Knowledge and Information Systems (2023) 65:1587–1609
https://doi.org/10.1007/s10115-022-01802-5

REGULAR PAPER


# An experimental study on classifying spatial trajectories

**Hasan Pourmahmood-Aghababa**[1] · **Jeff M. Phillips**[1]

Received: 25 August 2022 / Revised: 16 September 2022 / Accepted: 27 November 2022 /
Published online: 20 December 2022


**Abstract**
We provide the first comprehensive study on how to classify trajectories using only their spatial representations, measured on 5 real-world datasets. Our comparison considers 20 distinct classifiers arising either as a KNN classifier of a popular distance, or as a more general type of classifier using a vectorized representation of each trajectory. We additionally develop new methods for how to vectorize trajectories via a data-driven method to select the associated landmarks, and these methods prove among the most effective in our study. These vectorized approaches are simple and efficient to use, and also provide state-of-the-art accuracy on an established transportation mode classification task. In all, this study sets the standard for how to classify trajectories, including introducing new simple techniques to achieve these results, and sets a rigorous standard for the inevitable future study on this topic.

**Keywords** Computational geometry · Machine learning · Classification · Trajectory mining

## 1 Introduction

A trajectory is the tracing of an object through physical space, and is now a first-class object in spatial data analysis due to the ease of creating these objects via GPS trackers. Also, their analysis has been at the forefront of geo-spatial research in clustering [1–5], similarity measures [6–9], classification [10–25], and transportation mode detection [26–35]. In this paper, we do not factor in the absolute time these traces are made, but mostly treat the trajectories as geometric objects in the plane. Moreover, in this setting, we focus on the central machine learning tasks of being able to classify trajectories to predict if they belong to one of two classes.

To instantiate what is spatial trajectory classification and why we need to do classification, consider traffic data. Discovering driving behavior patterns are critical for any government

Hasan Pourmahmood-Aghababa and Jeff M. Phillips have contributed equally to this work.

✉ Hasan Pourmahmood-Aghababa
h.pourmahmoodaghababa@utah.edu

Jeff M. Phillips
jeffp@cs.utah.edu

[1] School of Computing, University of Utah, Central Campus Dr., Salt Lake City UT 84112, Utah, USA

(as part of infrastructure design), auto companies (to adapt products to future use cases), and auto insurance companies (to adjust rates for safe versus reckless behavior). The simplest version of this is classifying drivers or routes as part of a favorable/standard versus unfavorable/divergent class. We posit that most driver analysis will either directly rely on such a task, or build on it as part of larger model. For instance, consider an adversary that uploads some corrupted trajectories in the road network database of a government. Then the need for a discriminator is a must. In many cases, the spatial classifiers are used with the addition of extra metadata information about the vehicle, timing, or the person generating that route. The choice of which ones to use is very situation-dependent and such we do not focus on this. However, we observe that the methods we recommend are easily adjustable to adding these information to the classifiers, as we will demonstrate in two of our experiments.

While in some cases, these full applications are pre-mature or should enact ethical safeguards before deploying, the task of building a classifier for trajectories will surely play an important role in their ultimate use. As part of this project we have identified 5 datasets with at least two naturally occurring classes of trajectories, of which it makes sense to classify. These data sets are publicly available, and our methodology is simple and reproducible, and we hope that these evaluation tasks become benchmarks for future development in this area.

Moreover, by using known techniques to combine distances meant to model trajectories, and other trajectory representations, the literature provides for dozens of ways classifiers can be built. In particular, a distance between trajectories (e.g., dynamic time warping, discrete Fréchet distance) can be used within a KNN (K-nearest neighbor) classifier. In addition, methods for featurizing a trajectory [36, 37] into a vector representation can be paired with off-the-shelf classifiers, from say sk-learn. Thus, we explore and evaluate a large cross-section of classifiers in this paper seeking to provide guidance on which ones work the best.

Furthermore, we adapt and extend several of these techniques to produce new methods which are among the best performing in the benchmarks we have designed. These methods include a new data-driven method to select the landmark points which the vectorized representations are derived from, and a voting mechanism to boost the effect of several classifiers. Moreover, we show these vectorized approaches can be combined with other features (e.g., velocity) and this achieves state-of-the-art performance on the standard task for predicting mode of transportation.

In all, this work initiates a formal study of classification tasks for spatial trajectories, identifies state-of-the-art methods – some are developed as part of this work, and provides as set of benchmarks so this field can continue to evolve in a reproducible way.

### 1.1 Existing previous work on trajectory classification

Classifying trajectories is a fundamental task in spatial data analysis, but as far as we are aware, has not comprehensively been studied as a stand-alone challenge. Yet, the core trajectory classification task factors into many important challenges, and is destined to have an ever-expanding role as spatial data analysis increases in automation. For instance, given a GPS trace, can we determine which individual most likely made it, or what the mode of transportation was. Or does the trajectory represent favorable behavior (a healthy animal, a customer who will make a large purchase) or an unfavorable one (a diseased animal, a shoplifter).

Previous trajectory classification work includes [36] which constructs some specific experiments similar to ours to demonstrate where their proposed technique is effective; we compare to and build on this work, but aim for a more objective and comprehensive comparative study.

They have applied their technique to Car-Bus and Geolife trajectory datasets [38, 39] that this paper will explore. Other prior work considers subtrajectory [10, 40] or segment [11] classification, which is not analyzing trajectories as a whole, and so is not applicable to our tasks. For instance, [10] builds a recurrent neural net (RNN) tuned to specific properties of their datasets—it centrally uses an encoded location id of each waypoint not available in general trajectory datasets, like the ones we consider. The focus of [11] is on mode-of-flight identification from very short subsets of trajectories. Another line of work is on inferring transportation modes [26–35, 41], our work will directly compare against this line of work in Sect. 3.6; the methods we propose improve upon all prior work in these baselines. Most of these papers use deep neural networks, except for [41] which uses the hidden Markov model in order to infer travel modes.

Another method of trajectory classification, is to utilize a KNN classifier through a trajectory distance metric. For a list of widely used trajectory distances, which we will use in our studies, see Table 3. A binary/multi resolution trajectory sketch (for Hausdorff distance for example) was introduced and employed to get a distance on trajectories [6]. It is then applied for trajectory classification and clustering. In another example, a grid-based technique for sketching spatial trajectories is given in [42]. This representation is then utilized to introduce a Euclidean-type distance on trajectories in order to apply with KNN classifier. Two experiments were conducted on two small-size real world datasets. However, unfortunately, the datasets are not public and so not reproducible.

## 2 Preliminaries

In this section first we describe existing landmark-based feature mappings for curves that not only allow us to define a distance on curves but also enables using almost all machine learning algorithms on curves. Then we provide a couple of simple enhancements to these methods. The first one is a data-driven method to select the landmark points based on which one is often causing a mistake, and the second uses multiple such classifiers built from different randomly chosen landmarks and returns the majority vote of all of the classifiers. We will see that both methods provide small but tangible improvements in classification results thereafter.

### 2.1 Existing work on curve vectorization

Formally, by a *curve* we mean the image of a continuous non-constant mapping $\gamma : [0, 1] \to \mathbb{R}^2$. The set of all curves is denoted by $\Gamma$. We recall the notions of landmark-based feature mapping and distance which were introduced in [36]. For any landmark $q \in \mathbb{R}^2$ one can consider the distance function $v_q : \Gamma \to \mathbb{R}$ defined by

$$v_q(\gamma) = \text{dist}(q, \gamma) = \min_{p \in \gamma} \|q - p\|.$$

Now for a set of landmarks $Q = \{q_1, \ldots, q_n\}$ we can vectorize the $v_q$ function to get a feature mapping $v_Q : \Gamma \to \mathbb{R}^n$ by

$$v_Q(\gamma) = (v_1(\gamma), \ldots, v_n(\gamma)),$$

where $v_i(\gamma) = v_{q_i}(\gamma)$ for $1 \leq i \leq n$. Indeed, this contribution of each landmark are stacked to get an appropriate vector representation of curves which enables us to take advantage of machine learning models for trajectory datasets, makes some clustering (like K-means) trivial and K-nearest neighbor classifier very efficient using fast Euclidean near-neighbor libraries.

