International Journal of Data Science and Analytics (2024) 17:305–321
https://doi.org/10.1007/s41060-022-00368-3

REGULAR PAPER

Classification of orbits in Poincaré maps using machine learning

Chandrika Kamath[1]

Received: 1 July 2022 / Accepted: 23 October 2022 / Published online: 16 November 2022

Abstract
Poincaré plots, also called Poincaré maps, are used by plasma physicists to understand the behavior of magnetically confined plasma in numerical simulations of a tokamak. These plots are created by the intersection of field lines with a two-dimensional poloidal plane that is perpendicular to the axis of the torus representing the tokamak. A plot is composed of multiple orbits, each created by a different field line as it goes around the torus. Each orbit can have one of four distinct shapes, or classes, that indicate changes in the topology of the magnetic fields confining the plasma. Given the (x, y) coordinates of the points that form an orbit, we want to assign a class to the orbit, a task that appears ideally suited for a machine learning approach. In this paper, we describe how we overcame two major challenges in solving this problem—creating a high-quality training set, with few mislabeled orbits, and converting the coordinates of the points into features that are discriminating, despite the variation within the orbits of a class and the apparent similarities between orbits of different classes. Our automated approach is not only more objective and accurate than visual classification, but is also less tedious, making it easier for plasma physicists to analyze the topology of magnetic fields from numerical simulations of the tokamak.

Keywords Poincaré maps · Magnetic fusion · Orbits · Classification · Machine learning

1 Introduction

The quest for low-cost fusion power has led to the construction of experimental devices such as the DIII-D [1], an operational device for conducting magnetic fusion research, and ITER [2], an international project to help make the transition from studies of plasma physics to electricity-generating fusion power plants. These devices, called tokamaks, use magnetic fields to confine the fusion fuel in the form of a plasma, enabling physicists to perform experiments to determine the best shape for the hot reacting plasma and the magnetic fields necessary to hold it in place. To complement the experiments, computer simulations are used to gain an understanding of the complex physics of the plasmas, design new reactors, and select the parameters to be used in experiments. Data from both the experiments and the simulations are analyzed to provide the insights that will contribute to achieving the goal of fusion power.

In this paper, we focus on a specific analysis problem that arises in both simulation and experimental data, namely the classification of orbits in a Poincaré map, also called a Poincaré plot. These two-dimensional plots are obtained for planes, called poloidal planes, which intersect the torus-shaped tokamak perpendicular to the magnetic axis, as shown in Fig. 1a. A plot consists of several orbits, each composed of a number of points (Fig. 1b). For a given orbit, these points are the intersections of a field line (the solid lines in Fig. 1a) with a poloidal plane, as the field line is followed around the torus. The four distinct shapes traced out by these points correspond to four classes of orbits: quasiperiodic, separatrix, island chain, and stochastic, as shown in Fig. 2. An orbit may show its distinctive shape with just a few points, corresponding to the first few intersections of the field line with the poloidal plane. Or, an orbit may appear to be of one class, say island chain, initially, but become a separatrix as additional points are added to the orbit. When the data are generated by a computer simulation, floating point errors can build up as the field lines are traced around the torus, resulting in noise in the location of the later points.

These Poincaré plots provide a convenient way to diagnose, at a glance, changes in the topology of the magnetic fields confining the plasma over time. The goal is to avoid undesirable changes that would result in the hot core of the plasma escaping to mix with the cooler outer regions.

✉ Chandrika Kamath
kamath2@llnl.gov

[1] Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, CA 94551, USA

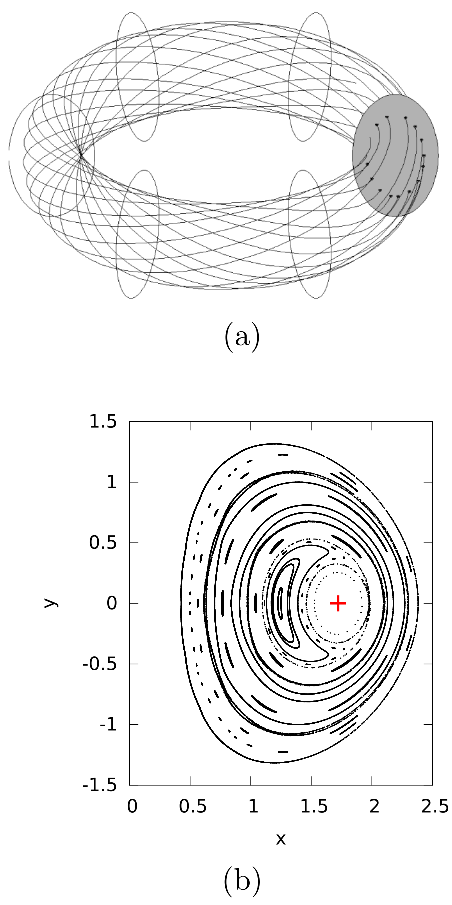

Fig. 1 **a** Schematic view of a tokamak, in the form of a torus, with six poloidal planes shown as ellipses. As a field line goes around the torus, its intersections with a poloidal plane form an orbit, as shown on the gray ellipse. **b** A collection of orbits in a poloidal plane forms a Poincaré plot. The magnetic axis is indicated by a red cross (colour figure online)

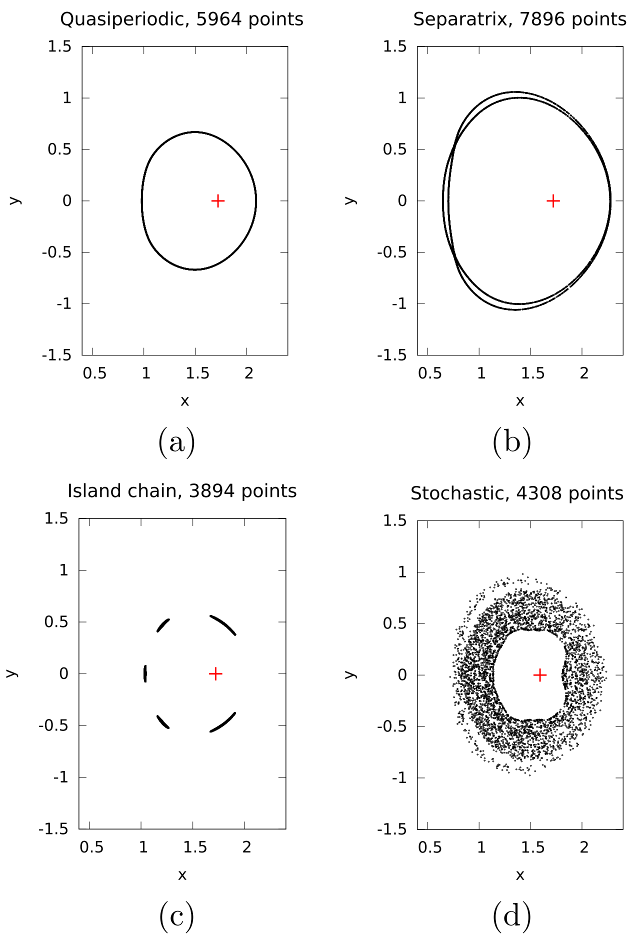

Fig. 2 Sample orbits illustrating the four classes—**a** a quasiperiodic orbit; **b** a separatrix orbit with three lobes, where a lobe is the region between the 'X'-points where the two curves of the separatrix cross; **c** an island orbit with five islands; and **d** a stochastic orbit. The number of points in the orbits ranges from 4000 to 8000

Whether the hot plasma is confined or not is reflected in the shape of the orbits: quasiperiodic orbits are indicative of nested magnetic surfaces which provide good confinement; a separatrix indicates reconnection, or rapid changes in the topology; while islands and stochastic regions indicate progressively worse confinement.

Classifying the orbits visually is tedious, error-prone, and subjective. When the number of orbits is large, as a result of many simulations run over many time steps, an automated approach to classification becomes essential. An obvious solution is to create a training set by extracting features representing each orbit, followed by a machine learning classifier for prediction. However, implementing this solution is not straightforward due to challenges in labeling the orbits and in identifying relevant features for orbits of different classes (Sect. 2). Following a description of prior work (Sect. 3), we describe how we address these challenges (Sect. 4) and discuss the results of our experiments with i) decision trees for predicting the class of an orbit, and ii) feature selection for understanding which features are more relevant in discriminating among the classes (Sect. 5). We conclude the paper with a summary of our work in Sect. 6.

Our contributions in this paper are as follows: for this rather unusual data set, where each orbit is a collection of points in two-dimensions, we describe how we can transform the (x, y) coordinates of these points into a representation of the visual structure of the orbit. To generate a high-quality training data set, we use simple, but interpretable, classifiers, and iteratively refine both the assignment of correct class labels to the orbits and the extraction of representative features for each orbit. This approach allows us to capture the subtle differences between orbits of different classes that appear similar when viewed as a whole, as well as the similarities within orbits of a class despite the variation in their shape. As a result, we achieve higher accuracy than that obtained using visual classification. By automating the step of classification of the orbits in a Poincaré plot, we provide plasma physicists an improved capability to understand their simulations.

2 Challenges to the analysis

At first glance, this problem of classification of the orbits appears relatively straightforward as the four orbits shown in Fig. 2 have distinctive characteristics. The quasiperiodic orbit appears as points on a single closed curve; the separatrix appears as two intertwined curves; the island chain has angular gaps; and there is no structure to the points in a stochastic orbit. Therefore, by assigning a class label to each orbit and representing it with a suitable set of features, we can create a training data set for use in building a machine learning model. However, our initial attempts at implementing a solution (Sect. 4) indicated two main challenges, which we discuss next.

2.1 Generating a training set

To generate the training set, we started by visually assigning one of the four class labels to each orbit. Further, since we had a limited set of orbits (four sets of 66 orbits each), many with several thousands of points, we created a series of *derived* orbits from each orbit by using only the initial intersections of the field line with the poloidal plane. Thus, for an orbit of a specific class defined by m points, $m > 1000$, we created separate derived orbits by using the first 1000, 1500, 2000, ... points and assigned them the same class. The lower bound on k was chosen as we required a certain minimum number of points in an orbit to extract meaningful features.

However, as discussed in Sect. 4.1, we accidentally found in our initial work that the class of the derived orbits could vary as the number of points was increased. For example, in Fig. 3a, an orbit appears as a quasiperiodic when we plot the first 500 points. At 1000 points, we see a handful of points forming an inner curve, points that could be easily overlooked in a visual labeling of the orbit. This inner curve takes shape only when many more points are added, clearly indicating a separatrix orbit in panels (c) and (d). A similar example is shown in Fig. 4, where the class of the orbit changes from island chain to separatrix to mildly stochastic as the number of points in the orbit is increased. This meant that each derived orbit had to be visually inspected to assign a class label; we could not simply assign them all to the same class.

We also observed that some orbits, especially the separatrix, have some stochasticity, especially for a large number of points. This results from the way in which the field lines are traced along the torus. A true stochastic orbit has the points spread out over a large radial distance as in Fig. 2d.

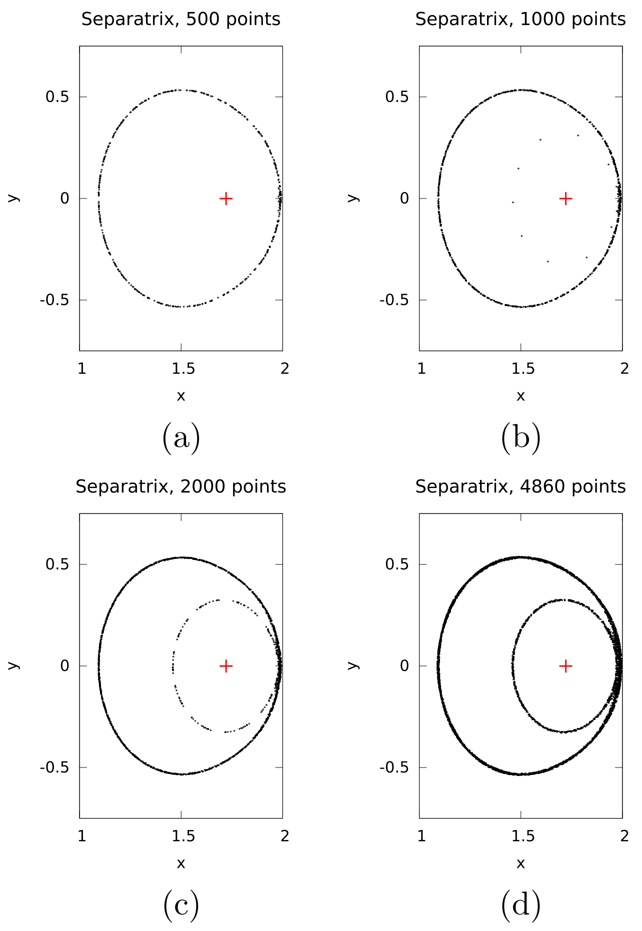

Fig. 3 A separatrix orbit as the number of points is increased from **a** 500, showing a quasiperiodic to **b** 1000 points, which is a separatrix with the inner curve made up of very few points, to a clear separatrix at **c** 2000 and finally, **d** 4860 points

Figure 4 also illustrates the second challenge in labeling orbits—this orbit, regardless of the number of points, appears to be quasiperiodic when viewed as a whole; we need to look at the detail to determine the correct class. Figure 5 shows other examples where a visual inspection of the full orbit is insufficient to assign the correct class label. The top panel appears to be a single island in the form of a crescent, but is actually an island composed of thin islands. In the middle and bottom panels, we see that a quasiperiodic orbit and an incomplete quasiperiodic orbit are actually a separatrix and an island chain, respectively, where the lobes of the separatrix and the islands are very thin, with a very small radial variation. Based on these examples, the orbit in Fig. 6 could be either an incomplete quasiperiodic or an island chain; as more points are added, and we view the fine-scale details in the right column, we find this orbit is quasiperiodic, with a very small amount of stochasticity when the number of points is large.

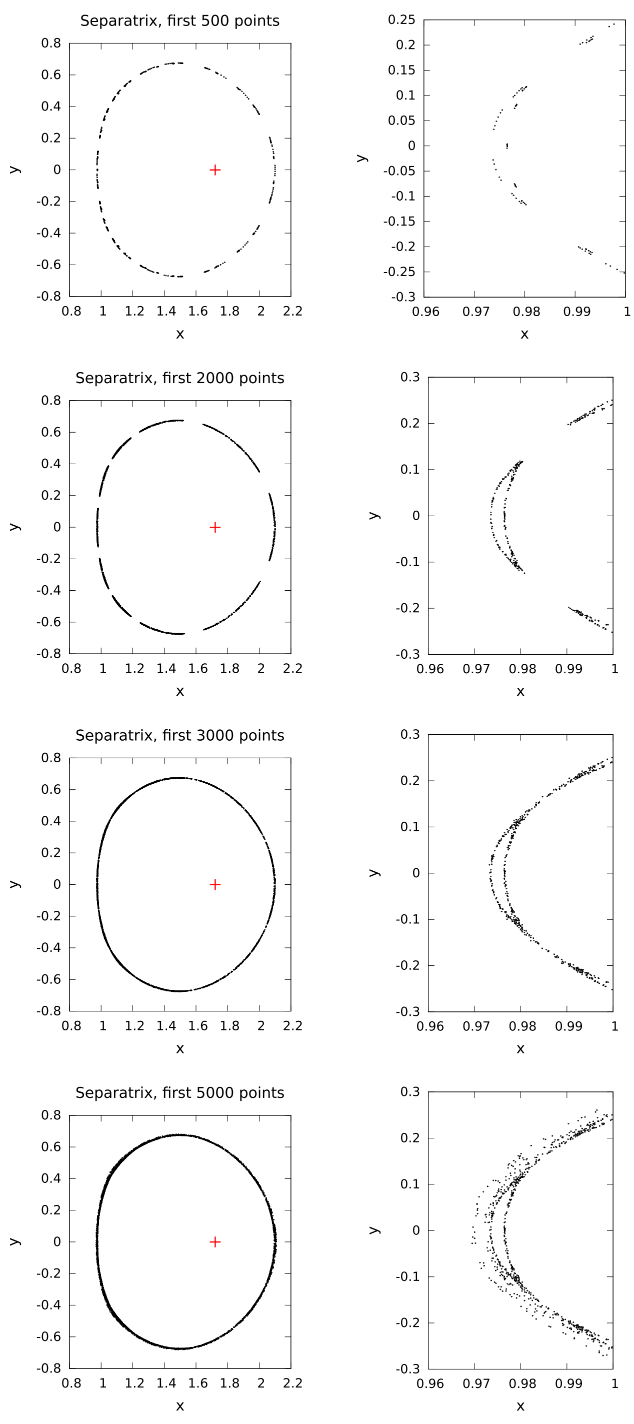

Fig. 4 A separatrix orbit is shown using the first (from top to bottom) 500, 2000, 3000, and 5000 points. The orbits appear to be quasiperiodic at a coarse scale (left column), but the fine-scale detail of the left part of the orbit (right column) indicates that with increasing number of points, the class changes from island chain, to separatrix, to mildly stochastic

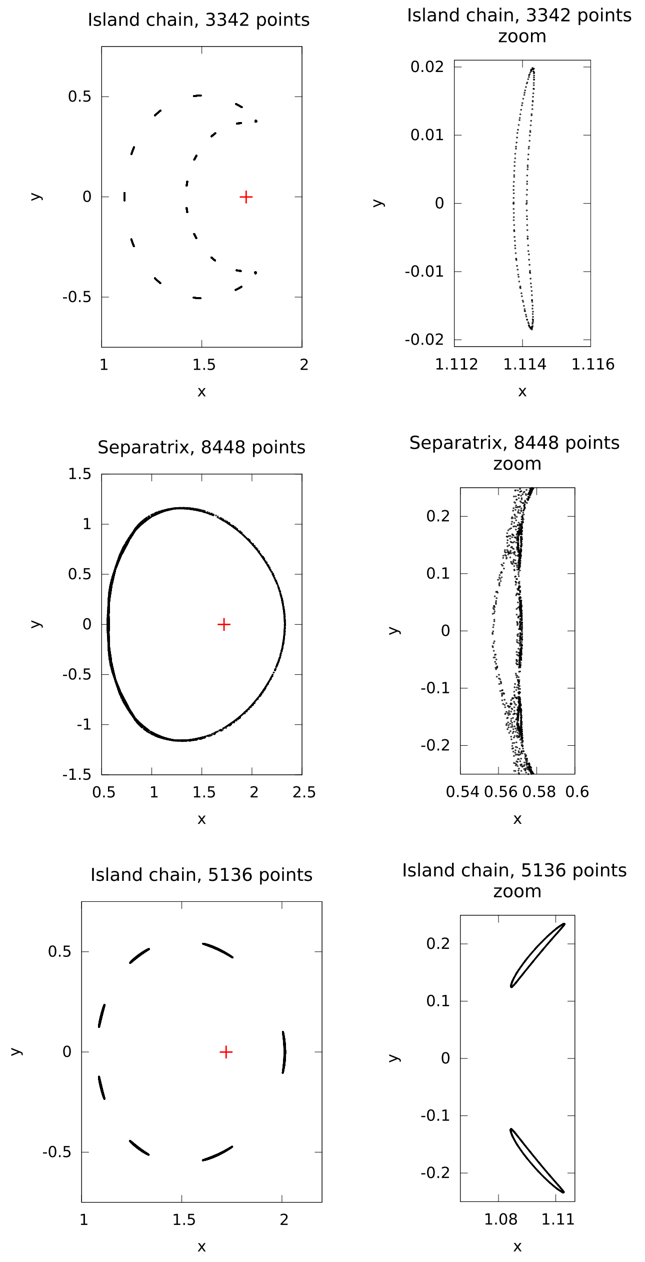

Fig. 5 Thin island chains and narrow separatrix orbits are difficult to classify visually, requiring a detailed view (right column) for correct classification. Top: a single island in the form of a crescent is composed of several segments that are islands themselves. Middle: a very thin separatrix that appears to be a quasiperiodic orbit. Bottom: an island chain with seven islands that appears as an incomplete quasiperiodic orbit

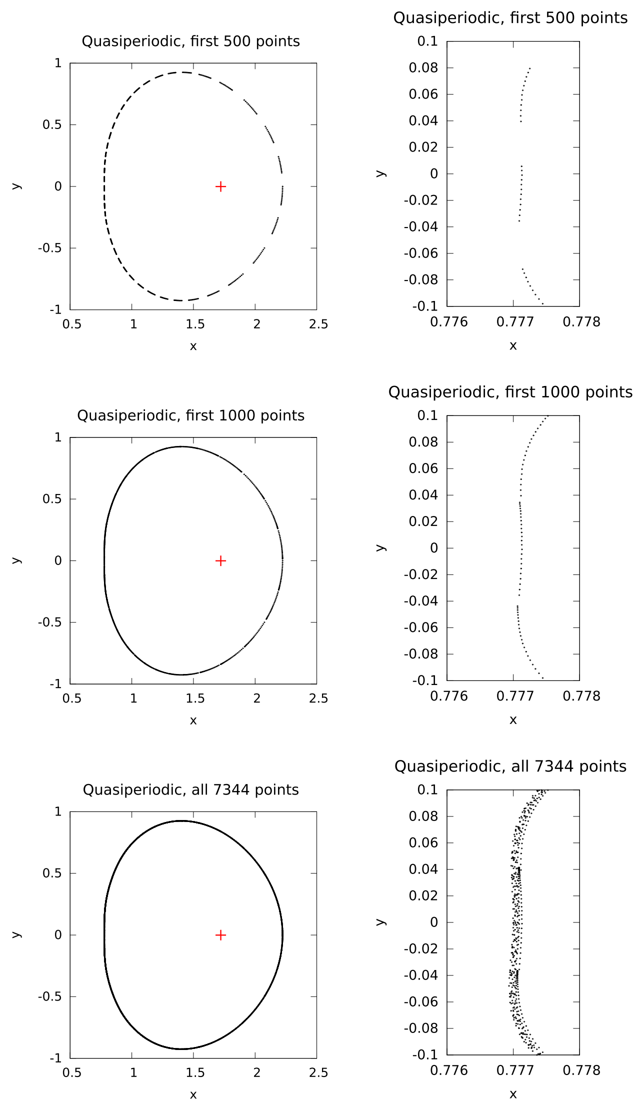

Fig. 6 A quasiperiodic orbit that appears as a thin island chain at 500 points (top) is confirmed to be a quasiperiodic when more points are added (middle and bottom) and details observed (right column). The mild stochasticity at 7344 points has a very small radial range of values, unlike a stochastic orbit

This multi-scale nature of some of the orbits, where an orbit appears to be of one class when viewed at a coarse scale, but is of a different class when viewed at a finer scale (that is, a zoomed-in view), further makes it challenging to label the orbits correctly. We can no longer rely solely on a visual inspection of the entire orbit to assign a class; requiring a closer inspection of the points makes the class assignment even more tedious. This provides a strong motivation for our work, and, as we show later in Sect. 4.1, we can use the process of automating the assignment of labels to bootstrap the correct assignment of class labels.

2.2 Generating features for an orbit

The second challenge in orbit classification is the identification and extraction of representative features for each orbit. Obviously, the features selected must be discriminating so they can be used to differentiate among the orbits of different classes. However, as we have seen, the orbits within a class can appear quite different, while orbits from different classes may appear very similar. As a result, obvious features, such as the angular gaps that characterize the island orbits, are not sufficient to differentiate them from incomplete quasiperiodic orbits, while a small radial variation of the points within a small angular window could indicate a quasiperiodic, a thin island chain, or a thin separatrix. Other considerations that influence the definition and extraction of representative features include:

- The data, available as (x, y) coordinates of the points in an orbit, must be converted to represent the structure we see when we visualize the orbit, either as a whole, or a zoomed-in view.
- These features must be invariant to scale, rotation, and translation, as the class of the orbit does not change if it is scaled, rotated, or translated.
- The features must be robust to noise as small changes in the coordinates of the points in an orbit do not change the orbit class.
- The features must not depend on the number of points in the orbit, or the radial distance of the orbit from the magnetic axis.

Transforming the (x, y) coordinates of an orbit into a representation of the visual structure is what makes this problem unusual and challenging.

3 Related work

We next briefly review existing work related to the classification of orbits in Poincaré maps to place our work in context. This topic is of interest to several communities, including plasma physics [3] and spacecraft trajectory design [4]. A common theme in these classical physics systems, where the dynamics is described in terms of Hamilton's equations, is identifying regular and chaotic regions in phase space. This specific problem has been addressed using box counting to determine the fractal dimension of a set of points [5] and by calculating the spectrum of Lyapunov exponents using fast algorithms, such as the structure-preserving Gaussian process surrogate [6].

The analysis of Poincaré maps in magnetic confinement fusion, where they are referred to as puncture plots, has been driven by two classes of methods. The visualization community has focused on topology-based methods, for example, using field lines with near minimal lengths to determine the topology of a magnetic field [7] and a six-stage approach, specifically tailored to the data, that is combined with visualization of scalar maps to provide context to the topological visualization [8]. These analyses have included island chain and separatrix orbits, in addition to quasiperiodic and chaotic orbits.

In contrast, the data mining community has taken a feature-based approach. Yip [9] represented an orbit as a minimal spanning tree, which was split into distinct subgraphs by removing edge lengths greater than a specified threshold. Next, features, such as the diameter of the tree, the number of subgraphs, and properties of branches that shared a single node with the diameter were obtained for the full tree and the subgraphs. These features were then used in rules to identify the class of an orbit.

In our previous work [10,11], we found that we could improve the approach of Yip on our data sets by including additional features and using machine learning classifiers. While we obtained nearly 90% accuracy rate for classification, our approach was not robust enough to be applied in practice. Accurate classification required a large number of points (2000–2500) in each orbit, and it was challenging to set the values of the many thresholds used in defining the features. In addition, the graph structure was difficult to implement in software, depended on the number of points in an orbit, and was prone to giving incorrect results if any of the key points determining the structure were perturbed slightly. This present work is an attempt to address these deficiencies.

4 Solution approach

Our three-step approach to classifying the orbits includes (i) labeling the orbits; (ii) generation of features for each orbit for use in a training data set; and (iii) building a classification model with the training set to discriminate among the different classes. We iterated among the steps to improve the accuracy of the model, as shown in the schematic in Fig. 7. We used decision trees for classification so we could understand how the features were used to assign a specific class label to an orbit. As we show next, this enables us to create a high-quality training data set by ensuring that the label assigned to an orbit is correct and the features used to describe it are discriminating.

4.1 Labeling the orbits

We started by using a visual inspection to assign each orbit to one of the four classes. As discussed earlier, we increased the size of our data set by creating derived orbits using the

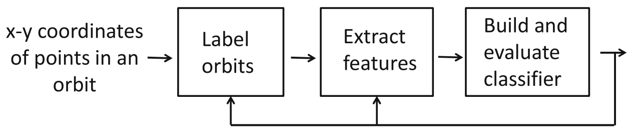
x-y coordinates of points in an orbit
Label orbits
Extract features
Build and evaluate classifier

Fig. 7 The three-step solution approach includes labeling the orbits and extracting the features to create a training set that is used to build and evaluate the classification model. The accuracy of the model may prompt the labels to be corrected or improved features to be extracted for each orbit

first k points in an orbit, with $k = 1000, 1500, \ldots$. Initially, we assigned each derived orbit the same class as the original orbit with all points. Next, we created a training set by extracting obvious features for the orbits, such as angular gaps and the radial spread of points, and built a decision tree model. We then found that we needed to improve the robustness of the features extracted as detailed in Sect. 4.2.3. As we iteratively improved the quality of the features, the prediction error reduced.

Eventually, we reached a point where we were unable to reduce the error any further. A closer look at the classification results indicated that the trees often mislabeled quasiperiodic orbits with gaps as island chains, or ones without gaps as separatrix orbits. On following the path taken by the features for these orbits through the decision tree, we found that at the leaf node, all training instances were of the same class as the one predicted for the orbit. This indicated that the orbit being labeled was in a region of the feature space where the class was different from the one we had assigned. We next checked whether we had correctly extracted the features for the orbit, which required a close inspection of each orbit. We then realized that the features extracted were correct, but, as described in Sect. 2.1, either the orbits had a multiscale nature and had been assigned a wrong class, or that the class of the derived orbit, with fewer points, was different from that of the orbit with all the points.

This accidental discovery indicated that we had to look closely at the details of each orbit to assign the correct class. To make this task less tedious, we used the classification results to focus only on the orbits with incorrect predictions to determine if we had assigned an incorrect label. This approach allowed us to bootstrap our way to a higher-quality training data set. Note that by including the derived orbits, we make the labeling of the orbits more tedious, but we increase the diversity of the instances in the training data set, resulting in a model that is less sensitive to small changes in the location of the points that form an orbit.

4.2 Feature extraction

The identification and extraction of discriminating features was performed iteratively—we selected a few sample orbits from each class, extracted a set of features, and analyzed them

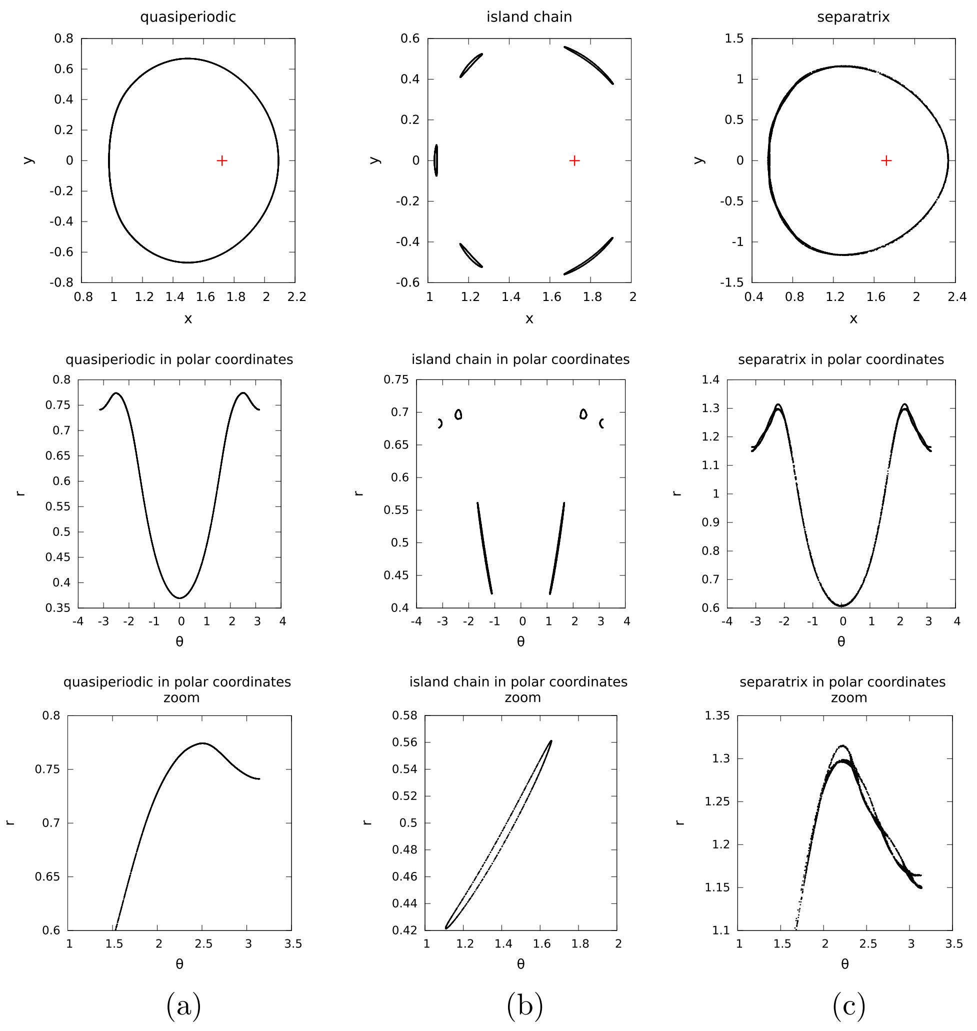

Fig. 8 The orbits in polar coordinates, with the angle, θ, on the x-axis and the radius, r, on the y-axis. **a** The quasiperiodic orbit from Fig. 2a; **b** the island chain orbit from Fig. 2c; and **c** the separatrix orbit from Fig. 5b. Top row: (x, y) coordinates, middle row: (r, θ) coordinates, and bottom row: a zoomed-in view of the polar coordinates that highlights the difference between the quasiperiodic and thin separatrix orbits

to determine if they were representative enough to account for the variation we observed among the orbits. As the quality of the features improved, we used them to build a decision tree classifier and evaluated its accuracy. If certain types of orbits were consistently misclassified by the model, we followed the features for such orbits through the decision tree, identified the aspect of the orbit that was missing in the existing features, and refined the features, continuing the process until sufficient accuracy was obtained.

The feature extraction consists of three phases—the initial preprocessing of the data, the extraction of local features representing the local region around the points, and the calculation of global features representing the orbit as a whole. We next discuss these phases in detail.

4.2.1 Preprocessing the data

It is possible to use the original (x, y) coordinates of the points to extract the features. However, we found that converting the data to polar coordinates, using the magnetic axis as the origin, exaggerated the radial variation in an orbit, making it easier to differentiate among the orbits. For example, Fig. 8, using (r, θ) coordinates, clearly shows the difference between the quasiperiodic orbit from Fig. 2a and the very thin separatrix orbit from Fig. 5b.

4.2.2 Extraction of initial local features

Once we converted the data into polar coordinates, we wanted to extract *local* features for the orbit, that is, features that represent the structure of the points in a small θ window. We started by defining these windows based on the θ value of the points in an orbit. If we created n windows, each subtending an angle $\delta_n = 360^\circ/n$ or $\delta_n = 2\pi/n$ radians at the magnetic axis, the points in the i-th window would satisfy

$$-\pi + (i-1)\delta_n \leq \theta < -\pi + i\delta_n.$$

Each point was assigned to one and only one window, and all orbits have the same number of windows, n.

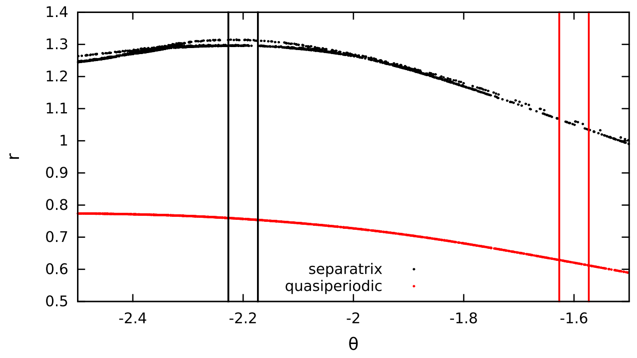

Fig. 9 A subset of two orbits—a separatrix orbit (top, in black) and a quasiperiodic orbit (bottom, in red)—in polar (r, θ) coordinates. The change in r over a 3° window centered at -2.2 for the separatrix is similar to the change in r over a 3° window centered at -1.6 for the quasiperiodic orbit. Thus, the range of r values in a window does not indicate whether the points in the window can be fit by one curve or two (colour figure online)

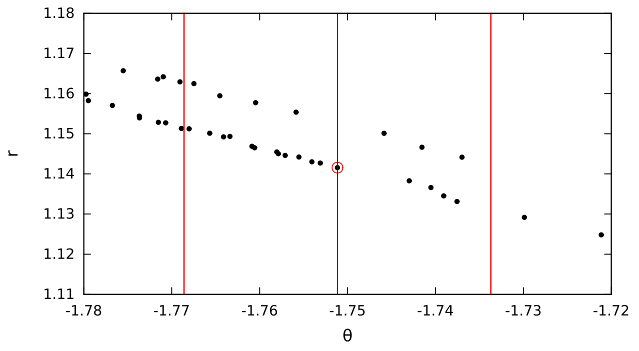

Fig. 10 A zoomed-in view of a separatrix orbit showing a window centered at the point with $\theta = 1.75115$, indicated by the blue line. The two red lines on either side indicate the boundaries of a window which is $2° = 0.0349$ radians wide. All points in this window are used to generate the features associated with the central point (colour figure online)

Next, for the points in each window, we obtained r_{max} and r_{min}, which are the largest and smallest values of r, respectively, and calculated $\Delta r_{max} = \max(r_{max} - r_{min})$, the largest radial width across all windows. The points in each window were then shifted radially by r_{min} so that the minimum r in each window became zero, and scaled by Δr_{max}, so that the new largest radial width across all the windows in an orbit became 1.0. This scaling normalized the radial variation in the points within the windows across all orbits, so that separatrix and island orbits with wide lobes were treated similar to those with thin lobes.

After the normalization, we extracted local features for each window, such as the variation in the r values of the points in a window and the number of points in a window. These local features for each of the n windows were converted to features for the entire orbit by taking the mean, minimum, and maximum values of the features across the windows. We also included global features, such as the number of windows with no points.

However, our early experiments indicated two problems with this approach to extracting representative features. First, the features extracted depended on the angular width of the window, δ_n, and were not rotation invariant. Thus features, such as the maximum density of points across all the windows, could vary substantially if we just rotated the orbit. Second, using just the range of r values within a window was insufficient to determine whether the points in a window lay on one curve, as in a quasiperiodic orbit, or two, as in a separatrix or island chain orbits. As shown in Fig. 9, the variation in r in a 3° window is similar for a separatrix (top curve, window centered at $\theta = -2.2$ radians) and a quasiperiodic (bottom curve, window centered at $\theta = -1.6$ radians). This holds whether we defined the variation as the difference between the largest and smallest r for the points in the window or the maximum difference between r values of consecutive points in the window.

4.2.3 Extraction of robust local features

These observations prompted us to define more robust local features. To create rotation-invariant features, we re-defined a window to be centered around *each* of the m points in an orbit, as shown in Fig. 10. Thus, instead of a fixed number, n, of non-overlapping windows for all orbits, we now had a variable number, m, of overlapping windows, each centered at a point and subtending an angle of δ, which we set to 2°. We continued to use the normalization of the r values as explained earlier so the features are not influenced by the distance of the orbit from the magnetic axis or the radial width of island and separatrix lobes.

Next, to determine if the points in a window lay on one curve or two, we obtained a least squares fit of a second order polynomial to the points in a window and calculated the maximum error across all points in this window, which we refer to as *lserr*, or the least square error for the window. This error is small for quasiperiodic orbits, but large for the other classes, especially for windows where there is a radial spread in the spatial distribution of the points.

As the least squares fit, and other features explained next, require a certain minimum number of points within a window, we only consider windows with greater than four points in the rest of the discussion. The points associated with such windows are referred to as *valid* points.

Next, to differentiate the random scattering of the points in a stochastic orbit from the more organized distribution of points on two curves seen in the island chain and separatrix orbits, we considered a feature derived from quadrat counts in spatial statistics [12]. A quadrat method divides a region into non-overlapping subsets, often rectangular in shape and then counts the number of points in each subset. These counts can then be used to determine any spatial pattern in the data. In our problem, we used a 4×4 grid in (r, θ) over each window, and since we have relatively few points in each window, we used the number of nonzero cells in the grid to represent the "spread" of the points. We expected that if a window has a larger spread, it will have a more stochastic distribution of points.

We calculated two additional features to represent the distribution of points across the orbit. The first was, *conc*, the concentration of points for each valid window, defined as the number of points in the window, scaled by the number of points in the orbit, so orbits with larger number of points do not result in biased features. The second feature was the difference in the θ values between a point and the point to its left, which we refer to as *ldtheta*. Note that we need to consider all points for *ldtheta*, not just the valid points (that is, those with more than four points in the window) because points with large *ldtheta* are often in windows with few points.

4.2.4 Calculation of derived features

Thus far, we have considered an angular window in θ around each point in the orbit and calculated quantities that characterize the distribution of the points in the window. These local features are calculated for windows with more than 4 points and, for each such window, include *lserr*, which is the maximum error in fitting a second order polynomial to the points; the spread of the points based on quadrat counts; and *conc*, the concentration of points in the window, which is the scaled number of points. In addition, the feature *ldtheta* at each point represented angular gaps in the orbit.

To convert these vector-valued, local features into scalar, global features that can represent the orbit as a whole, we calculated the mean, maximum, minimum, and standard deviation of the local features across all the valid windows in an orbit. In addition, for *ldtheta*, we defined a new binary feature, *ldtheta_large* that was set to 1 if an orbit had any value of *ldtheta* greater than 10°, and 0 otherwise.

However, these global quantities did not capture the subtle variations among the orbit classes, such as changes in the local features as we traversed along the orbit or any patterns when two of the local features were considered together. For example, a closer look at the separatrix orbits indicated that the *conc* value in windows near an X point is higher than in windows near the middle of a lobe, while the *lserr* value is higher near the middle of the lobe and tapers off near the X points. A similar behavior is seen between the center and the two corners of each island in an island chain.

This observation prompted us to consider the *lserr* and *conc* features at each window together. Figures 11 and 12, top row, show the sample orbit of each class from Fig. 2. The second row shows the values of *lserr* and *conc* features, in red and black, respectively, where the x axis indicates all the valid points in the orbit, going counter clockwise from $-\pi$ to π radians. We observe that the quasiperiodic orbit has much smaller values of *lserr*, which is expected. Also, as expected, we see clear peaks and valleys in *lserr* and *conc* for the island chain and separatrix orbits, though surprisingly, similar peaks and valleys are also present for the quasiperiodic and stochastic orbits, but at smaller magnitudes.

Figures 11 and 12, third row, show the *lserr* and *conc* features after both have been scaled to lie between [0, 1] and after minimal smoothing with a mean filter of width 3. We make two main observations on these plots. First, the number of peaks is equal to the number of lobes in the separatrix or the number of islands in the island chain, but we cannot relate the number of peaks to any obvious structures in the quasiperiodic and stochastic orbits. Second, there is a clear pattern in the locations of the peaks and valleys for *lserr* and *conc*. Stochastic orbits have the peaks and valleys of the two curves aligned, both in magnitude and location. In island chain, the peaks and valleys are of different magnitude, with the peak of one aligned with the valley of the other, and vice versa. For the separatrix orbit, the magnitudes of the two features are different, with the valleys in *lserr* aligned with the peaks in *conc*. In the quasiperiodic orbit, the two features have similar magnitudes and their peaks and valleys are aligned, with the latter more so than the former.

Finally, the last row in Figs. 11 and 12 shows the values of *ldtheta* for each of the orbits—note the high peaks for the island chain that are large enough to completely dwarf the values in between the peaks. We used these plots to select the threshold for the *ldtheta_large* feature.

To translate these observations on *lserr* and *conc* into scalar features, we need to identify the peaks and valleys in these variables and evaluate their alignment. However, this is challenging as both curves are quite noisy despite the minimal smoothing. To further reduce the high-frequency components, we applied wavelet smoothing using the biorth 3, 1 wavelet [13,14], which is a symmetric wavelet with a small support. However, we could not use a fixed number of wavelet levels in the smoothing as the length of the two curves varies across orbits. Instead, we applied the decimated wavelet transform until the number of points in the smooth, low-frequency part of the data was in the range [250, 500) for the fine level, and less than 100 for the coarse level, as shown in Figs. 13 and 14 , first and second rows, respectively. We chose two levels to identify the peak-valley alignment in *lserr* and *conc* in case the fine level still had some remaining noise, as seen in some of the orbits.

Next, we used a simple peak finder, and its converse as a valley finder, to identify the peaks and valleys in the coarse and fine resolutions of the two vectors. At each resolution, given the total number of peaks and valleys in *lserr* (or *conc*), we identified the fraction of the peaks and valleys of *lserr* aligned with the peaks and valleys, respectively, of *conc*, and

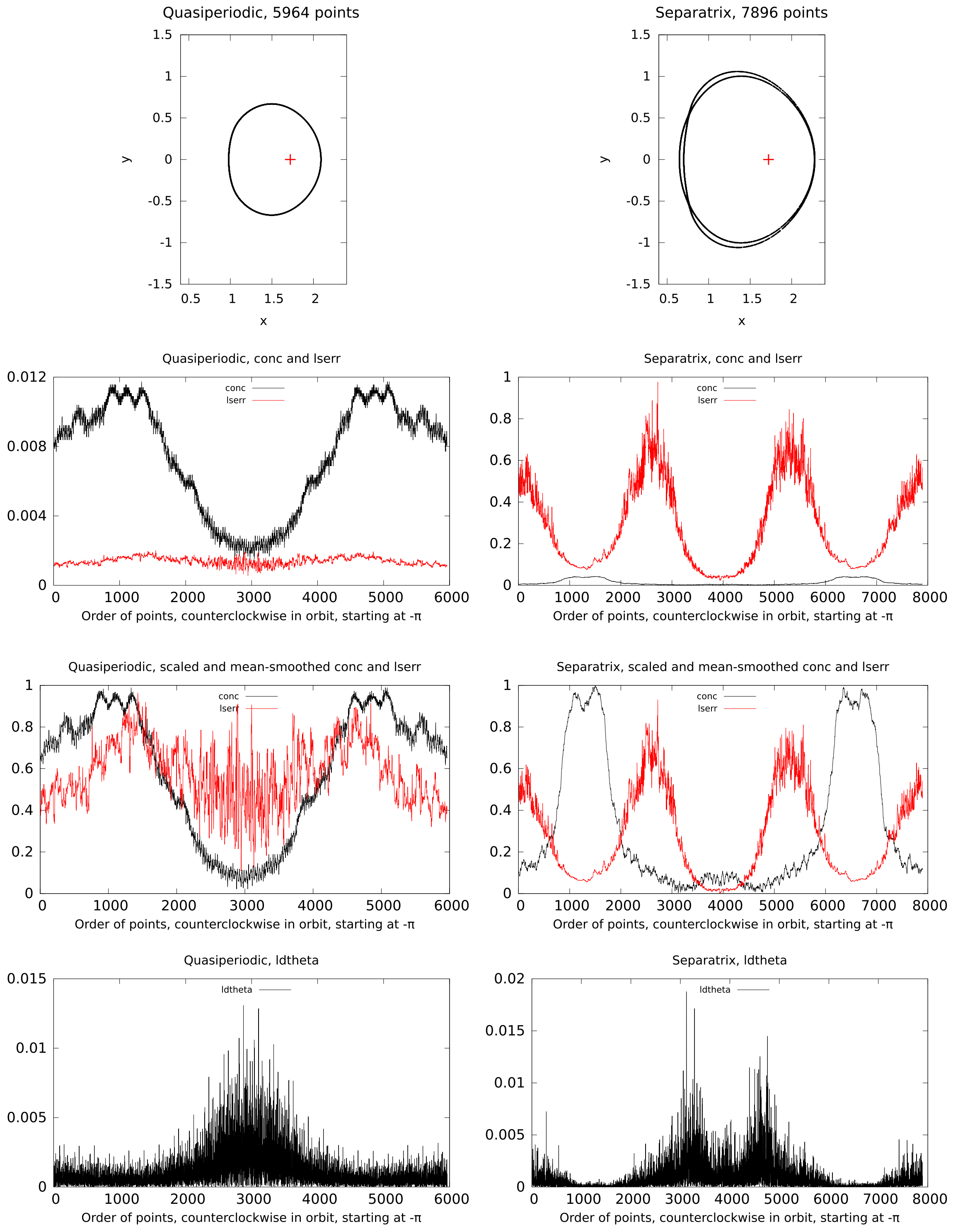

Fig. 11 The *lserr* and *conc* features, in red and black, respectively, for the quasiperiodic (left) and separatrix (right) orbits from Fig. 2. Top row: original orbits. Second row: the *lserr* and *conc* features in the window around each valid point. Third row: the features scaled to [0, 1] and smoothed using a simple mean filter of width 3. Bottom row: value of the *ldtheta* variable. The points in rows two to four are ordered by value of θ, going counterclockwise from $-\pi$ to π radians

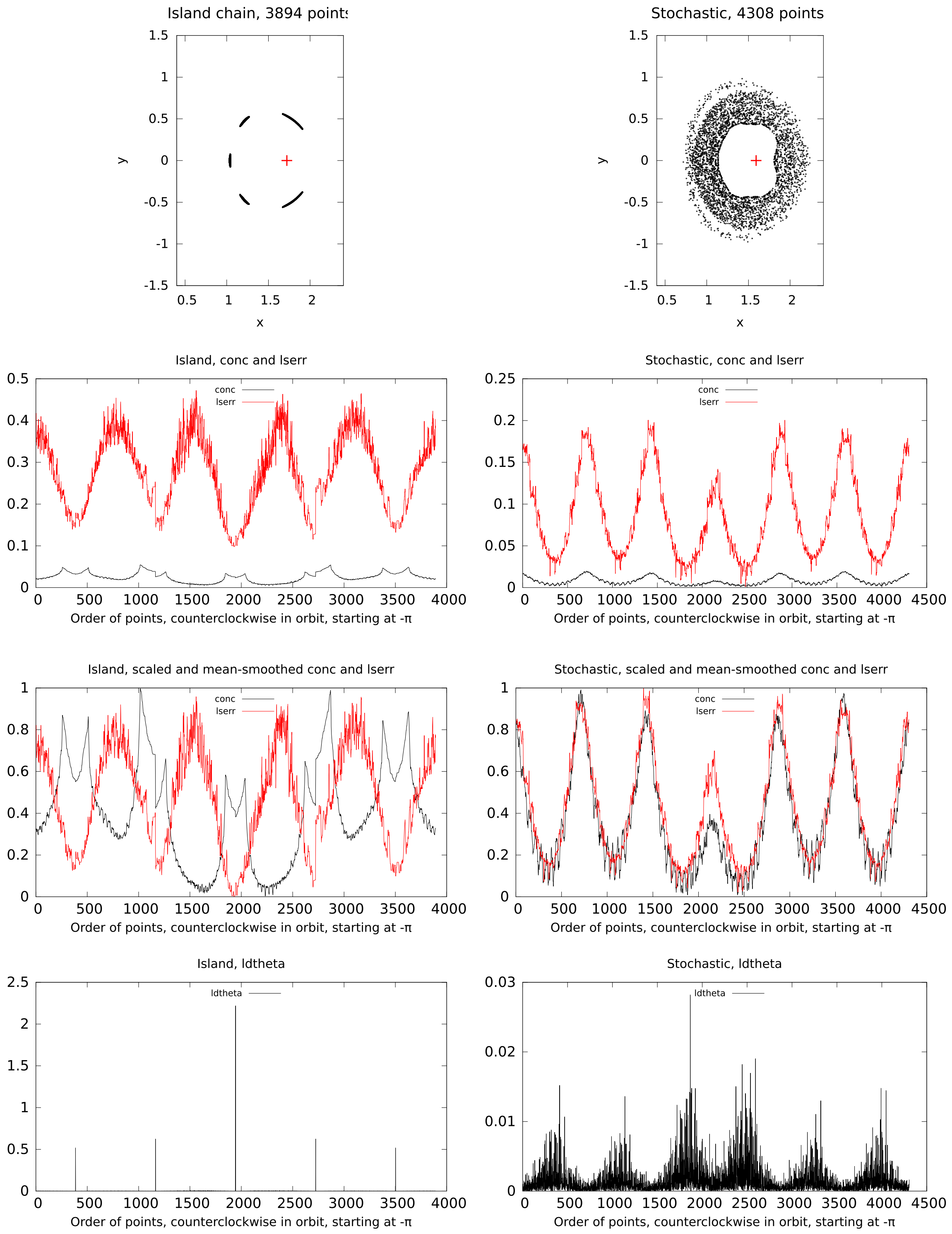

Fig. 12 The *lserr* and *conc* features, in red and black, respectively, for the island chain (left) and stochastic (right) orbits from Fig. 2. Top row: original orbits. Second row: the *lserr* and *conc* features in the window around each valid point. Third row: the features scaled to [0, 1] and smoothed using a simple mean filter of width 3. Bottom row: value of the *ldtheta* variable. The points in rows two to four are ordered by value of θ, going counterclockwise from $-\pi$ to π radians (colour figure online)

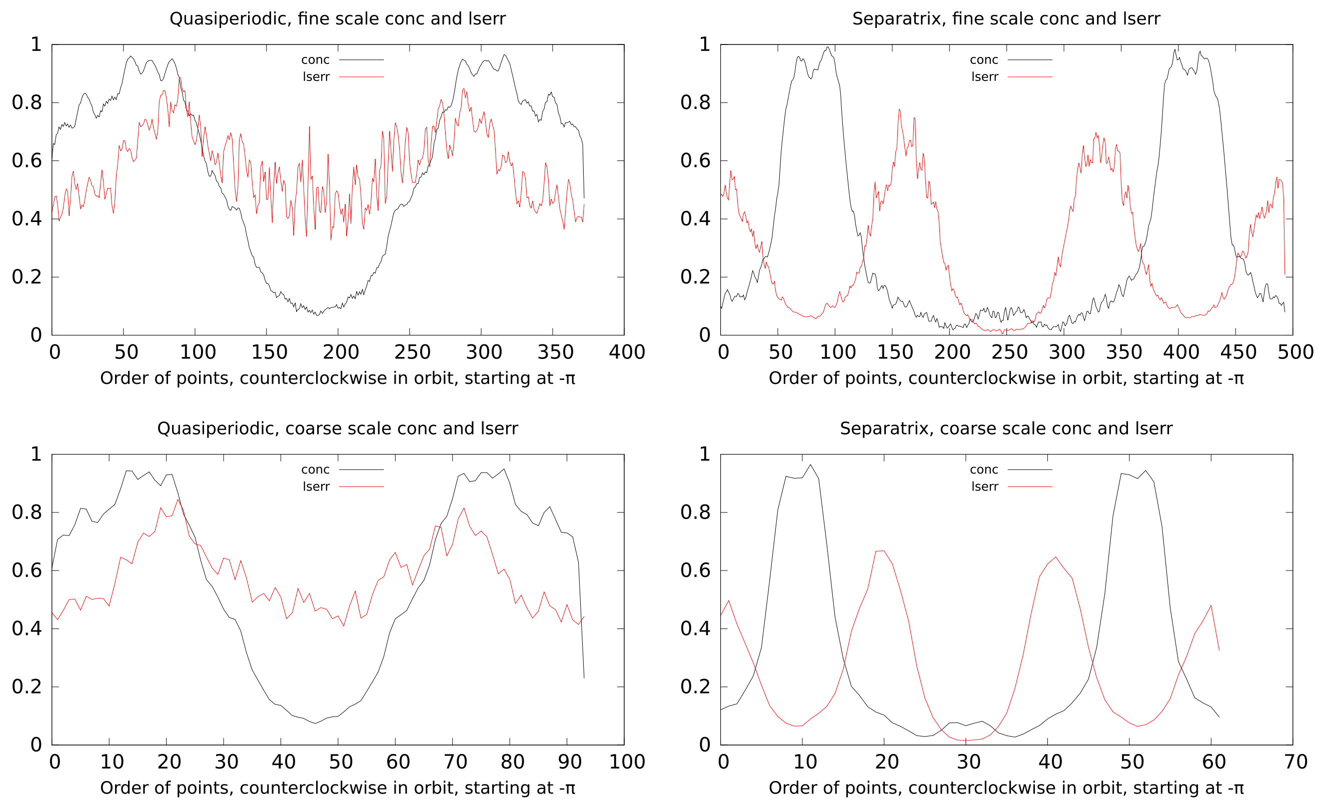

Fig. 13 The smoothed *lserr* and *conc* features, in red and black, respectively, for the quasiperiodic (left) and separatrix (right) orbits from Fig. 2. Top row: features obtained after decimated wavelet transform until the number of points in an orbit is between 250 and 500. Bottom row: the features after additional smoothing until fewer than 100 points are left in the orbit. These are the fine and coarse versions of the two features. The points are ordered by value of θ, going counterclockwise from $-\pi$ to π radians (colour figure online)

the fraction that were misaligned, that is, the peaks/valleys of *lserr* were aligned with the valleys/peaks of *conc*. These features allowed us to represent the behavior we see in Figs. 13 and 14. In addition, to account for cases where the peak and valley locations could be off by a small amount, we calculated an additional feature, the sign alignment; this is the fraction of points where the sign of $\Delta lserr$ and $\Delta conc$ is the same, that is, where *lserr* and *conc* are both increasing or both decreasing. This fraction would be large in orbits where the peaks and valleys in the two vectors are aligned.

Table 1 lists the 17 features that represent each orbit; the first three (file name, total number of points in orbit, and the number of points used in the instance) identify the provenance of an orbit and are not used in classification. The remaining features were selected after initial experiments with ensembles of decision trees to remove features that were not discriminating. For example, features related to *conc*, such as its minimum, maximum, and mean across the windows, did not convey any information about the class of an orbit, but the vector of *conc* values in windows around valid points was useful when combined with the *lserr* vector to identify the relationship of peaks and valleys between the two variables.

4.3 Feature selection

Once we have identified and extracted the features for each orbit, we used three feature selection methods (see [15] for details) to determine if some features are more important than others in classifying the orbits. These methods are:

- *Distance filter:* this filter identifies a more discriminating feature as one with greater distance between the histograms of the different classes. For each feature, we create a histogram of its values for each class and obtain the Kullback–Leibler distance between these histograms. The features are ranked by sorting them in descending order of the distances.
- *Chi-square filter:* this method ranks features by sorting them in descending order of chi-square statistics computed from their contingency tables [16]. As our features are all numeric (except for f16), they are first discretized

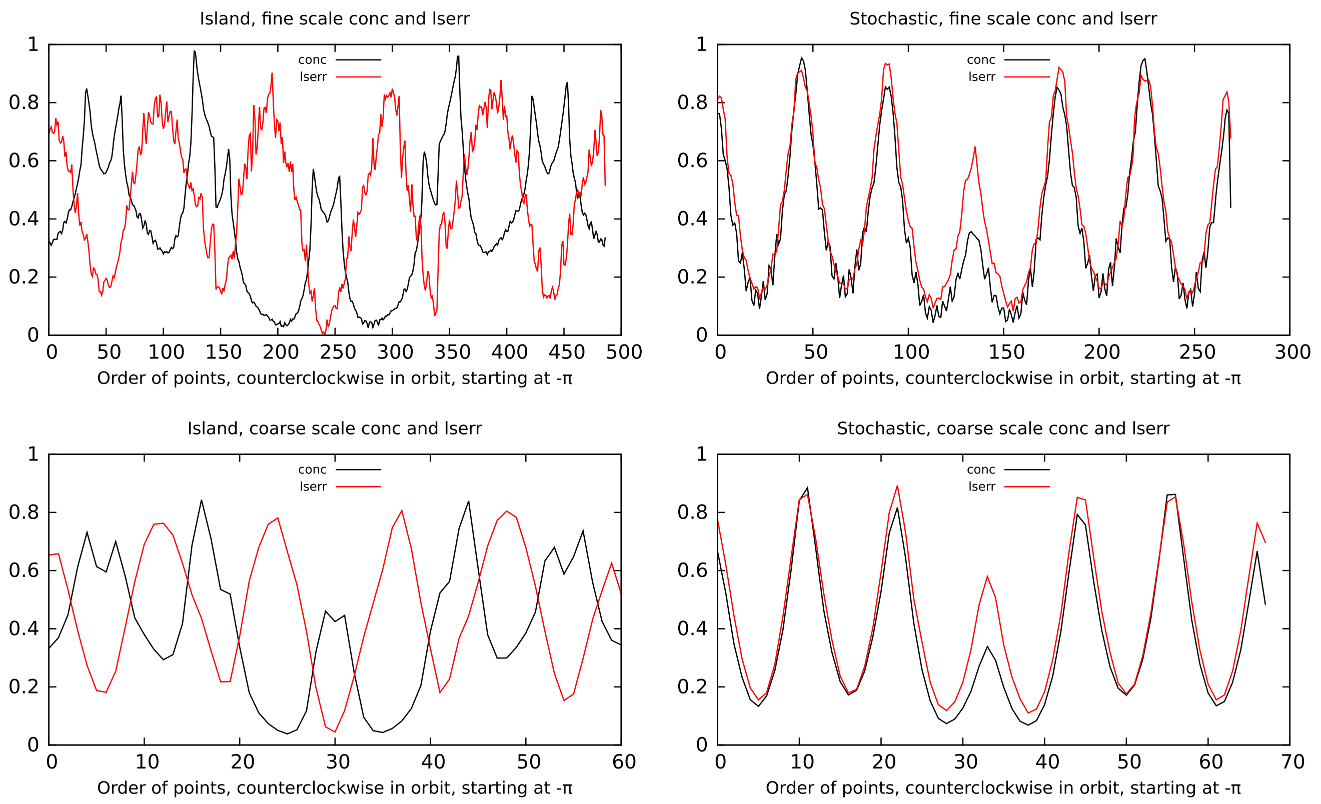

Fig. 14 The smoothed *lserr* and *conc* features, in red and black, respectively, for the island chain (left) and stochastic (right) orbits from Fig. 2. Top row: features obtained after decimated wavelet transform until the number of points in an orbit is between 250 and 500. Bottom row: the features after additional smoothing until fewer than 100 points are left in the orbit. These are the fine and coarse versions of the two features. The points are ordered by value of θ, going counterclockwise from $-\pi$ to π radians (colour figure online)

Table 1 A description of the features extracted for the orbits

Feature name	Description
f0	Orbit file name
f1	Total number of points in orbit
f2	Number of points used in this instance
f3	Maximum *lserr*, across windows, from polynomial fit in each window
f4	Minimum *lserr*, across windows, from polynomial fit in each window
f5	Mean *lserr*, across windows, from polynomial fit in each window
f6	Standard deviation of *lserr*, across windows, from polynomial fit in each window
f7	Mean spread, across windows
f8	Maximum spread, across windows
f9	Minimum spread, across windows
f10	Fraction of peaks and valleys aligned at fine scale
f11	Fraction of peaks and valleys misaligned at fine scale
f12	Fraction of valid points with sign alignment of $\delta lserr$ and $\delta conc$ at fine scale
f13	Fraction of peaks and valleys aligned at coarse scale
f14	Fraction of peaks and valleys misaligned at coarse scale
f15	Fraction of valid points with sign alignment of $\delta lserr$ and $\delta conc$ at coarse scale
f16	Any points where ldtheta is greater than 10 degrees (Boolean variable)

The first three features are for identifying the provenance of each orbit and are not used in the classification using histograms, before the chi-square statistic for each feature is calculated.

- *Stump filter:* a simple approach to ranking features by importance is to use the split criterion in decision trees that determines which feature to split on at any node of the tree. We consider just the root node, where the entire training data set is used. In our work, we use the Gini index [17] and rank the features according to the purity of their optimal split.

4.4 Classification

We have many different classifiers we can use to build the predictive model. We selected decision trees as they allowed us to understand how the features were used to assign a class to an orbit. This helped to improve the correctness of class labels and the suitability of the features. We used the ASPEN algorithm, which generates an ensemble of trees based on approximate splits [18]. We introduce randomization at each node of the tree in two ways—by randomly sampling the instances at a node and selecting a fraction (we use 0.7) for further consideration, and by replacing the sorting of feature values by a histogram. At each node, we create a histogram of feature values for the sampled instances, evaluate the splitting criterion at the mid-point of each bin of the histogram, identify the best bin, and then select the split point randomly in this bin. The use of the histograms and the smaller number of samples speeds up the creation of the ensemble. We use two different splitting criteria—Gini [17] and InfoGain [19].

As we have described in the previous sections, extracting high-quality features and correctly labeling the orbits formed an important part of our work. One may well suggest if we could skip the explicit extraction of features and use deep neural nets for classification. Unfortunately, the format of the data in the form of coordinates of the points in an orbit, precludes this option. Even the use of shallow neural nets, as well as many other classifiers, is not an option as they are not interpretable and could not have helped with the iterative improvement of the accuracy through better labeling and more representative features.

5 Experimental results

We next present the results of feature selection and classification for the features listed in Table 1. We started with 264 orbits, removed the orbits with too few points, and split the remaining orbits into derived orbits comprised of the first 1000, 1500, ... points. The resulting training set had 1884 instances, with 778 quasiperiodic, 250 separatrix, 440 island chain, and 416 stochastic orbits, each represented by 14 features, f3 through f16, listed in Table 1.

Table 2 Ordering, by importance, of the 14 features representing the orbits, using three different feature selection methods

Rank order	Distance filter	Chi-square filter	Stump filter
1	f3	f8	f16
2	f8	f7	f3
3	f5	f3	f8
4	f6	f5	f5
5	f7	f6	f6
6	f16	f16	f7
7	f15	f15	f4
8	f4	f4	f14
9	f12	f14	f15
10	f10	f12	f9
11	f14	f10	f12
12	f11	f13	f10
13	f13	f11	f13
14	f9	f9	f11

The first six features identified by all three methods are the same, and based on Fig. 15 reduce the error rate to ≈ 5%

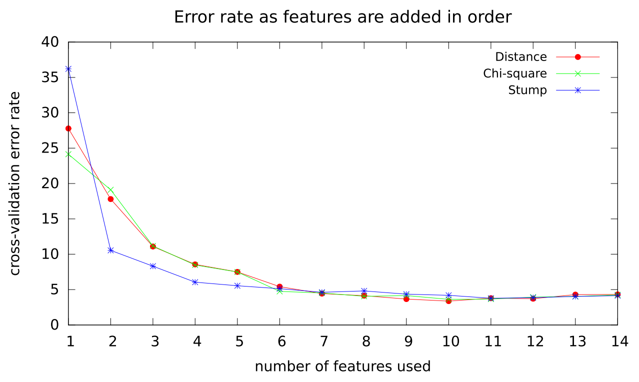

Fig. 15 Error rate from five runs of fivefold cross-validation for the top k features as identified by each of the three feature selection methods. The error stabilizes at 6–7 features. The ensemble used Gini split criterion with eleven trees

Table 2 lists the features in descending order of importance as identified by the three feature selection methods when applied to the full data set. Figure 15 shows how the error rate for five runs of fivefold cross-validation varies when we use only the first k features in order of importance. The error rate is for eleven trees using the Gini split criterion and the ensemble approach described in Sect. 4.4.

These results indicate that all three feature selection methods select the same top six features as important, though the order changes with the method as each uses a different metric to evaluate the importance. When these top six features are considered, the error rate drops to ≈ 5%, and stabilizes thereafter. These six features are *ldtheta_large*, the mean and maximum spread across windows, and the maximum, mean, and standard deviation of *lserr*, taken across windows, where

Table 3 Error rate for five runs of fivefold cross-validation using Gini and InfoGain split criteria

	5 trees	11 trees	21 trees
Gini	4.55 (0.05)	4.02 (0.13)	3.98 (0.07)
InfoGain	3.89 (0.08)	3.60 (0.15)	3.46 (0.13)

The values in parenthesis are the standard error across the five runs

lserr for a window is the maximum error from a polynomial fit in that window. Interestingly, the peak-valley features are less important. For the full data set considered in feature selection, features indicating the angular gaps, the radial spread, and the number of curves (one or two) in each window, are more discriminating in identifying the class of an orbit.

Next, in Table 3, we show the results of five runs of fivefold cross-validation as the number of trees in the ensemble is varied. We observe that the InfoGain split criterion gives slightly better accuracy than the Gini split criterion. To obtain greater insight, especially as the training set has an unequal number of orbits of the four classes, we consider the leave-one-out metric that gives us the types of misclassification when a model, built with all but one instance, is used to predict the class of that instance. Table 4 lists the confusion matrix for five, eleven, and twenty-one trees in the ensemble for the Gini and InfoGain split criteria. These results indicate that both the number of trees and the split criterion have relatively little effect on the overall accuracy of prediction. In addition, the largest numbers of misclassifications are when a separatrix orbit is misclassified as a quasiperiodic orbit and vice versa; this would likely be the case when the orbit is very thin or has mild stochasticity. The InfoGain split criterion results in fewer such misclassifications.

Finally, Fig. 16 shows a single tree created with all 1884 instances and the InfoGain split criterion. Examining the tree provides the following insights:

- The initial splits on features f16 and f6 create three distinct parts of the tree: The top part is a mix of quasiperiodic and separatrix orbits, followed by a part with a mix of separatrix and stochastic orbits, and the island chain orbits at the bottom of the tree. This confirms our observations that very thin separatrix orbits appear quasiperiodic at coarse scale and many separatrix orbits have some stochasticity.
- The tree selects f16 and f6 as the most important features. Feature f16, indicating large angular gaps, is an obvious choice as it is critical to identifying island chains. The choice of f6, which is the standard deviation of *lserr* across windows around valid points, allows quasiperiodic and very thin separatrix orbits, both of which have small values of f6, to be differentiated from other non-island orbits.
- Some leaf nodes have a large number of instances of the majority class. Two leaf nodes, with 531 and 151 instances, account for 87.6% of the quasiperiodic orbits, while two leaf nodes with 261 and 146 instances account for 92.5% of the island chain orbits, and one leaf node with 347 instances accounts for 83.4% of the stochastic orbits. However, instances of separatrix orbits are scattered throughout the tree, with the largest number at a leaf node being 99, or just 40% of the total separatrix orbits, confirming that separatrix orbits have the greatest overlap with other orbits, as observed visually.

Table 4 Confusion matrices obtained using the leave-one-out approach, with the Gini and InfoGain split criteria, and five, eleven, and twenty-one trees in the ensemble

		Predicted classes			
		Q	S	I	R
(a) Five trees, Gini, 4.3% error					
Actual classes	Q	757	15	3	3
	S	28	212	2	8
	I	6	2	432	0
	R	2	6	6	402
(b) Five trees, InfoGain, 3.4% error					
Actual classes	Q	762	8	3	5
	S	18	221	2	9
	I	4	3	433	0
	R	0	6	6	404
(c) Eleven trees, Gini, 3.5% error					
Actual classes	Q	762	12	2	2
	S	21	220	2	7
	I	6	2	432	0
	R	0	6	6	404
(d) Eleven trees, InfoGain, 3.45% error					
Actual classes	Q	763	9	3	3
	S	19	220	2	9
	I	4	2	434	0
	R	0	8	6	402
(e) Twenty one trees, Gini, 3.4% error					
Actual classes	Q	765	9	2	2
	S	21	221	2	6
	I	7	2	431	0
	R	0	7	6	403
(f) Twenty one trees, InfoGain, 3.08% error					
Actual classes	Q	762	11	2	3
	S	16	226	3	5
	I	4	2	434	0
	R	0	6	6	404

The confusion matrix is a summary of the actual and predicted classes of each instance in the data set

```
f16 = 0:
:  f6 < 0.0254451:
:  :  f6 < 0.00402432:
:  :  :  f13 < 0.875: class Q (531/0)         <---------- Q
:  :  :  f13 >= 0.875:
:  :  :     f11 < 0.0618687: class Q (17/0)  <---------- Q
:  :  :     f11 >= 0.0618687: class I (7/1)
:  :  f6 >= 0.00402432:
:  :     f4 < 0.000891132:
:  :     :  f5 < 0.0561031:
:  :     :  :  f12 < 0.654276:
:  :     :  :  :  f4 < 0.00048728:
:  :     :  :  :  :  f12 < 0.515885: class S (41/0)  <---------- S
:  :     :  :  :  :  f12 >= 0.515885: class S (4/3)
:  :     :  :  :  f4 >= 0.00048728:
:  :     :  :  :     f12 < 0.416771:
:  :     :  :  :     :  f7 < 0.334634: class S (14/1)  <---------- S
:  :     :  :  :     :  f7 >= 0.334634: class Q (3/0)
:  :     :  :  :     f12 >= 0.416771: class Q (8/0)
:  :     :  :  f12 >= 0.654276: class Q (21/0)    <---------- Q
:  :     :  f5 >= 0.0561031: class R (8/0)
:  :     f4 >= 0.000891132:
:  :        f15 < 0.363317:
:  :        :  f3 < 0.0837068: class Q (4/1)
:  :        :  f3 >= 0.0837068: class S (12/0)   <---------- S
:  :        f15 >= 0.363317:
:  :           f6 < 0.0142841: class Q (151/1)   <---------- Q
:  :           f6 >= 0.0142841:
:  :              f5 < 0.0638765:
:  :              :  f10 < 0.1125: class S (4/0)
:  :              :  f10 >= 0.1125:
:  :              :     f3 < 0.101987: class Q (10/0)
:  :              :     f3 >= 0.101987: class S (3/1)
:  :              f5 >= 0.0638765: class Q (23/0)    <---------- Q
:  f6 >= 0.0254451:
:     f8 < 0.8125:
:     :  f4 < 0.00613825:
:     :  :  f7 < 0.195314: class I (5/0)
:     :  :  f7 >= 0.195314:
:     :  :     f15 < 0.481: class S (99/0)   <---------- S
:     :  :     f15 >= 0.481:
:     :  :        f8 < 0.6875:
:     :  :        :  f9 < 0.125: class S (34/0)  <---------- S
:     :  :        :  f9 >= 0.125: class S (9/3)
:     :  :        f8 >= 0.6875: class R (7/0)
:     :  f4 >= 0.00613825:
:     :     f13 < 0.166667: class S (14/3)
:     :     f13 >= 0.166667:
:     :        f8 < 0.625: class S (8/0)
:     :        f8 >= 0.625:
:     :           f3 < 0.501971: class S (2/3)
:     :           f3 >= 0.501971: class R (43/1)  <---------- R
:     f8 >= 0.8125: class R (347/1)     <---------- R
f16 = 1:
   f10 < 0.121403: class I (261/0)  <---------- I
   f10 >= 0.121403:
      f5 < 0.239159: class I (146/4)   <---------- I
      f5 >= 0.239159:
         f9 < 0.125: class R (5/0)
         f9 >= 0.125: class I (19/1)    <---------- I
```

Fig. 16 Single tree created using the entire data set, InfoGain split criterion. The data set has 1884 instances—778 quasiperiodic, 250 separatrix, 440 island chains, and 416 stochastic (indicated by R) orbits. The node leaves with a large number of orbits of one class have been highlighted with the class initial. Together, they comprise 743 (95%) of the quasiperiodic orbits, 200 (80%) of separatrix orbits, 426 (97%) of the island chain orbits, and 386 (94%) of the stochastic orbits

- The relatively small size of the tree, along with the leaf nodes with a large percentage of each of the four classes of orbits, indicates that the features used in the training set are able to discriminate among the classes successfully.
- By focusing on the leaf nodes with a large number of orbits of one class, and the decisions made in the path to these nodes, we can understand how many of the labels are assigned. For example, a large number $(531 + 17)$ of Q orbits are identified by very small values (< 0.004) of f6, the standard deviation of *lserr* across windows around valid points. A majority of the stochastic orbits are identified by larger values of both f6 (> 0.0254) and f8 (> 0.8125), the maximum spread across windows. The key identifying feature for island chains is obviously f16. It is the separatrix orbits that take a deeper path through the tree as they share many similarities with both quasiperiodic and stochastic orbits.
- The highly ranked features in Table 2, which were identified using the full data set, occur at levels closer to the root of the tree, suggesting they remain important when the subsets of the data at the nodes are considered.
- The feature f14, identifying the fraction of peaks and valleys misaligned at coarse scale, does not appear in the tree, though its counterpart, f11, at fine scale, appears at a leaf node to distinguish a small number of quasiperiodic from island chain orbits. Both f11 and f14 are lower ranked features in Table 2 when the entire training data set is considered.

Finally, a limitation of our approach is that for orbits with few points, the features extracted may not represent the class. A lower bound on the number of points would depend on the type of orbit and the locations of the points. This limitation, perhaps to a lesser extent, is also present for visual classification as seen in Figs. 3 and 4 .

6 Conclusions

In this paper, we considered the task of assigning a class label to orbits in a Poincaré plot, where each orbit is represented by the coordinates of a set of two-dimensional points and the class label indicates the shape formed by the points. Though the data set is small, and the task is ideally suited for a machine learning solution, creating a high-quality training data set is challenging. We presented an approach that iteratively refines the class assignment and the features extracted to represent each orbit. Our decision tree-based approach results in less than 5% error rate, demonstrating that an automated machine learning approach can replace a tedious, error-prone, subjective visual classification of orbits, providing plasma physicists a useful tool for analysis of simulation data. It would be useful to extend this work to understand whether the model obtained from one simulation code could be directly applied to data generated by a different code, and to determine how the features may need to be modified to accomplish this.

Acknowledgements I would like to thank Scott Klasky (ORNL) for introducing us to the problem and Joshua Breslau (PPPL) for the data and domain expertise. Erick Cantú-Paz proposed the use of polar coordinates to magnify the radial variation in an orbit and Siddharth Manay created the Poincaré plot schematic in Fig. 1a and wrote the initial codes

for fitting the second degree polynomial.
LLNL-TR-834167 This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Funding This work was part of the SDM Scidac-II Center, funded by the Office of Science, US Department of Energy.

Declarations

Conflict of interest The author states that there is no conflict of interest.

References

1. DIII-D fusion website (2009). http://fusion.gat.com/global/DIII-D
2. The ITER project website (2009). http://www.iter.org/
3. Sanderson, A.R., et al.: Visualizing patterns in the Poincare plot of a magnetic field. In: IEEE Visualization Conference 2006 Compendium (2006)
4. Tricoche, X., Schlei, W., Howell, K.C.: Extraction and visualization of Poincare map topology for spacecraft trajectory design. IEEE Trans. Vis. Comput. Graph. **27**(2), 765–774 (2021). https://doi.org/10.1109/TVCG.2020.3030402
5. Albert, C.G., Kasilov, S.V., Kernbichler, W.: Accelerated methods for direct computation of fusion alpha particle losses within, stellarator optimization. J. Plasma Phys. **86**(2), 815860201 (2020). https://doi.org/10.1017/S0022377820000203
6. Rath, K., Albert, C.G., Bischl, B., von Toussaint, U.: Orbit classification and sensitivity analysis in dynamical systems using surrogate models. Phys. Sci. Forum (2021). https://doi.org/10.3390/psf2021003005
7. Sanderson, A., et al.: Analysis of recurrent patterns in toroidal magnetic fields. IEEE Trans. Vis. Comput. Graph. **16**(6), 1431–1440 (2010). https://doi.org/10.1109/TVCG.2010.133
8. Tricoche, X., Garth, C., Sanderson, A.: Visualization of topological structures in area-preserving maps. IEEE Trans. Vis. Comput. Graph. **17**(12), 1765–1774 (2011). https://doi.org/10.1109/TVCG.2011.254
9. Yip, K.M.-K.: KAM: A System for Intelligently Guiding Numerical Experimentation by Computer. MIT Press, Cambridge (1991)
10. Bagherjeiran, A., Kamath, C.: Graph-based methods for orbit classification (extended version). Technical Report UCRL-CONF-215802, Lawrence Livermore National Laboratory (2005)
11. Bagherjeiran, A., Kamath, C.: Graph-based methods for orbit classification. In: Proceedings. SIAM International Conference on Data Mining, pp. 574–578. SIAM, Philadelphia (2006)
12. Cressie, N.A.C.: Statistics for Spatial Data. John Wiley, Hoboken (1993)
13. Burrus, C.S., Gopinath, R.A., Guo, H.: Introduction to Wavelets and Wavelet Transforms: A Primer. Prentice Hall, Upper Saddle River (1998)
14. Hubbard, B.B.: The World According to Wavelets: The Story of a Mathematical Technique in the Making. A. K. Peters, Wellesley (1998)
15. Cantú-Paz, E., Newsam, S., Kamath, C.: Feature selection for scientific applications. In: Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 788–793 (2004)
16. Huang, S.H.: Dimensionality reduction on automatic knowledge acquisition: a simple greedy search approach. IEEE Trans. Knowl. Data Eng. **15**(6), 1364–1373 (2003)
17. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. CRC Press, Boca Raton (1984)
18. Kamath, C., Cantú-Paz, E., Littau, D.: Approximate splitting for ensembles of trees using histograms. In: Proceedings, 2nd SIAM International Conference on Data Mining, pp. 370–383 (2002)
19. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)