n_estimators': 100}
## For both Betti curve and landscape RFs
```

Then, we fit the the test data with the optimal hyperparameters:

```
rf = RandomForestClassifier(max_depth=best_params['max_depth'],
                            min_samples_split=best_params['min_samples_split'],
                            n_estimators=best_params['n_estimators'],
                            n_jobs=-1)

rf.fit(curves_train, y_train)

y_pred = rf.predict(curves_test)
y_pred_probs = rf.predict_proba(curves_test)[:, 1]
print(classification_report(y_test, y_pred))

# get ROC curve
fp, tp, threshs = roc_curve(y_test, y_pred_probs)
auc = roc_auc_score(y_test, y_pred_probs)
print('\nRF AUC:', auc)
```