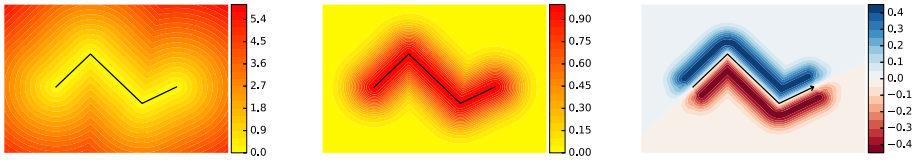


**Fig. 1** Left: Example of feature mapping $v_q$ for a curve. Middle: Example of feature mapping $v_q^{\exp}$ for a curve ($\eta = 2$). Right: Example of feature mapping $v_q^{\varsigma}$ for a curve, with sidedness encoded by positive/negative values ($\varsigma = 1$)

Now the landmark-based distance $\mathrm{d}_Q : \Gamma \times \Gamma \to \mathbb{R}$ is defined by

$$\mathrm{d}_Q(\gamma, \gamma') = \frac{1}{\sqrt{n}} \|v_Q(\gamma) - v_Q(\gamma')\|,$$

where $\|\cdot\|$ denotes the Euclidean distance in $\mathbb{R}^n$. We next consider 2 alternative ways to define the $v_q$ mapping, but for a set of landmarks $Q$ they create a vector and distance in an analogous way.

The second feature mapping is $v_q^{\exp}$, which is a vectorization of the function

$$v_q^{\exp}(\gamma) = \exp\big(-v_q(\gamma)^2/\eta^2\big),$$

and localizes the feature mapping $v_q$ to a neighborhood induced by a learned scale term $\eta$.

We remark that, however, the feature mappings $v_Q$ and $v_Q^{\exp}$ and the distance $\mathrm{d}_Q$ do not capture the orientation of curves, which encode the direction of the trajectories. An orientation preserving version which extends $v_Q^{\exp}$ is introduced in [37]. First assume we can define $n_p(q)$ the normal vector of a curve $\gamma$ (using the curve orientation to keep its direction consistent), where $p$ is the closest point on $\gamma$ to $q$, and a few other technical conditions [37]. Assume $\Gamma'$ shows the set of all simple curves which are differentiable almost everywhere. Then define $v_q^{\varsigma} : \Gamma' \to \mathbb{R}$ for some scale parameter $\varsigma > 0$ by

$$v_q^{\varsigma}(\gamma) = \langle n_p(q), q - p\rangle \exp\big(-\|q - p\|^2/\varsigma^2\big)/\varsigma.$$

When $p$ is an endpoint (i.e., $p = \gamma(0)$ or $p = \gamma(1)$), we need a slightly different definition which keeps the values local and continuous

$$v_q^{\varsigma}(\gamma) = \frac{1}{\varsigma}\langle n_p, \frac{q - p}{\|q - p\|}\rangle \|q\|_{\infty,p} \exp\big(-\|q - p\|^2/\varsigma^2\big),$$

where $\|q\|_{\infty,p}$ is the $l^\infty$-norm of $q$ in the coordinate system with axis parallel to $n_p$ and tangent line at $p$ and origin at $p$.

Another landmark-based distance, denoted $\mathrm{d}_Q^{\pi}$, was introduced in [36], which we will evaluate via KNN classifier. Considering a landmark set $Q = \{q_1, \ldots, q_n\}$ and two curves $\gamma, \gamma'$ we can define $\mathrm{d}_Q^{\pi}(\gamma, \gamma') = \frac{1}{n}\sum_{i=1}^{n} \|p_i - p_i'\|$, where $p_i = \operatorname{argmin}_{p\in\gamma} \|q_i - p\|$ and $p_i' = \operatorname{argmin}_{p\in\gamma'} \|q_i - p\|$. Notice if there are multiple points available as argmin points, considering the natural parametrization of curves on the interval [0, 1] and thus a linear order on the curve, the first point will be chosen. For an illustration of feature mappings $v_q$, $v_q^{\exp}$, $v_q^{\varsigma}$ see Fig. 1.

### 2.2 Proposed methods: mistake-driven algorithm for choosing landmarks

In previous work [36, 37], the landmarks were chosen randomly (from some bounding domain) for experimental evaluation, or sometimes on a fine grid within theoretical analysis. In this section we propose a new and more data-driven mechanism to select the landmarks. We refer to has a *mistake-driven* approach since it selects points nearby trajectories for which a classifier makes a mistake on, reminiscent of the perceptron algorithm.

Consider two training trajectory datasets $\mathcal{T}_1$ (examples with label 1) and $\mathcal{T}_2$ (examples with label 0) and set $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ (as the training set), and a particular classifier we denote `clf`. Our goal here is to identify a set of landmarks that help to improve misclassification error rates for use with `clf`. To accomplish this, first we designate a scale parameter, say $\eta$, which controls the standard deviation of a random process placing landmarks near curves. We set $\eta = \|m_1 - m_2\|$, where $m_1$ and $m_2$ are the mean of $(x, y)$-coordinates of trajectories in the first and second training datasets, respectively. Next, we choose a random landmark $q_0$ by adding a Gaussian noise with standard deviation of $\eta$ to a uniformly randomly chosen waypoint of a uniformly randomly chosen trajectory. Then we train a classifier of type `clf` on the mapped data $v_Q^{\exp}(\mathcal{T}, \eta)$, where $Q = \{q_0\}$. Then we choose the most misclassified trajectory, say $\gamma$, according to their scores given by the trained classifier and select a random landmark $q_1$ by adding a Gaussian noise with standard deviation of $\eta$ to a uniformly randomly chosen waypoint of $\gamma$ and append it to $Q$. We do the same process with $Q = \{q_0, q_1\}$ to choose another landmark. We repeat this process until we get the desired number of landmarks. The pseudo code is given in Algorithm 1. These landmarks are then used to generate a vector representation for trajectories, and a final `clf` classifier is trained.

**Algorithm 1** Mistake-driven approach to choose landmarks

**Input:** Two sets of trajectories $\mathcal{T}_1, \mathcal{T}_2, n$ (the number of needed landmarks), $\eta$ and a classifier `clf`.
**Output:** A set of landmarks $Q$ of size $n$.
Randomly select a trajectory $\gamma_0$ in $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$.
With standard deviation of $\eta$, select a random landmark $q_0$ near to a point of $\gamma_0$; initialize $Q = \{q_0\}$.
**for** $i = 1, \ldots, n-1$ **do**
  Train a model using `clf` on the vectorized data $v_Q^{\exp}(\mathcal{T}, \eta)$.
  Select a trajectory $\gamma_i$ that is most misclassified among $\mathcal{T}$.
  With standard deviation of $\eta$, select a random landmark $q_i$ near to a point of $\gamma_i$ and append it to $Q$.
**end for**
**return** $Q$

## 3 Experimental evaluation of trajectories classification methods

***Datasets***

We have done experiments on 5 real world and 1 synthetic datasets which are explained in detail in the following subsections. Table 1 shows an overview of these datasets, detailing the data size and also the size of a bounding box. We also show the scale parameter $\varsigma$ used in experiments, we generally aim to set it about the same or a bit larger than the bounding box. In the Characters, T-drive and Geolife datasets where the data spread changes over the same domain, we keep it fixed for all experiments in that dataset type.

As a general preprocessing step for all experiments first we remove stationary waypoints (i.e., consecutive waypoints with no movement in between) and then remove all trajectories

**Table 1** This table represents an overview of datasets used in the experiments in this paper. Here "Size" shows the number of selected trajectories after preprocessing step and "L × W" shows the length and width of the rectangular region containing all the sampled trajectories

| Dataset | Pairs | Size | L × W | $\varsigma$ |
|---|---|---|---|---|
| **Car-Bus** | – | 76, 44 | 0.67 × 0.61 | 1 |
| **Simulated Car-Bus** | – | 228, 220 | 0.67 × 0.62 | 1 |
| **Two Persons** | – | 124, 89 | 30.53 × 15.64 | 100 |
| **Characters** | *u, w* | 125, 131 | 112.63 × 80.55 | 100 |
| | *n, w* | 125, 130 | 121.38 × 87.84 | 100 |
| | *n, u* | 131, 130 | 124.82 × 84.34 | 100 |
| | *b,c* | 174, 141 | 83.11 × 82.55 | 100 |
| | *c, o* | 141, 142 | 98.75 × 61.08 | 100 |
| **T-drive** | 3142,6834 | 143, 116 | 14.97 × 16.64 | 10 |
| | 6168, 9513 | 143,119 | 0.97 × 0.83 | 10 |
| | 1950, 5896 | 140, 123 | 1.16 × 1.31 | 10 |
| | 2876, 3260 | 100, 132 | 0.81 × 0.51 | 10 |
| | 1350, 5970 | 132, 127 | 0.76 × 0.48 | 10 |
| **Geolife** | 15, 44 | 154, 197 | 0.32 × 0.6 | 1 |
| | 15, 125 | 154, 122 | 14.37 × 4.07 | 1 |
| | 16, 44 | 140, 197 | 0.18 × 0.32 | 1 |
| | 33, 40 | 117, 193 | 0.21 × 0.52 | 1 |

The notation $\varsigma$ shows the scale parameter

with less than 10 waypoints. The trajectories are randomly split (70/30) into train and test data and we report misclassification error on test data for several classifiers, averaged over 50 random train-test splits.

The experimental setup for the mistake-driven algorithm is as follows. We split data into train and test data (70/30) and calculate the threshold value $\eta$ using the training data. Then we apply our method of choosing landmarks (Algorithm 1) 3 times (unless otherwise specified) on training data and opt for one with the lowest training error. This one is then used on test data to report the results in tables (as usual, averaged over 50 test/train splits).

We apply the same methodology for different choices of landmarks $v_Q$, $v_Q^{\exp}$, $v_Q^{\varsigma}$ and using endpoints, when appropriate. The generic random landmarks are chosen from normal distribution with mean of all waypoints and standard deviation 4 times the standard deviation of all waypoints, and are denoted "Rand $v_Q$". The ones that are mistake-driven are denoted "MD $v_Q$". When voting is used it is marked Vote$(\cdot)$.

***Classification methods used.*** Given the various ways of achieving vectorized representations (including just mapping to $\mathbb{R}^4$ via the coordinates of the endpoints), we experiment with Convolutional Neural Networks and 7 classifiers from `sklearn` that are given in Table 2. The parameter $\varsigma$ in $v_Q^{\varsigma}$ vectorization for all experiments in this section is chosen sufficiently large with respect to the length of the rectangular region containing all trajectories in order to reduce the Gaussian weight impact on the vectorization while capturing the orientation. For the CNN we use a 1-layer architecture with 10 filters for the hidden layer. Padding and stride of 1 are utilized and convolution kernel size is set to 2 or 5 depending on the dimension of the input data.

**Table 2** This table shows an overview of classification methods used. All other hyperparameters/parameters are set to the default ones

| Classification method | In short | Hyperparameters |
|---|---|---|
| Linear Kernel SVM | LSVM | $C = 100$ |
| Gaussian Kernel SVM | GSVM | $C = 100$ |
| Polynomial Kernel SVM | PSVM | $C = 100, deg =$ '$auto$' |
| Decision Tree | DT | No limit on '$max\_depth$' |
| Random Forest | RF | 50 Estimators |
| K-Nearest Neighbor | KNN | 5 Neighbors |
| Logistic Regression | LR | – |
| Convolutional Neural Network | CNN | See the text |

**Table 3** The table shows 12 famous trajectory distances used for KNN classifier

| Distances | Reference | Implementation |
|---|---|---|
| Continuous Fréchet Distance | [44] | [45] |
| Discrete Fréchet Distance | [46] | [45] |
| Hausdorff Distance | [47] | [45] |
| Dynamic Time Warping (DTW) | [48] | `tslearn` |
| Fast Dynamic Time Warping (fastdtw) | [49] | [50] |
| Soft Dynamic Time Warping (soft-dtw) | [51] | [52] |
| Symmetric Segment-Path Distance (SSPD) | [1] | [45] |
| Longest Common Subsequence (LCSS) | [53] | [45] |
| Edit Distance on Real sequence (EDR) | [54] | [45] |
| Edit distance with Real Penalty (ERP) | [55] | [45] |
| $\mathrm{d}_Q^\pi$ (Landmark-based distance) | [36] | [56] |
| LSH (with binary sketches for Hausdorff distance) | [6] | By authors |

We compare these results with KNN (K-nearest neighbor) classifier with $K = 5$ estimators from `sklearn` library, using 12 different distances given in Table 3. In all experiments with KNN using LSH, we have applied 20 random circles to get binary sketches utilized in LSH distance. To choose the circles' radius $r$, like computing $\eta$ in the mistake-driven algorithm, we set $r = \|r_1 - r_2\|$, where $r_1$ and $r_2$ are the mean of $(x, y)$-coordinates of trajectories in the 70% of the first and second datasets used as training data. Finally, in all landmark-based experiments in this paper we use 20 landmarks. Indeed, in random $v_Q$, $v_Q^{\exp}$ and $v_Q^{\varsigma}$ we randomly generate 20 landmarks around the curves. For mistake-driven approaches, Algorithm 1 chooses 20 landmarks accordingly. To make it easy to reproduce or compare against our results, we link to all datasets, and host cleaning and testing methods on a public github page [43].

All resulting figures (Figures 3, 4, 6, 12, 13, 8, and 10 and Table 4) show the average classification error over 50 trials on random test/train splits. The error bars show standard-deviation of these 50 trials.

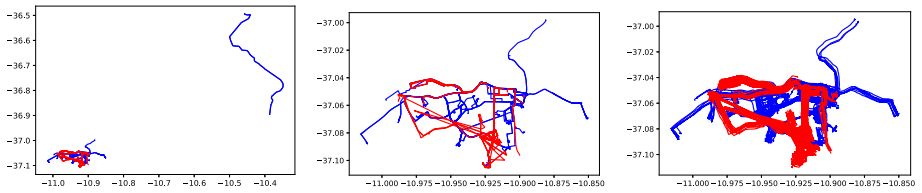

**Fig. 2** Left: Car (blue) and bus (red) trajectories (full original dataset used in experiments). Middle: zooms in for a close up of most interesting data. Right: shows zoomed in region of the simulated car-bus dataset (Color figure online)

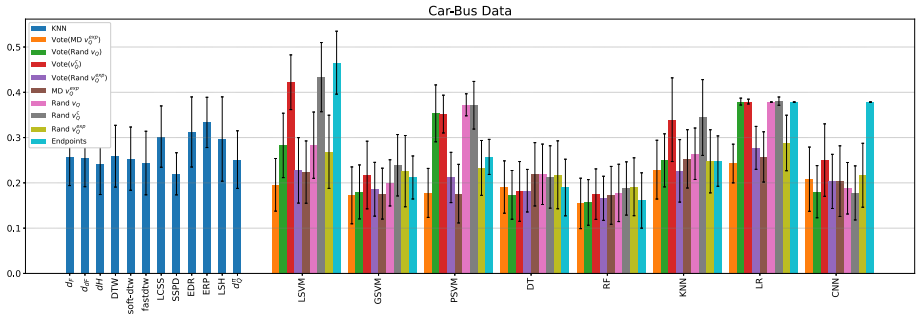


**Fig. 3** Average classification test errors of Car-Bus dataset of several classification techniques (in bar charts) with different methods of featurizing the data (in color), mostly based on the way the landmark set $Q$ is chosen. Also, average KNN-classification test errors with various distances. Parameters: soft-dtw: $\gamma = 1e-15$, LCSS and EDR: $\epsilon = 0.02$

### 3.1 Car-Bus dataset

We first consider the GPS Trajectories Dataset from UCI Machine Learning Repository [38], recorded in Aracuja, Brazil (see Fig. 2). There are 87 car and 76 bus trajectories in this dataset. After the preprocessing step described above, a set of 78 car and 45 bus trajectories remained. Moreover, in order to make the problem more challenging, we removed the 2 clear outliers from car and 1 from bus trajectories and ended up with 76 car and 44 bus trajectories. In this experiments $C = 10$ is applied for PSVM classifier.

The misclassification rates are shown in Fig. 3. The figure shows results of applying a variety of baseline distances, and classification using KNN; all of these approaches have error of at least 22% on the test data. Other methods like decision trees are not generically possible to do with only access to a distance. The best misclassification rate on the test error of 15.46% is shown in Fig. 3 and achieved by Random Forest using voting technique with mistake-driven landmarks. In general, the mistake-driven landmarks perform better than the similar methods with randomly chosen landmarks or just endpoints as shown in Fig. 3. The $v_Q^{\text{exp}}$ landmarks mostly work a bit better than $v_Q$ and $v_Q^{\varsigma}$ landmarks across techniques. The Random Forest methods do consistently well (with between 15% and 19% error). In addition to the good performance of Random Forest classifier, Gaussian SVM tends to perform well with between 17% and 24% error, but in this case does not achieve the smallest overall test error. In fact, just using endpoints (see Fig. 3), Random Forest achieves the third best result of 16.11% error—however, other classifier types other than DT, do poorly with this representation, so this does not appear to be a good general purpose way to vectorize the trajectories.

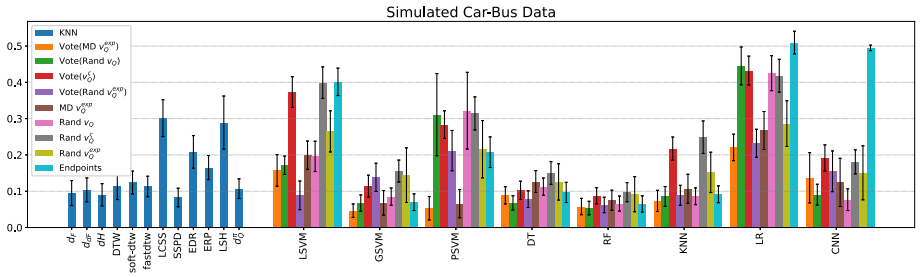


**Fig. 4** Average classification test errors of Simulated Car-Bus dataset of several classification techniques (in bar charts) with different methods of featurizing the data (in color), mostly based on the way the landmark set $Q$ is chosen. Also, average KNN-classification test errors with various distances on trajectories. Parameters: soft-dtw: $\gamma = 1e-15$, LCSS and EDR: $\epsilon = 0.02$

### 3.2 Simulated car-bus dataset

In this experiment, to create a larger and more balanced dataset, we add some noise to the preprocessed trajectories in car-bus dataset in Sect. 3.1 to get 2 noisy copies of each car and 4 noisy copies of each bus trajectories and combine them with the original preprocessed car and bus trajectories. Thus we end up with a roughly balanced data including 226 car and 220 bus trajectories (see Fig. 2). To be more precise, we add a noise offset vector $v \in \mathbb{R}^2$ to each waypoint. We initialize $v = 0.001$, and this is the noise added to the start point. Then before modifying each subsequent waypoint we update $v \leftarrow v + N(0, 0.0001)$, which is the two-dimensional normal distribution with mean $(0, 0)$ and standard deviation 0.0001.

The misclassification rates are presented in Fig. 4. The relative accuracy of classification methods is similar to that of the original car-bus dataset, but the accuracy in this setting is generally better. The left side bars show results of applying 12 baseline distances, and classification using KNN. They all have error of at least 8% on the test data; the best performing one is SSPD-KNN. The overall best misclassification rate of 4.64% shown in Fig. 4 is achieved by Gaussian SVM using voting technique of mistake-driven landmarks. In general, the voting method with mistake-driven landmarks does better than other vectorization techniques or just endpoints as shown in Fig. 4. Comparing the performance of featurization methods we observe that random $v_Q$ vectorization technique tends to perform a bit better than $v_Q^{\zeta}$ and $v_Q^{\exp}$ vectorizations. Perhaps surprisingly, with just using endpoints as the vectorized features (see Fig. 4), Random Forest achieves a low error rate of 6.49%; this may be partially explainable due to the relatively low noise of starting points in the noisy copies of each trajectory. Linear classifiers generally perform poorly on most featurizations.

### 3.3 Two persons trajectory data

The trajectory data in this experiment was obtained from the GPS carried by two members of "Databases and Mobile Computing Laboratory in University of Illinois at Chicago" during their daily commute for 6 months (see Fig. 5). The dataset was created in 2006 and is available for public [57]. Each trajectory represents a continuous trip of a member in Cook county and/or Dupage county, Illinois. One of the persons has 124 trajectories and the other 89 trajectories. We removed all stationary waypoints and trajectories with less than 10 waypoints as the general preprocessing step; however, the number of trajectories for each person did not change. Similar to other experiments the goal is to classify each person's trajectory.

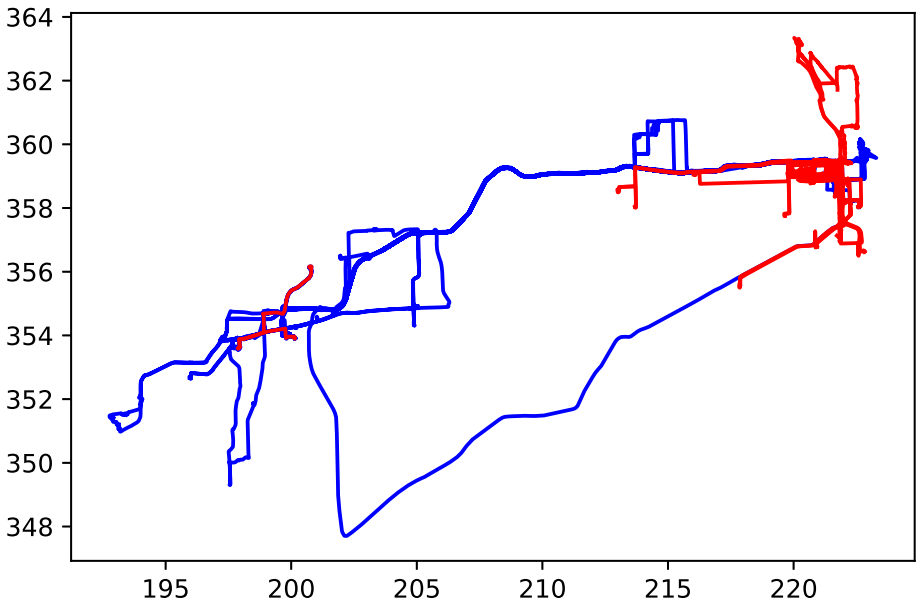


**Fig. 5** GPS trajectories of two members (one in blue and the other in red) of "Databases and Mobile Computing Laboratory in University of Illinois at Chicago" during their daily commute for 6 months which are captured in the Cook county and/or the Dupage county of Illinois

Most of the trajectories in this dataset are very long and thus the running time of KNN classifier with almost all distances are high. Specially, the continuous Fréchet distance did not complete in 48 hours, and so we do not report any results. Note that since our vectorization methods are efficient using optimized Euclidean libraries, they are very fast in comparison with KNN classifier using variety of distances (running times are discussed more in Sect. 5).

The results in Fig. 6 show the misclassification errors on test data. The lowest misclassification rate of 4.49% is achieved by mistake-driven landmarks using Linear SVM and also by Convolutional neural networks applying voting technique on $v_Q^{exp}$-vectorization. Other classifiers included cannot achieve an error rate less than 4.83% (which is also gained by mistake-driven landmarks) and KNN with variety of popular distances cannot do better than 5.26%. All other vectorization techniques tend to work roughly similar on this dataset, where all could get under 6% misclassification rate with at least one classifier. It seems that the direction of trajectories is not essential as the $v_Q^{\zeta}$-vectorization did comparatively poorly with about 8% test error applying Random Forest and KNN classifier using voting technique except CNN that could get 5.69% misclassification rate.

### 3.4 Characters dataset

For a change of pace, this experiment is done on the Character Trajectories Dataset from UCI Machine Learning Repository that consists of handwritten characters captured using a WACOM tablet – so is not mobility data. We chose 5 similar pairs of letters {(u, w), (n, w), (n, u), (b, c), (c, o)} (see Fig. 7) to perform a binary classification for each pair. For other pairs of characters we did a binary classification with $v_Q$-vectorization using random landmarks

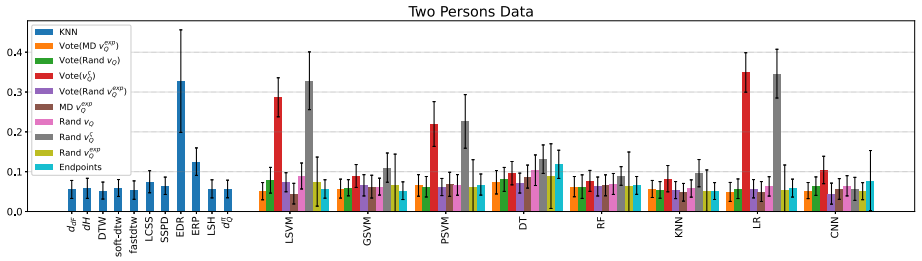


**Fig. 6** Average classification test errors of Two Persons dataset of several classification techniques (in bar charts) with different methods of featurizing the data (in color), mostly based on the way the landmark set $Q$ is chosen. Also, average KNN-classification test errors with various distances. Parameters: soft-dtw: $\gamma = 1e-10$, LCSS and EDR: $\epsilon = 0.1$

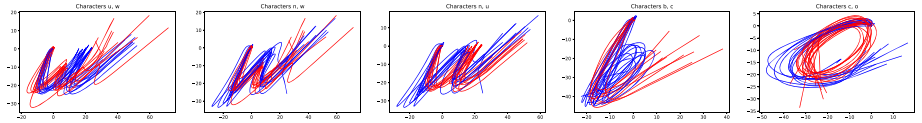


**Fig. 7** A subset of trajectories (10 trajectories of each character) are presented

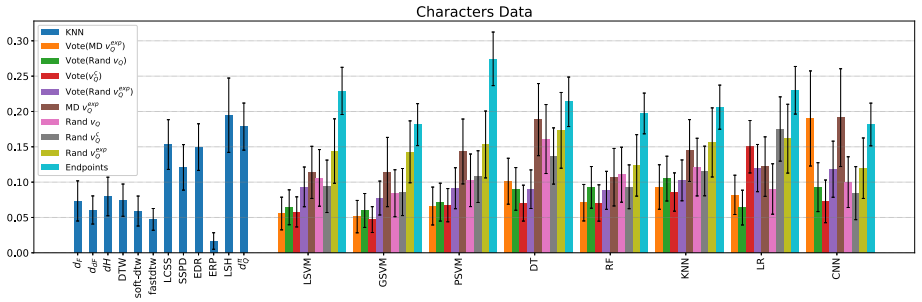


**Fig. 8** Average classification test errors of 5 pairs from Characters dataset of several classification techniques (in bar charts) with different methods of featurizing the data (in color), mostly based on the way the landmark set $Q$ is chosen. Also, average KNN-classification test errors with various distances. Parameters: soft-dtw: $\gamma = 0.15$, LCSS: $\epsilon = 1$ and EDR: $\epsilon = 1, 2$

and got near zero test error. The selected pairs are the more challenging ones. Here $\varsigma = 100$ is considered for $v_Q^S$-featurization.

Aggregated test errors for 5 chosen pairs are given in Fig. 8. The KNN classifier using edit distance with real penalty (ERP) achieves the best misclassification rate. There is a considerable gap between the best misclassification rate of 1.67% and the next best performed classifier with 4.70% misclassification rate which is achieved by Gaussian SVM classifier employing $v_Q^S$-vectorization with voting technique. KNN with fastdtw also generated a low error rate of 4.73%. Other vectorized GSVM classifiers did well, but most others had misclassification rate higher than 6%, and often above 10%. Generically, however, the mistake-driven vectorized technique, as well as Vote($v_Q^S$) and endpoints, tends to work better than other landmark-based featurization methods across all classifiers, which are better than other KNN-based ones. The best classifiers among these are SVM-based ones, Random Forest, and Fréchet- or DTW-based KNN classifiers.

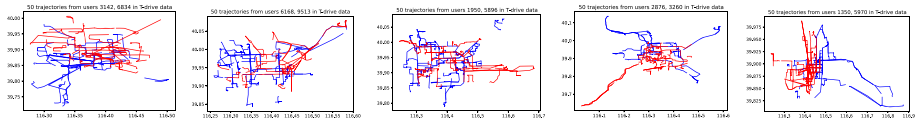

**Fig. 9** A subset of trajectories (50 trajectories of each user) of each pair of taxies from T-drive dataset are presented

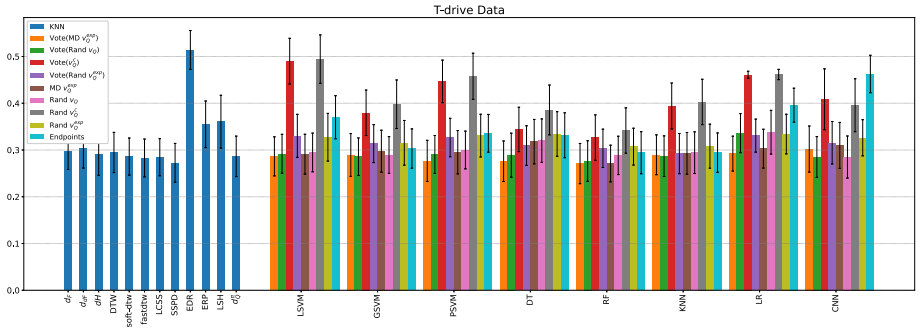


**Fig. 10** Average classification test errors of 5 pairs from T-drive Trajectory dataset of several classification techniques (in bar charts) with different methods of featurizing the data (in color), mostly based on the way the landmark set $Q$ is chosen. Also, average KNN-classification test errors with various distances. Parameters: soft-dtw: $\gamma = 1e-14$, LCSS: $\epsilon = 0.0055$ and EDR: $\epsilon = 0.005$

### 3.5 T-drive trajectory dataset

In this experiment we consider the T-drive Trajectory dataset released by Microsoft in 2011 [58, 59]. It consists of trajectories of 10,357 taxis captured by GPS within 6 months in Beijing. Each taxi's trajectories are stacked together so that each taxi has one very long trajectory. We have chosen 5 pairs of taxis ((3142, 6834), (6168, 9513), (1950, 5896), (2876, 3260), (1350, 5970)) with high misclassification rate (measured with both KNN classifier with different distances and featurization via $v_Q$ with 20 random landmarks). Then, after removing stationary waypoints, we partitioned each selected taxi's long trajectory to short trajectories with trip duration of at most 20 minutes and removed all trips with less than 10 and more than 200 waypoints. This way we ended up with between 100 and 200 trajectories for each taxi (see Table 1 for exact numbers and see Fig. 9).

The misclassification rates on test data are reported in Fig. 10. The best performing classifier is Random Forest with 27.1% error rate applying with/without voting technique with the mistake-driven landmarks. KNN with SSPD is the next well performed classifier with error rate of 27.26%. KNN with 4 distances, the mistake-driven algorithm with all classifiers but CNN, and voting technique with $v_Q$-vectorization with all classifiers but LR could get misclassification rates between 28% and 29%. We can easily observe that the mistake-driven technique tends to do a better job. Furthermore, in contrast to Characters dataset, we see that KNN classifier with ERP could not achieve a low test error.

### 3.6 Geolife GPS trajectory dataset

The dataset we employ here is Geolife GPS Trajectory dataset [39], which was released by Microsoft in 2012. It consists of trajectories of 182 users from 2007 to 2012. In total there are 17,621 trajectories which are mostly recorded in Beijing, China. Here we took 4

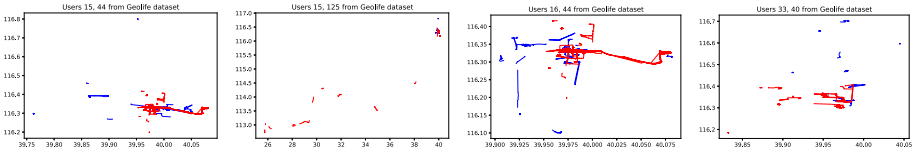


**Fig. 11** 4 pairs of users from Geolife trajectory data after applying 20 minutes partitioning

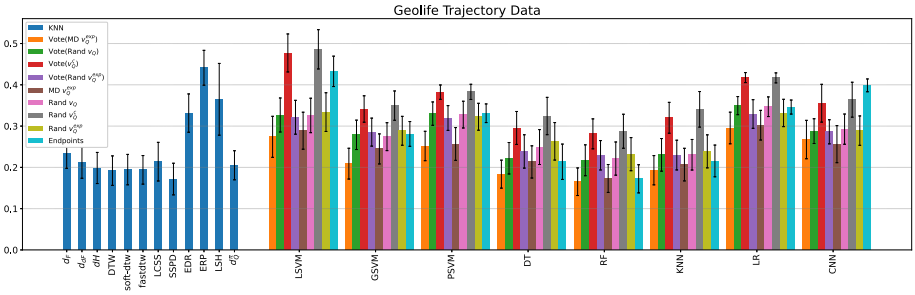


**Fig. 12** Average classification test errors of 4 pairs from Geolife GPS Trajectory dataset of several classification techniques (in bar charts) with different methods of featurizing the data (in color), mostly based on the way the landmark set $Q$ is chosen. Also average KNN-classification test errors with various distances on trajectories. Parameters: soft-dtw: $\gamma = 1e-15$, LCSS and EDR: $\epsilon = 0.001$

pairs of users {(15, 44), (15, 125), (16, 44), (33, 40)} with a high binary misclassification error rate using both KNN (with different distances) and $v_Q$ featurization via a random set of landmarks. Then we tried to classify trajectories of users in each pair from each other. In the preprocessing phase after removing stationary waypoints, like in T-drive dataset, we partitioned each trajectory to trips with a duration of at most 20 minutes, and then applied our standard filtering and cleaning. We ended up with users including between 100 and 200 trajectories (see Fig. 11).

Most raw trajectories were quite long and so classification with KNN using most distances is inefficient as their run time is quadratic in the number of waypoints of trajectories. Therefore, we used the 20 minutes threshold to partition trajectories into smaller and perhaps meaningful (sub-)trajectories. The landmark-based vectorization methods' complexities are linear in the number of waypoints of trajectories and thus very efficient.

The aggregated misclassification results for the chosen 4 pairs are given in Fig. 12. The best performed method is the mistake-driven algorithm with voting technique using Random Forest with 16.52% error rate on test data. The KNN classifier with SSPD, endpoint classification with Random Forest and the mistake-driven landmarks with Random Forest gained next lowest misclassification rates between 17% and 18%. In general, Random Forest tends to perform better than other classifiers across all vectorization methods. Moreover, like in T-drive dataset and unlike Characters dataset, on the Geolife trajectory dataset KNN-ERP performed poorly with a high misclassification rate of 44.13% on test data.

***Key observations*** A broad take away is that Random Forest and Gaussian SVM on vectorized representations are consistently among the best performing. A more detailed and holistic analysis in deferred to Sect. 5.

## 4 Including other features

Recall that a trajectory is a finite sequence of waypoints, which we transform into a curve connecting consecutive waypoints by line segments. In this section, we associate a time value $t_i$ with each waypoint as well – this is common but not universal in their collection. In addition to an *average length* of a segment, this allows us to define other features, namely *average velocity*, *acceleration* and *jerk*. Using these features, in addition to $v_Q$ and $v_Q^\varsigma$, we can have the following feature mappings which are a combination of $v_Q$ or $v_Q^\varsigma$ and average length, velocity, acceleration and jerk.

$$v_Q^+(\gamma) = (v_Q(\gamma), \text{length, velocity, acceleration, jerk}) \in \mathbb{R}^{|Q|+4},$$
$$v_Q^{\varsigma+}(\gamma) = (v_Q^\varsigma(\gamma), \text{length, velocity, acceleration, jerk}) \in \mathbb{R}^{|Q|+4}.$$

The motivation for introducing these physical features is that usually these features vary for different types of transportation modes. For example, the velocity and acceleration of a car is completely different from that of a train and both different from hiking or biking. Thus, these features will likely be useful in trajectory classification tasks for transportation data. Therefore, these feature mappings will be capable of capturing both geometric and physical aspects of trajectories. To be able to compare the contribution of geometric and physical attributes, we will evaluate misclassification rates with geometric features and physical features separately as well.

### 4.1 Car-Bus

We reconsider the car-bus dataset utilized in Sect. 3.1. We use the same preprocessing step here and try to classify car versus bus trajectories using 20 landmarks. As it can be observed from Fig. 13, the lowest misclassification rate of 1.95% is achieved by Linear SVM using $v_Q^{\varsigma+}$ featurization (i.e., the signed feature mapping $v_Q^\varsigma$ plus physical features). The next well performed method is again Linear SVM but with mistake-driven algorithm with 2.59% test error, where we combined the physical features with geometric features from landmarks. Recall that the best performance without physical features in Fig. 3 was 15.46% misclassification rate; so this is a significant improvement. Also an improvement over 3.89%, which is the best error rate with *only* physical features, which uses Linear SVM again. Looking at linear classifiers (Linear SVM and Logistic Regression) in Fig. 13 and misclassification rates reported in Fig. 3, one may conclude that adding physical features makes the car-bus data almost linearly separable. Other classifiers with all kinds of vectorizations have at least 5% error rate on test data. Figure 13 shows that on car-bus dataset the $v_Q^{\varsigma+}$ vectorization generically works better than other vectorization techniques in presence of physical features. $C = 100$ (LSVM and GSVM), $\gamma = auto$ (GSVM), $C = 1000$ and deg $= auto$ (PSVM), n_estimators $= 50$ (RF) and n_neighbors $= 5$ (KNN) are selected hyperparameters.

### 4.2 Transportation modes

The Geolife GPS Trajectory dataset used in Sect. 3.6 has a standard mode-of-transportation prediction task. Among 182 users 69 of them have labeled their trajectories with transportation modes such as walk, bike, bus, car, taxi, subway, railway, train, airplane, motorcycle, run. As other modes are in extreme minority in the labeled data, we only consider walk, bike, bus, car, taxi, subway, railway, and train. Moreover, as it is recommended in the user guide of

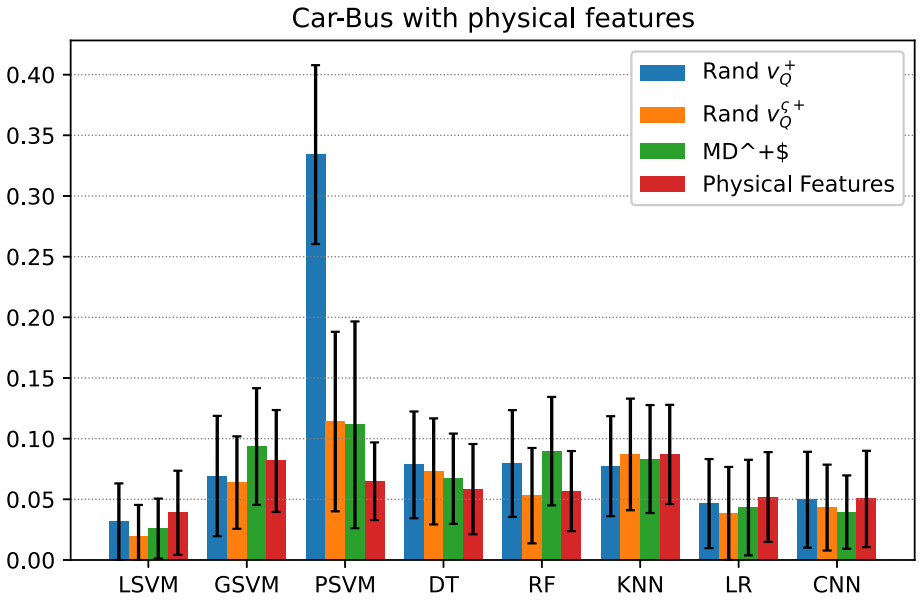


**Fig. 13** Car-Bus classification test errors with different classifiers and vectorizations

data, we regard car and taxi as one transportation mode, namely car, and similarly, we regard all three labels train, railway and subway as train. Therefore, we deal with 5 labels (walk, bike, bus, car, and train).

In this experiment we try to identify transportation modes of GPS trajectories in the data, so it is a multi-class classification task. First we apply our usual preprocessing step including removing stationary points and trajectories less than 10 waypoints. Since we would like to fairly compare our results with the results from state-of-the-art papers, we try to apply the experimental setup used in [26, 27], for example. Thus, in this experiment we use 80-20 train-test split. Moreover, each trajectory is partitioned into short trajectories if the time interval between two consecutive waypoints exceeds the 20 minutes threshold according to the recommendation in [32], which is also employed in the aforementioned references. The distribution of transportation modes after the preprocessing step is as follows: walk: 2060, bike 1110, bus: 1094, car: 975, train: 359, which in total are 5174 trajectories.

The misclassification rates on test data are provided in Table 4. The hyperparameter $\varsigma$, like in Sect. 3.6 is set to 1. As it can be viewed from Table 4, Random Forest with $v_Q^+$ vectorization achieves the best misclassification rate of 11.89%. Furthermore, Random Forest consistently performs better than other classifiers on all vectorization methods, where the next lowest test errors rates of 15.46% and 16.42% are obtained by Random Forest with MD$^+$ and $v_Q^{\varsigma+}$ featurizations. As it can be seen, the mistake-driven technique for choosing landmarks does not work as well as random choice of them. We believe this is because (a) the mistake-driven algorithm is designed for binary classification tasks but this task is a multi-class classification task, and (b) the mistake-driven algorithm helps optimize the landmarks, but does not factor in the physical characteristics which are critical for this task. Moreover, without using physical features, with random choice of landmarks we could achieve misclassification error rate of 18.08% using Random Forest whilst using only physical features the best gained test error rate is 22.73%. Therefore, the contribution of both physical and geometric features helps to improve the misclassification rate.

**Table 4** Transportation modes classification test errors of Labeled Geolife Trajectory dataset with different classifiers and different vectorizations. Hyperparameters: LSVM: $C = 1e10$, GSVM: $C = \gamma = 1000$, PSVM: $C = 1000$, deg $= 3$, RF: n_estimators $= 100$ and KNN: n_neighbors $= 5$

| **FM** | Rand $v_Q^+$ | Rand $v_Q^{S+}$ | MD$^+$ |
|---|---|---|---|
| **Clf** | **Mean (std)** | **Mean (std)** | **Mean (std)** |
| LSVM | 0.2869 (0.0139) | 0.4059 (0.0138) | 0.4416 (0.0178) |
| GSVM | 0.2068 (0.0126) | 0.5696 (0.0145) | 0.2251 (0.0160) |
| PSVM | 0.6335 (0.0326) | 0.5379 (0.0149) | 0.6292 (0.0114) |
| DT | 0.1973 (0.0116) | 0.2688 (0.0123) | 0.2289 (0.0131) |
| RF | **0.1189** (0.0094) | 0.1642 (0.0100) | 0.1546 (0.0085) |
| KNN | 0.2086 (0.0105) | 0.2611 (0.0111) | 0.2303 (0.0300) |
| LR | 0.6564 (0.1775) | 0.6840 (0.1552) | 0.6261 (0.0109) |
| **FM** | Rand $v_Q$ | Rand $v_Q^S$ | Physical Features |
| **Clf** | **Mean (std)** | **Mean (std)** | **Mean (std)** |
| LSVM | 0.4959 (0.0289) | 0.6772 (0.0141) | 0.3978 (0.0137) |
| GSVM | 0.2313 (0.0126) | 0.5711 (0.0150) | 0.2906 (0.0134) |
| PSVM | 0.6332 (0.0141) | 0.6537 (0.0225) | 0.7349 (0.0927) |
| DT | 0.2418 (0.0143) | 0.4167 (0.0126) | 0.2915 (0.0112) |
| RF | 0.1808 (0.0120) | 0.3022 (0.0120) | 0.2273 (0.0097) |
| KNN | 0.2241 (0.0131) | 0.5271 (0.0148) | 0.2464 (0.0106) |
| LR | 0.6183 (0.0233) | 0.6282 (0.0150) | 0.7449 (0.1371) |

Note that for two reasons we avoided doing KNN classification with different distances like in previous experiments. The main reason is that our aim here is to compare our methods' performance with state-of-the-art papers' results as a baseline. Furthermore, since the physical characteristic features are clearly important, yet it is not clear how to integrate them with other distances. The next reason is that most of the preprocessed trajectories are too long – even though the trajectories are partitioned by time – and thus the KNN approach can be very inefficient. We remark that the partitioning method in this experiment, as explained above, differs from the method used in binary classification task for Geolife and T-drive trajectory datasets in Sects. 3 and 4.

***Comparison with previous studies.*** To the best of authors' knowledge the best reported misclassification rate is 15.2% [26], which is achieved by intricately designed convolutional neural network. The average misclassification rate of 11.9% we achieved with our vectorized model, using $v_Q$ featurization plus the above 4 physical features, with Random Forest, outperforms this state-of-the-art result. In addition, notice that our method is very efficient and easy to apply as we use simple featurization and simple out-of-the-box machine learning algorithms. Moreover, we only need to tune a single benign hyperparameter in Random Forest for example, namely, the number of estimators, while in CNN one needs to tune many hyperparameters in addition to the design of a useful architecture.

To compare more, our method outperforms misclassification rates reported in many other studies like [28] with 32.1%, [30] with 25.9% and [31] with 25.8%. A summary of these misclassification rates is given in Table 5. Note that the experimental setup of each study is

**Table 5** Test error comparison with previous studies

| Study | Misclassification rate |
|---|---|
| Using CNN [28] | 32.1% |
| Using CNN [30] | 25.9% |
| Inference plus Decision Tree [31] | 23.8% |
| Using CNN [27] | 23.2% |
| Our Model with $v_Q$ vectorization | 18.1% |
| Our Model with $v_Q^{5+}$ vectorization | 16.4% |
| Our Model with $MDv_Q^+$ vectorization | 15.4% |
| Using CNN [26] | 15.2% |
| Our Model with $v_Q^+$ vectorization | 11.9% |

a bit different (we did our best to match the state-of-the-art results [26]) but all are trying to infer transportation modes.

## 5 Discussion

This paper establishes the first comprehensive and large-scale empirical comparison of methods to classify trajectories based on their spatial position. It compares KNN classifiers using standard distance measures, along with new techniques that first create a vectorized feature representation, and can then apply a wide variety of classifiers. Data for trajectories in these experiments mostly comes from human movement, potentially via a vehicle. On these tasks, the vectorized representations typically performed superior to the KNN-based ones across various distance measures. Moreover, the Random Forest classifier (as is common in other settings [60]) was typically the one that achieved the best or nearly best performance.

Other experiments considered different data sources including characters (written on a tablet), and other features including physical properties such as speed and acceleration. In these tasks, other methods sometimes showed better performance, specifically a KNN classifier using edit distance with real penalties [55] performed exceedingly well on the character classification tasks, and linear classifiers on the mode-of-transportation tasks. We did not specifically design tasks to take advantage of the distances or embeddings that preserved the orientation of direction of the trajectories (as has been done elsewhere [37], e.g., by modifying the Pigeons dataset [61–63]), and there was no significant observed advantage of that class of techniques over ones that simply treated trajectories as geometric objects. Identifying naturally occurring tasks where this is needed, and performing the empirical study is an intriguing future direction.

Beyond the large empirical study, we introduced a new mechanism to select the landmarks for the vectorized representations. These are dubbed *mistake-driven* and are inspired by the perceptron algorithm and other active learning paradigms. At each step one of the most misclassified training data trajectories is identified, and a new landmark is selected, with some randomness, near this trajectory with the goal of being informative toward its prediction. This approach is mostly effective, but can be noisy since (unlike the perceptron algorithm where a data element serves as a support vector) the selected landmark is not guaranteed to boost the accuracy of the identified trajectory. As a result, we also introduce a simple voting method to aggregate these mistake-driven collections of landmarks. This approach generates several

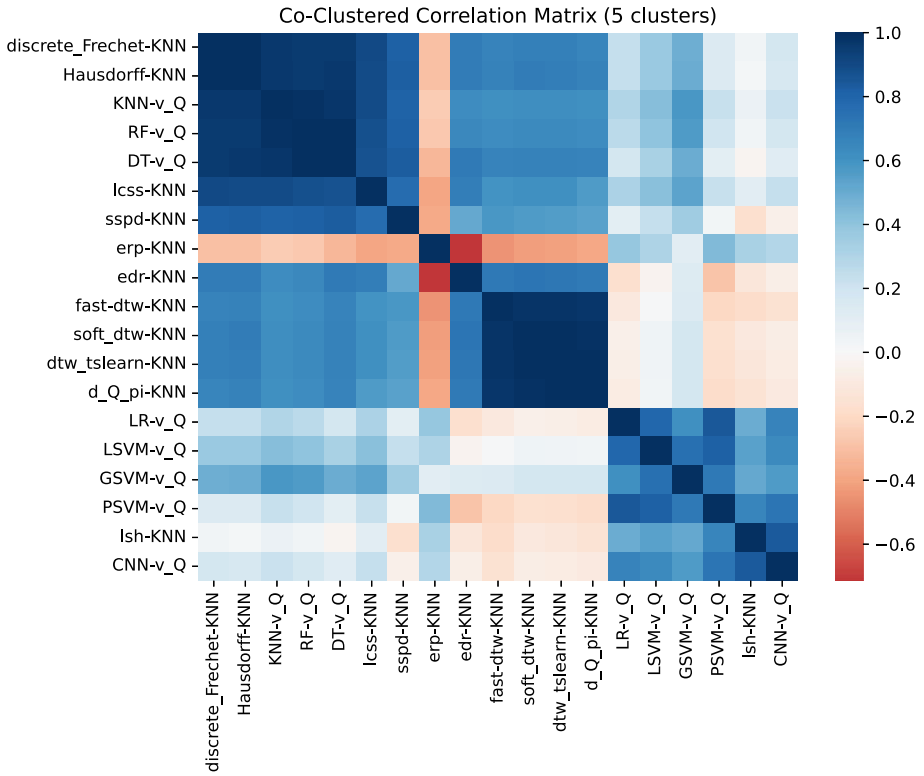


**Fig. 14** Co-clustered correlation matrix for test errors with 19 classifiers

landmark sets this way, or just chosen from normal distribution at random, and then classifies each trajectory by a majority vote of the classifiers from each landmark set. The combination of new methods in the mistake-driven landmarks and voting shows to be especially effective. Note that it is not obvious how to achieve similar voting-based improvements with the KNN-based classifiers for the various distances since the classifiers themselves are deterministic, and would thus all vote the same way.

We remark that we investigated (but do not present) several other potential data-driven mechanisms to choose landmarks. These include deriving the gradient for the position of a landmark and performing various gradient-descent-based approaches. These appeared to consistently get stuck in local minimum, and this line of attack proved unfruitful.

***Correlation among classifiers.*** This paper compared 20 different types of classifiers across 6 main experimental tables, each one may be aggregating several related tasks. Most of the discussion focused on which classifiers performed the best. However, many of the classifiers are fairly effective at many of the tasks. One lingering question is how correlated are the performances of similar classifiers – if they are clustered, in practice it may be useful to check one technique from each grouping. This we summarize in a correlation matrix.

Figure 14 shows the co-clustered correlation matrix of misclassification rates calculated for 217 experiments with the 19 classifiers discussed in Sect. 3. The correlation is calculated from the 6 main tables, where several experiments are aggregated in the figures. 200 more are included from 100 randomly chosen pairs from Geolife trajectory data, and 100 randomly chosen pairs from T-drive trajectory data. The big cluster in the figure indicates that Decision

Tree, Random Forest and KNN classifier with $v_Q$-vectorization and KNN with discrete Fréchet, Hausdorff, SSPD and LCSS are highly positively correlated. There is another cluster containing all variants of DTW and KNN with $d_Q^{\pi}$. Moreover, all variants of SVM, Logistic Regression and CNN with $v_Q$-vectorization and KNN with LSH have a positive correlation. The clear outlier is the KNN classifier for edit distance on real penalties (erp-KNN), including negative correlations with all but the vectorized SVM-based methods and CNN. The KNN classifier with continuous Fréchet distance is not included in Fig. 14 as its complexity is high and for most pairs in Geolife and T-drive trajectory datasets it could not be completed within 24 hours (remember that we have chosen more than 200 pairs from these two datasets).

***Efficiency.*** In general, the vectorized approaches are significantly more efficient and scalable. For instance, training a vectorized classifier on the entire car-bus dataset takes between 0.05 and 0.2 seconds, or a factor 11 more (typically under 2 seconds) with the mistake-driven approach which takes the best of 11 trials. In contrast the KNN-based approaches typically took 6 to 15 seconds, with only fastdtw (about 1.5 seconds) and LSH (0.08 seconds, using a similar sketch-based approach) nearly as efficient. On the Two Persons dataset where the trajectories are larger, the difference is more dramatic. Verctorized classifiers take 1 to 2.5 seconds, with mistake-driven approach again a factor 11 more. Where as KNN classifiers typically took 1,000 to 10,000 seconds; with exceptions fastdtw (32 seconds) and LSH (1.5 seconds). And among the KNN classifiers, the most divergent one in terms of performance (erp-KNN) was typically the slowest measured (Fréchet was even slower – more than 80, 000 seconds on Two Persons).

***Size of landmarks*** $|Q|$. To choose the right landmark size for experiments, we opted for 20 as we observed 20 landmarks are enough to get a good performance on most datasets. However, we did a simple experiment on the car-bus dataset for different values of $|Q|$ (10, 20, 30, 40 and 50). As it can be observed, generally speaking, increasing the number of landmarks slightly improves the test errors but kind of flattening at about 20. As an example, we have given the results with $v_Q$-vectorization in Fig. 15.

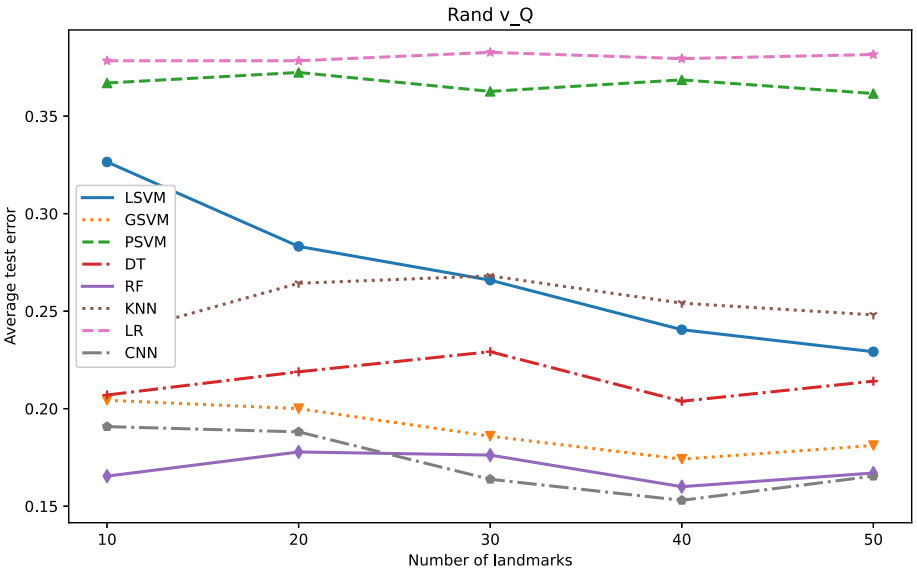


**Fig. 15** Classification test errors of Car-Bus data with 8 classifiers employing $v_Q$-vectorization and different number of random set of landmarks

***Final Recommendations*** Ultimately, for the consistent best accuracy, we recommend Random Forest (RF) with Vote(MD $v_Q$) features as a first choice to most likely achieve the best accuracy. However, RF with $v_Q$ features offers a very simple, effective, and efficient classifier. Still, one may want to try several approaches, for instance KNN with ERP. Finally, if one has meta data available, or physical characteristics (e.g., length, velocity, acceleration, jerk) make sense for the application, it is recommended to append them to the featurized representation.

**Acknowledgements** Jeff Phillips thanks his support from NSF CCF-1350888, CNS-1514520, CNS-1564287, IIS-1816149, CCF-2115677, and from Visa Research.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

1. Besse PC, Guillouet B, Loubes J-M, Royer F (2016) Review and perspective for distance-based clustering of vehicle trajectories. IEEE Trans Intell Transp Syst 17:3306–3317
2. Buchin K, Driemel A, Gudmundsson J, Horton M, Kostitsyna I, Loffler M ( 2019) Approximating $(k, l)$-center clustering for curves. In: SODA
3. Driemel A, Krivosija A, Sohler C ( 2016) Clustering time series under the Frechet distance. In: ACM-SIAM Symposium on Discrete Algorithms
4. Buchin K, Driemel A, van de L'Isle N, Nusser A ( 2019) klcluster: Center-based clustering of trajectories. In: Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 496– 499
5. Zhang Z, Huang K, Tan T ( 2006) Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In: 18th International Conference on Pattern Recognition. ICPR'06
6. Astefanoaei M, Cesaretti P, Katsikouli P, Goswami M, Sarkar R ( 2018) Multi-resolution sketches and locality sensitive hashing for fast trajectory processing. In: SIGSPATIAL
7. Cuturi M ( 2011) Fast global alignment kernels. In: Proceedings of the 28th International Conference on Machine Learning
8. Magdy N, Sakr MA, Mostafa T, El-Bahnasy K(2015) Review on trajectory similarity measures. In: IEEE Seventh International Conference on Intelligent Computing and Information Systems. ICICIS ( 2015)
9. Alt H, Knauer C, Wenk C (2004) Comparison of distance measures for planar curves. Algorithmica, 45–58
10. de Freitas NCA, da Silva TLC, de Macêdo JAF, Junior LM, Cordeiro MG ( 2021) Using deep learning for trajectory classification. In: Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021)
11. Garcia J, Concha OP, Molina JM, de Miguel G ( 2006) Trajectory classification based on machine-learning techniques over tracking data. In: IEEE 9th International Conference on Information Fusion
12. Lin W-Y, