Classification of stochastic processes with topological data analysis

İsmail Güzel1,2 | Atabey Kaygun1

1Department of Mathematics Engineering, Istanbul Technical University, Istanbul, Turkey
2Network Technologies Department, TÜBİTAK ULAKBIM, Ankara, Turkey

Correspondence
İsmail Güzel, Department of Mathematics Engineering, Faculty of Science and Letters, Istanbul Technical University, Maslak, Istanbul, Turkey.
Email: iguzel@itu.edu.tr

Funding information
National Center for High-Performance Computing of Türkiye (UHeM), Grant/Award Number: 4010242021

Summary
In this study, we demonstrate that engineered topological features can distinguish time series sampled from different stochastic processes with different noise characteristics, in both balanced and unbalanced sampling schemes. We compare our classification results against the results of the same classification on features coming from descriptive statistics and the wavelet transform. We conclude that machine learning models built on engineered topological features alone perform consistently better than those built on the standard statistical and wavelet features for time series classification tasks. We also apply dimension reduction techniques to our engineered features and compare the result of our classification models before and after dimensionality reduction. Finally, we also show that in our calculations of the engineered topological features, employing parallel computing methods does yield significant improvements in run time and memory footprint.

KEYWORDS
Levy process, persistent homology, stochastic processes, time series classification, Wiener process

1 INTRODUCTION

In this study, we investigate if one can distinguish time series samples from different stochastic processes using topologically engineered features based on persistent homology. We chose to work with classification algorithms because they complement our use of persistent homology in the engineering of discrete topological features that have certain built-in transformation invariances (rotations, reflections, translations, and some small perturbations). Although we refer to these features as topological, this is a well-established misnomer in the literature as they are actually homotopical and discrete in nature. These features are designed to distinguish certain topological features, such as the number of connected components and circle- or sphere-like cavities within the data cloud we extracted from each time series we sampled. As such they are not well-suited for most optimization algorithms, or machine learning algorithms designed for forecasting continuous dependent variables for that matter, since such algorithms depend exclusively on the convexity, continuity, and/or smoothness of the transformed features. On the other hand, since the persistent homological features are discrete by design, they are particularly well-suited for classification and clustering algorithms. The main objective of this article is to demonstrate that persistent homological features are very effective in distinguishing time series samples taken from different families of stochastic processes compared to more traditional and some non-traditional features extracted from such data.

Persistent homology is a topological data analytic method. Topological data analysis (TDA) is a new field of data science that uses topological and geometric tools to infer relevant features from potentially complex data. TDA has developed procedures that are not based on traditional statistical or machine learning algorithms. These methods are now used in a variety of fields and proved themselves to be effective for the analysis of time series in a variety of domains even when the data is noisy. For example, TDA methods have been used in finance, in medicine, in quantifying periodicity in data, in clustering tasks, in classifying tasks, in detecting chaos and even in detecting early signals for critical transitions.
The main problem we tackle in this article is whether one can distinguish time series samples taken from different stochastic processes. The novel contribution of this article is the use of persistent homology to distinguish time series samples. To test our general methodology, we chose two specific subfamilies of Levy processes that have different Levy subordinators: (i) Wiener processes whose Levy subordinators have the scale parameter $\sqrt{t}$, and (ii) Cauchy processes whose Levy subordinators have the scale parameter $\frac{t^2}{2}$, over a time interval of length $t$. We tested our methods on synthetic data we sampled from these stochastic processes using both balanced and unbalanced sampling schemes. We show that features we engineer using persistence homology are extremely effective in distinguishing time series samples compared to standard statistical features, and features engineered using wavelet transforms in both sampling schemes.

The information that persistent homology yields on the change of topological features of a given point cloud can be presented in various different ways such as barcodes, persistence diagrams, landscapes, images, terraces, entropy, and curves. However, the common difficulty one encounters in TDA calculations is that persistent homology, in all of the representations we enumerated above, is computationally expensive. This fact renders all persistent homology calculations to be impractical without the help of state-of-the-art high-performance computing (HPC) tools, and libraries used for calculations often resort to parallel computing and/or off-loading heavy computations to GPUs to alleviate the problem. We also verify that parallelization does yield significant improvements in speed in our calculations.

Here is a short overview: First, in Section 2 we introduce our framework on persistent homology, stochastic processes, and wavelet transforms. Next, we describe the engineered statistical, wavelet, and topological features we are going to use in Section 3. We then apply various machine learning classification algorithms on the features we engineered before and after we apply dimensional reduction techniques, in both balanced and unbalanced sampling schemes in Section 4. We also discuss the effect of parallelization in our computations in the same section. Finally, we conclude and discuss potential future work in Section 5.

## 2 BACKGROUND

In this section, we briefly introduce the necessary background for the later sections.

### 2.1 Stochastic processes

Stochastic processes have been used to model systems that evolve probabilistically through time in a variety of fields such as finance, physics, and image-signal processing. In this article, we consider a family of stochastic processes called the Levy process with applications in insurance risk modeling, finance, and economics. A Levy process is a continuous-time stochastic process that has stationary independent increments with three main components: the deterministic component, the Brownian motion component, and a measure quantifying the rate at which discrete jumps occur. In this section, we describe two major examples of Levy processes.

#### 2.1.1 Wiener process

A standard Wiener process (often called Brownian motion) $X = (X_t, t \leq 0)$ for a random variable $X_t$ that depends continuously on $t \in [0, T]$ is a process with the state space $\mathbb{R}$ and the stationary independent increments such that $X_{t+s} - X_t$ has the normal distributions

$$X_{t+s} - X_t \sim \sqrt{t}N(0, 1),$$

where $N(0, 1)$ is the normal distribution with zero mean and unit variance with $X_0 = 0$.

#### 2.1.2 Cauchy process

A Cauchy process $X = (X_t, t \geq 0)$ is a process with the state space $\mathbb{R}$ and the stationary independent increments such that $X_{t+s} - X_t$ has the Cauchy probability density function

$$\rho_t(x) = \frac{t f(x)}{t^2 + x^2}.$$

In fact, Wiener processes are a special case of the Levy process family where the random variable has independent and identically distributed Gaussian increments with variance equal to increment length. Cauchy processes, on the other hand, are Brownian motions with a Levy subordinator.
FIGURE 1 The delay embedding of time series as $x$ variable of Rössler system. The delay parameter $\tau = 25$ is chosen by utilizing the mutual information while the embedding dimension $d$ is selected 2 because of visualization.

using the location parameter 0 and the scale parameter $t^2$ over the maximum time interval $t$. In other words, the difference between Wiener and Cauch processes comes from their Levy subordinators. We refer readers to Reference 26 (Sec. 1.3) for details about the Levy processes and their subordinators. In the sequel, we will examine if this difference can be better explained statistically or topologically.

2.2 | Takens’ delay embedding for time series

Time series do not naturally have point cloud representations. We are going to use the method of delay embeddings to convert a time series to a point cloud before we apply TDA techniques to capture the topological features of the underlying time series. We know that the delay embedding preserves topological features of a time series but not necessarily its geometrical properties due to Takens’ embedding theorem.27 See Figure 1 for such an example.

Given a time series $x_t$ for $t = 1, 2, \ldots, T$, we obtain a point cloud with points $v_i = (x_{t_i}, x_{t_i+\tau}, \ldots, x_{t_i+(d-1)\tau})^T$, where $\tau$ is the delay length, and $d$ is the dimension of the ambient space of the point cloud where the two arbitrary parameters $\tau$ and $d$ must be determined heuristically. The false nearest neighbor method28 and the first minimum of the mutual information of a time series and its time delay are commonly used methods in the literature to determine the embedding dimension $d$ and the delay parameter $\tau$, respectively.29-31 In the sequel, we will use the same methods to determine these parameters for our experiments.

2.3 | Topological data analysis

One of the main goals of algebraic topology is to classify topological spaces based on their features. In data science, on the other hand, extracting important features from large datasets to classify or cluster these extracted features appears as a recurrent problem. TDA aims to bridge topology and data science from this perspective.

2.3.1 | Persistent homology

One important tool that TDA employs in the process is persistent homology. Persistent homology offers an effective method for discovering evolutionary links within topological structures by investigating certain topological invariants of the data depending on a fixed natural number $n$. For instance, these invariants for $n = 0, 1, 2$ indicate the number of connected components, loops, and 2-dimensional voids in a given data set. Persistent homology relies on tracking such topological features through a filtration where we first construct a filtered simplicial complex from a given data, and then we record changes in the homology of the filtered complex. The main types of simplicial complexes used in typical computations are Alpha complexes, Čech complexes, or Vietoris-Rips complexes. However, in the sequel, we only pursue the use of Vietoris-Rips complexes. We refer interested readers to References 32 and 33 for an introduction to persistent homology and mathematical background on building topological invariants from point clouds.

Persistent homology yields promising results when applied to various machine learning and data analysis tasks. However, the fact that computations have exponential complexity in both space and time restricts their potential uses.34,35 Even after the recent computational improvements in various stages in calculating persistent homology of point clouds,36-38 there is still an urgent need for developing methods to reduce the memory
The persistence barcodes (left) and the corresponding persistence diagram (right). For example, the feature labeled as e is the interval (2, 3) on the barcodes and it becomes the point (2, 3) in the persistence diagram.

footprint and run-time complexity. One such avenue is parallelization where an effective use of parallel HPC may result in substantial reductions in space and time complexity per computation core. 16,35,39

2.3.2 Barcodes

Persistent homology of a point cloud is typically represented by a barcodes or a persistence diagram. (See Figure 2.) Each bar in the barcode corresponds to a homology class of the parameterized family of topological spaces obtained from the point cloud. The left endpoint of each bar corresponds to the parameter value at which the topological feature first appears, also referred to as the birth time $b_i$, and the right endpoint corresponds to the parameter value at which the topological feature disappears, referred to as the death time $d_i$. Thus, the bars in a barcodes can be represented as a collection of pairs $(b_i, d_i)$.

2.3.3 Persistence diagrams

For a given barcode as a pair of birth and death times, one can represent the pair as a point in $\mathbb{R}^2$. Thus, a persistence diagram $D$ becomes a multiset of points $D = \{(b_i, d_i)|i \in I\}$. Note that since $b_i \leq d_i$, for all $i$, all points in a persistence diagram lie on or above the line $y = x$.

2.3.4 Persistence landscapes

Given a persistence diagram $D = \{(b_i, d_i)|i \in I\}$, one can construct the corresponding persistence landscape as a sequence of piece-wise linear functions, $\lambda_1, \lambda_2, \ldots : \mathbb{R} \rightarrow \mathbb{R}$ with slopes 0, 1, or −1 where we let

$$\lambda_k(t) = k\max \left\{ f_{(b_i, d_i)}(t) \right\}_{i \in I},$$

where $k\max$ is the $k$th largest element and

$$f_{(a, b)}(t) = \max(0, \min(a + t, b - t)).$$

For example, let us consider the point cloud sampled from the two intertwined circles as in Figure 3. To investigate the topological features of that point cloud, one can consider the number of local maxima of the persistence landscapes or the area under the curve. As seen in Figure 3, there are two local maxima of the first persistence landscapes of a given point cloud.

2.3.5 Betti curves

Let $D = \{(b_i, d_i)|i \in I\}$ be a persistence diagram. The Betti curve of the persistence diagram $D$ is the function $\beta_D : \mathbb{R} \rightarrow \mathbb{N}$ whose value on $s \in \mathbb{R}$ is the number of points $(b_i, d_i)$ in $D$ such that $b_i \leq s < d_i$. More formally,

$$\beta_D(s) : \#\{(b_i, d_i) \in D|b_i \leq s < d_i\}.$$
FIGURE 3  The persistence diagram and corresponding first two persistence landscapes for 1-dimensional homology classes for the intertwined two circles.

FIGURE 4  From the persistence diagram to the corresponding Betti curve. For example, there are three orange pairs in the shaded region on the persistent diagram at $s_3$ while two orange pairs are at $s_4$.

FIGURE 5  The best matching on given two persistent diagrams $D_1$ and $D_2$, and their bottleneck distance. From left to right, the cost of matching is decreasing. Hence, the right figure has the minimum matching cost.

An example of the Betti curve of a persistence diagram is shown in Figure 4. The shaded gray region is the persistence diagram region containing pairs contributing $\beta_D(s)$.

### 2.3.6 The Wasserstein and the Bottleneck distances

Given two persistence diagrams $D_1$ and $D_2$, we have a bijection (a perfect matching) of the form $\varphi : D_1 \to D_2$ as seen in Figure 5. We include the points on the diagonals $\Delta = \{(s, s)|s \in \mathbb{R}\}$ to ensure such a perfect matching always exists. Then we define the Wasserstein distance\(^{11}\) of $D_1$ and $D_2$ to be

$$W_p(D_1, D_2) = \inf_{\varphi} \left( \sum_{x \in D_1} ||x - \varphi(x)||_p \right)^{1/p}.$$
where the infimum is taken over all perfect matchings. The Wasserstein distance is called the bottleneck distance if we let $p \to \infty$.

$$d_W(D_1, D_2) = \inf_{\phi} \sup_{x \in D_1} ||x - \phi(x)||_\infty.$$ 

### 2.4 Wavelet transform

The wavelet transform examines a timeseries at various scales to uncover its structure by using both time and frequency components. Given a timeseries $y(t)$, its wavelet transform defined as

$$WT(y, a, b) := \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} y(t) \cdot \psi \left( \frac{t - b}{a} \right) dt. \quad (1)$$

where $a$ is a scale parameter called dilation, $b$ is a translation parameter called the time shift, and $\psi(t)$ is called the mother wavelet function.

The integral transform given in Equation (1) analyzes the given time series through a wavelet function that is a confined short-time oscillation that resembles a wave. There is a whole collection of wavelets that are tailored for particular applications. The Fourier transform is an example of the Wavelet transform, but they differ significantly in a key way: while the Fourier representation of the signal only has a frequency component, the wavelet representations of the signal have both time and frequency components. As a result, it makes localized signal information easily accessible.

Continuous and discrete wavelet transforms (abbreviated CWT and DWT, respectively) are the two primary types of wavelet transformation. DWT uses a finite number of wavelets at a specific set of scales and locations, whereas CWT uses every wavelet over a range of scales and locations. As a result, DWT is simple and quick to employ. On the other hand, CWT can support high resolutions in the time-frequency domain but has higher time and space complexity. Since adopting CWT leads to more precise component identifications, in our analyses we are going to use CWT with the Morlet wavelet that is defined as

$$\psi(t) = \frac{1}{\sqrt{f_b}} \cos(2\pi f_c t) \exp(-t^2/f_b^2). \quad (2)$$

where $f_b$ is the bandwidth parameter, $f_c$ is the central wavelet frequency.

### 3 FEATURE ENGINEERING

Feature engineering plays an important role on classification and clustering tasks in many machine learning models. In this section, we outline the details of the engineered features our analyses are based on.

#### 3.1 Statistical features

Most common techniques of engineering features on time series use descriptive statistical features such as mean, variance, and entropy. For this study, we chose the following list of the statistical features:

1. Mean
2. Variance
3. Entropy
4. Lumpiness
5. Stability
6. Hurst
7. Std of 1st derivative
8. Linearity
9. Binarize mean
10. Unitroot KPSS
11. Histogram mode
12. Heterogeneity
13. Trend strength
3.2 Wavelet features

A univariate time series has a 2-dimensional wavelet representation called a scaleogram. This representation is obtained by applying a fixed wavelet function with varying dilation and time-shift parameters allowing for a multi-resolution analysis. By scaling the wavelet over a fixed time interval, we get information about the frequency domain. For instance, on a large scale (which corresponds to having a longer wavelet), we obtain information about slow-changing features visible in the low frequencies. Conversely, by having a smaller scale, we observe fast-changing features visible in the high frequencies. In other words, scales essentially detect features inversely correlated with the frequency. This frequency-time analysis enables us to understand how the transform varies with respect to frequency and phase, thus allowing for a concise expression and a simple analysis of the underlying time series. See Figure 6 for an example of a scaleogram that detects shifts in frequencies.

Obtaining a suitable scaleogram of a time series requires determining a suitable range of the scale parameter heuristically. For this purpose, we use the scale parameters changes exponentially as a power of 2 as described in Reference 53. Since the output of the CWT is a matrix, for the engineered wavelet features we record the column-wise and row-wise means and deviations. Hence, we obtain $4 \times M \times N$ wavelet features where $M$ is the number of scales parameter we sampled and $N$ is the number of time observations (the length of the signal).

3.3 Topological features

There are three popular indirect methods that use persistence diagrams along with machine learning algorithms.

(i) Kernel methods in conjunction with the distance metrics on persistence diagrams.

(ii) Extracting descriptive statistical features from the persistence diagram such as Adcock-Carlsson coordinates, the lifetime, the persistence entropy and so forth.

(iii) Embedding persistence diagrams into a Hilbert space using persistence landscapes, or Betti curves.

Taking the methods we enumerated above as our base, we engineered a total of 18 new topological features in homology dimensions 0 and 1. In this subsection, we present the extraction features from starting the persistence diagram. We refer the reader to References 34 and 54 for more details about the use of topological tools in machine learning tasks.
For this subsection, we fix a persistence diagram \( D = \{(b_i, d_i) | i \in I\} \) of a point cloud obtained from delay embedding of a time series with optimal parameters time delay \( \tau \) and dimension \( d \) as in described in Section 2.2.

### 3.3.1 Wasserstein and bottleneck distances

We can obtain a total of four new topological features by using Wasserstein and bottleneck distances between the given diagram \( D \) and the diagonal diagram \( \Delta \) for dimensions 0 and 1.

\[
W_1(D, \Delta) = \sum_{i \in I} \frac{d_i - b_i}{2} \quad \text{and} \quad d_B(D, \Delta) = \sup_{i \in I} \left( \frac{d_i - b_i}{2} \right).
\]

### 3.3.2 Adcock-Carlsson coordinates

Let \( d_{\text{max}} \) be the maximum death time over all pairs in the persistence diagram, and let \( \ell_i = d_i - b_i \) be the lifetime of the topological features. In Reference 55 the authors used the following summaries of a given persistence diagrams.

1. \( f_1(D) = \sum_{i \in I} b_i \ell_i \).
2. \( f_2(D) = \sum_{i \in I} (d_{\text{max}} - d_i) \ell_i \).
3. \( f_3(D) = \sum_{i \in I} b_i^2 \ell_i^2 \).
4. \( f_4(D) = \sum_{i \in I} (d_{\text{max}} - d_i)^2 \ell_i^2 \).

Using these features we obtain a total of 8 new features from persistence diagrams in dimensions 0 and 1.

### 3.3.3 Persistence entropy

The persistence entropy of the persistence diagram \( D \) is defined by

\[
E(D) = -\sum \ell_i \log \left( \frac{\ell_i}{L_D} \right),
\]

where \( L_D = \sum_{i \in I} \ell_i \). This adds two more topological features for our analyses for dimensions 0 and 1.

### 3.3.4 L1 norms

Let \( \beta_0 \) be the corresponding Betti curve, and let \( \lambda_1 \) be the corresponding first persistence landscapes. Since the Betti curve and the persistence landscapes are functions, we can calculate their \( L_p \) norms. See Figure 7. In practice, we simply use \( p = 1 \) for the features of a given persistence diagram and we obtain two new features using the norms \( \|\beta_0\|_1 \) and \( \|\lambda_1\|_1 \) for each dimension 0 and 1.

### 3.4 Dimensionality reduction

As we described above, we engineer a large number of statistical, wavelet, and topological features. To resolve potential time and space complexity issues, it is crucial to reduce the dimensionality of the engineered features, and more importantly, measure the effect of this reduction in the performance metrics of the machine learning algorithms we employ. For this purpose, we use principal component analysis on each class of engineered features: statistical, wavelet, and topological. We chose the reduction with the explained variance ratio at 95%. In this way, we take advantage of removing correlated features if we do not have any prior information about features that should be chosen.

### 4 EXPERIMENTAL CONTRIBUTIONS

All implementations are done using the python programming language and its library ecosystem. For sampling the stochastic processes, we used the \texttt{stochastic} package. We obtained the topological features and persistence diagrams using the \texttt{giotto} and the \texttt{giotto-ph} libraries.
We extracted the statistical features from time series using the Kats library. To engineer the wavelet features, we used the packages PyWavelets and scaleogram.

4.1 Results with balanced sampling

4.1.1 Simulations

We generated 1000 time series with the same length of 500-time steps for both Cauchy and Wiener processes on the time interval \([0, 2]\). So, in total, we have 2000 time series and their corresponding labels as “Cauchy” and “Wiener.” We then generated statistical features for each time series as described in Section 3.1. Next, we get the wavelet features by using the Morlet wavelet as described in Section 3.2, and the topological features as described in Section 3.3. For the topological features in this study, we use homology degrees 0 and 1 as opposed to using only degree 1 as we did in Reference.

4.1.2 Classification

After simulation and featurization, we can now apply machine learning models to our datasets that consist of statistical, wavelet, and topological features with labels. We used logistic regression (LGR), decision tree (DCT), k-nearest neighbor (KNN), random forest (RFT), support vector classifier (SVC), multi-layer perceptron (MLP), linear discriminant analysis (LDA) and XGBoost (XGB). We used python library sklearn and xgboost with default parameters. Before we start building models, we split our synthetic dataset into the train and the test datasets with sizes 80% and 20%, respectively. We also did 5-fold cross-validations for each model over the whole dataset. We display below the results for the consistently worst-performing model in our experiments which is KNN.

As one can see in Figure 8, the result shows that models built on topological features consistently perform better than the models built on statistical and wavelet features. The results of 5-fold cross-validation indicate that the most performant representative features, in order, are topological, statistical, and wavelet features, respectively.

When we restrict to statistical and wavelet features the SVC, LDA, and LGR models are relatively less performant compared to other models while the XGB model is the most performant on all features. On the other hand, it appears that the DCT model over-fit on the wavelet and statistical features for the train set. The confusion matrix for the KNN model for the balanced dataset, which is the worst-performing model for all features, is shown in Figures 9 and 10.

We use the receiver operating characteristic (ROC) curve to analyze classifier performances. The ROC curve depicts the trade-off between the true positive rate and the false positive rate, or sensitivity versus specificity, for different thresholds of the classifier output. Figures 11 shows the ROC curve for different features with the KNN algorithm. The area under the ROC curve (AUC) indicates the models based on topological features alone show better performance that the models based on the statistical and wavelet features alone.
The results for the machine learning classification models are based on the topological, statistical, and wavelet features, as well as results for each feature class obtained after applying the dimension reduction for the balanced dataset.

The confusion matrices from the wavelet, statistical features, and results after dimension reduction for the balanced dataset from the KNN algorithm.

The confusion matrices for the selected and for all topological features for the balanced dataset for the KNN algorithm.
4.2 Results with unbalanced sampling

Most supervised machine learning models require a balanced dataset for the training phase of model building and perform rather poorly if the dataset is unbalanced in favor of one class. In order to test the performance of our topologically engineered features, we repeated the same processes with an unbalanced sampling of the stochastic processes we are interested in. To show that our proposed approach performs well with unbalanced datasets, we generated 50 samples from a Cauchy process and 1000 observations from a Wiener process. After applying the same validation steps (single train/test and 5-fold cross-validation schemes), we still observe that the topological features consistently perform better even on unbalanced datasets. The results are shown in Figure 12. For the unbalanced dataset, the confusion matrix of the KNN algorithm on both features is shown in Figures 13 and 14. We chose the KNN algorithm because for balanced and unbalanced sampling experiments it gave us the worst accuracy results.

To check if the degree of unbalance has an effect, we sampled the stochastic processes with varying distributions of classes where the minority class varied between 1% and 20% of the majority class. In all experiments, the majority class has a sample size of 1000 observations. The resulting ROC curves are displayed in Figure 15.

Before we present our analyses, we must note that even though the confusion matrices we present in Figures 13 and 14 may be as helpful in assessing which models and which engineered features are preferment in the unbalanced sampling scheme as in the balanced scheme, the areas under the ROC curves we present in Figure 15 are better metrics than the accuracy scores we present in Figure 12 for the sake of completeness. Both the confusion matrices and the area under the ROC curves for the worst-performing KNN model in the unbalanced scheme collectively indicate that the models based on the proposed topological features still perform better than the models based on the statistical and wavelet features.
4.3 Computation details

Engineering topological features require working with persistence diagrams, and constructing a persistence diagram from a point cloud is computationally expensive due to the high computational cost of persistent homology. To make the calculations, we used the Turkish National Center for High-Performance Computing (UHeM) at Istanbul Technical University. We used the *giotto-ph* library to compute the topological features in parallel. All the computations are performed on a Centos 7 Linux UHeM cluster with 128 GB RAM, Intel(R) Xeon(R) E5-2680 CPU 2.40GHz with 28 and 128 cores in a node, respectively.

To compute the computational cost of our approaches, we sampled 50-time series from a Cauchy process with randomly varying lengths between 500 and 1500. For each time series, we individually calculated the topological features using both serial and parallel computation because of their high complexity, while the statistical and wavelet features are computed in the serial case because of low complexity. We ran 7 experiments and recorded the time required to compute topological, wavelet, and statistical features for all of the 50-time series. The results are presented in Table 1.
TABLE 1  Comparison of computation time.

| Features  | Mean  | Std  |
|-----------|-------|------|
| Wavelet   | 6.62 s | 35 ms|
| Statistics| 10.2 s | 28.6 ms|
| TDA extended | 112 s | 1.93 s|
|           | 28 cores | 43.8 s | 138 ms|
|           | 128 cores | 36.4 s | 156 ms|

The results indicate that parallelization dramatically reduces computation time, but the topological features are computationally more expensive than the statistical and wavelet features even after parallelization. Despite their time complexity issues, one must emphasize that the topological features consistently yield much better performance in classification tasks even on unbalanced data sets. The computational bottleneck in calculating the topological features is calculating persistence diagrams from a point cloud. Fortunately, the Python library giotto-ph automatically parallelizes the calculations.

5  CONCLUSION

We used three different classes of engineered features (statistical, wavelet, and topological) to classify time series sampled from different stochastic processes. In our simulations, we sampled times series from Wiener and Cauchy processes in both balanced and unbalanced sampling schemes. We then compared machine learning classification models built on different classes of features we engineered on the sampled time series. The results show that the engineered topological features perform consistently better than statistical or wavelet features in building machine learning classification models even when a given data set are unbalanced. Our experimental results show that the topologically engineered features alone can distinguish between different stochastic processes, even when statistical or wavelet features do not. This means that topological feature engineering can play a critical role in time series classification tasks.

Even though the result indicates that the topological features show the best results, the fact that the theoretical computational complexity is exponential requires that one must employ state-of-the-art HPC tools in their calculations in applications. We also showed that parallelization significantly speeds up computations.

5.1  Future work

In the classification of time series obtained from Wiener and Cauchy processes, our experiments indicate that topological features perform consistently better than statistical and wavelet features for building machine learning models even when classes are not balanced. The main difference between the stochastic processes we used in this study is their Levy subordinators. We showed that the differences between time series samples from different stochastic processes can be explained by topological features. Whether topological features can distinguish different Levy subordinators is an intriguing research question one might pursue, but unfortunately, is beyond the scope of this article. Other possibilities are repeating our experiments on real-time series datasets with different noise characteristics, and comparing our methods with methods such as the DWT, the discrete Fourier transforms, and the power spectral densities. One may also focus on distances between topological features to apply clustering algorithms such as hierarchical clustering, and density-based clustering methods.

ACKNOWLEDGMENTS

Computing resources used in this work were provided by the National Center for High-Performance Computing of Türkiye (UHeM) under the first author’s project which Grants number 4010242021. We also thank the two anonymous reviewers whose comments helped to improve the quality of this study.

CONFLICT OF INTEREST STATEMENT

The authors declare that there is no conflict of interest.
DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

İsmail Güzel https://orcid.org/0000-0002-5888-7025

REFERENCES

1. Majumdar S, Laha AK. Clustering and classification of time series using topological data analysis with applications to finance. Expert Syst Appl. 2020;162:113868.

2. Nanda V, Saadanyič R. Simplicial models and topological inference in biological systems. In: Jonoska N, Saito M, eds. Discrete and Topological Models in Molecular Biology. Springer; 2014:109-141.

3. Perea J, Harer J. Sliding windows and persistence: An application of topological methods to signal analysis. Found Comput Math. 2015;15(3):799-838.

4. Seversky LM, Davis S, Berger M. On time-series topological data analysis: New data and opportunities. Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Los Alamitos, CA; 2016:1014-1022; IEEE Computer Society.

5. Umeya T. Time series classification via topological data analysis. Inf Media Technol. 2017;12:228-239.

6. Güzel I, Munch E, Khasawneh FA. Detecting bifurcations in dynamical systems with CROCKER plots. Chaos. 2022;32(9):093111.

7. Gidea M. Topological data analysis of critical transitions in financial networks. In: Shmueli E, Barzel B, Puzis R, eds. 3rd International Winter School and Conference on Network Science. Springer; 2017:47-59.

8. Gidea M, Katz Y. Topological data analysis of financial time series: Landscapes of crashes. Physica A. 2018;491:820-834.

9. Ghrist R. Barcodes: The persistent topology of data. Bull New Ser Am Math Soc. 2008;45(1):61-75.

10. Cohen-Steiner D, Edelsbrunner H, Harer J. Stability of persistence diagrams. Discret Comput Geom. 2007;37(1):103-120.

11. Bubenik P. Statistical topological data analysis using persistence landscapes. J Mach Learn Res. 2015;16(1):77-102.

12. Adams H, Emerson T, Kirby M, et al. Persistence images: A stable vector representation of persistent homology. Adv Comput Math. 2015;42(1):1-42.

13. Chung YM, Lawson A. Persistence curves: A canonical framework for summarizing persistence diagrams. Adv Comput Math. 2022;48(1):1-42.

14. Merelli E, Rucco M, Sloot P, Tesei L. Topological characterization of complex systems: Using persistent entropy. Entropy. 2015;17(10):6872-6892.

15. Chung YM, Lawson A. Persistence images: A stable vector representation of persistent homology. Adv Comput Math. 2015;42(1):1-42.

16. Pérez JB, Hauke S, Lupo U, Caorsi M, Dassatti A. giotto-ph: A Python library for high-performance computation of persistent homology of Vietoris-Rips filtrations. arXiv preprint arXiv:2107.05412, 2021.

17. Zhang S, Xiao M, Wang H. GPU-accelerated computation of Vietoris-Rips persistence barcodes. In: Cabello S, Chen DZ, eds. 36th International Symposium on Computational Geometry (SoCG 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik; 2020.

18. Kijima M. Stochastic Processes for Physicists: Understanding Noisy Systems. Cambridge University Press; 2010.

19. Won CS, Gray RM. Stochastic Image Processing. Springer Science & Business Media; 2004.

20. Klüppelberg C, Kyprianou AE, Maller RA. Ruin probabilities and overshoots for general Lévy insurance risk processes. Ann Appl Probab. 2004;14(4):1766-1801.

21. Klüppelberg C, Kyprianou AE, Maller RA. Ruin probabilities and overshoots for general Lévy insurance risk processes. Ann Appl Probab. 2004;14(4):1766-1801.

22. Landriault D, Li B, Kocabas MA. On the analysis of deep drawdowns for the Lévy insurance risk model. Insur Math Econ. 2021;100:147-155.

23. Schoutens W. Lévy Processes in Finance: Pricing Financial Derivatives. Wiley Online Library; 2003.

24. Tour G, Thakoor N, Khaliq A, Tangman D. COS method for option pricing under a regime-switching model with time-changed Lévy processes. Quant Financ. 2018;18(4):673-692.

25. Geman H. Pure jump Lévy processes for asset price modelling. J Bank Financ. 2002;26(7):1297-1316.

26. Applebaum D. Lévy Processes and Stochastic Calculus. Cambridge University Press; 2009.

27. Takens F. Detecting strange attractors in turbulence. In: Rand D, Young LS, eds. Dynamical Systems and Turbulence, Warwick 1980. Springer; 1981:366-381.

28. Kennedy MB, Brown R, Abarbanel HDI. Determining embedding dimension for phase-space reconstruction using a geometrical construction. Phys Rev A. 1992;45:3403-3411.

29. Tsang P. An Exploration of Topological Properties of High-Frequency One-Dimensional Financial Time Series Data Using TDA. PhD thesis. KTH Royal Institute of Technology; 2017.

30. Khasawneh FA, Munch E. Chatter detection in turning using persistent homology. Mech Syst Signal Process. 2016;70:527-541.

31. Pereira CM, de Mello RF. Persistent homology for time series and spatial data clustering. Expert Syst Appl. 2015;42(15-16):6026-6038.

32. Carlsson G. Topology and data. Bull New Ser Am Math Soc. 2009;46(2):255-308.

33. Edelsbrunner H, Harer J. Computational Topology: An Introduction. American Mathematical Soc; 2010.

34. Otter N, Porter MA, Tillmann U, Grindrod P, Harrington HA. A road map for the computation of persistent homology. Epj Data Sci. 2017;6:1-38.

35. Malott N, Chen S, Wilsey P. A survey on the high performance computation of persistent homology. IEEE Trans Knowl Data Eng. 2022;35(5):4466-4484.

36. Zhang X, Zhang Q. NASA Public Access. ALEMS Geosci. 2017;32(2):163-186.

37. Morozov D, Nigmetov A. Towards lockfreeness: persistent homology. Proceedings of the Annual ACM Symposium on Parallelism in Algorithms and Architectures 2020: 555-557.

38. Boissonnat JD, Durrant J. Edge collapse and persistence of flag complexes. Leibniz International Proceedings in Informatics, (LIPIcs), vol 164; 2020:1-15.

39. Bauer U. Ripser: Efficient computation of Vietoris-Rips persistence barcodes. J Appl Comput Topol. 2021;5:391-423.

40. Bubenik P. The persistence landscape and some of its properties. In: Baas N, Carlsson G, Quick G, Thaule M, eds. Topological Data Analysis, Abel Symposia. Springer International Publishing; 2020:97-117.

41. Kerber M, Morozov D, Nigmetov A. Geometry helps to compare persistence diagrams. ACM J Exp Algorithmics. 2017;22:1-20.

42. Allen RL, Mills D. Signal Analysis: Time, Frequency, Scale, and Structure. John Wiley & Sons; 2004.

43. Daubechies I. Ten Lectures on Wavelets. SIAM; 1992.

44. Stéphane M. A Wavelet Tour of Signal Processing. 3rd ed. Academic Press; 2009.

45. Guido RC. Wavelets behind the scenes: Practical aspects, insights, and perspectives. Phys Rep. 2022;985:1-23.
46. Heil CE, Walnut DF. Continuous and discrete wavelet transforms. SIAM Rev. 1989;31(4):628-666.
47. Jorgensen PET, Song MS. Comparison of discrete and continuous wavelet transforms. In: Meyers R, ed. Encyclopedia of Complexity and Systems Science. Springer; 2009:1163-1177.
48. Grossmann A, Morlet J. Decomposition of Hardy functions into square integrable wavelets of constant shape. SIAM J Math Anal. 1984;15(4):723-736.
49. Strang G, Nguyen T. Wavelets and Filter Banks. SIAM; 1996.
50. Jiang Y, Tang B, Qin Y, Liu W. Feature extraction method of wind turbine based on adaptive Morlet wavelet and SVD. Renew Energy. 2011;36(8):2146-2153.
51. Fulcher BD. Feature-based time-series analysis. In: Dong G, Liu H, eds. Feature Engineering for Machine Learning and Data Analytics. CRC Press; 2018:87-116.
52. Byeon YH, Pan SB, Kwak KC. Intelligent deep models based on scalograms of electrocardiogram signals for biometrics. Sensors. 2019;19(4):935.
53. Torrence C, Compo GP. A practical guide to wavelet analysis. Bull Am Meteorol Soc. 1998;79(1):61-78.
54. Chazal F, Michel B. An introduction to topological data analysis: Fundamental and practical aspects for data scientists. Front Artif Intell. 2021;4:667963.
55. Adcock A, Carlsson E, Carlsson G. The ring of algebraic functions on persistence bar codes. Homol Homotopy Appl. 2016;18(1):381-402.
56. Van Rossum G, Drake FL. Python 3 Reference Manual. CreateSpace; 2009.
57. Flynn C. Stochastic; 2021. https://github.com/crflynn/stochastic
58. Tauzin G, Lupo U, Tunstall L, et al. giotto-tda: A topological data analysis toolkit for machine learning and data exploration. arXiv preprint arXiv:2004.02551; 2020.
59. Facebook. Kats; 2021. https://github.com/facebookresearch/kats
60. Lee GR, Gommers R, Waselewski F, Wohlfahrt K, O’Leary A. PyWavelets: A Python package for wavelet analysis. J Open Source Softw. 2019;4(36):1237.
61. Sauve A. Scaleogram; 2018. https://github.com/alsauve/scaleogram
62. Güzel İ, Kaygun A. Classification of stochastic processes with topological data analysis. BAŞARIM, İstanbul, Türkiye; 2022.
63. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res. 2011;12:2825-2830.
64. Chen T, Guestrin C. Xgboost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016:785-794.

How to cite this article: Güzel İ, Kaygun A. Classification of stochastic processes with topological data analysis. Concurrency Computat Pract Exper. 2023;35(24):e7732. doi: 10.1002/cpe.7732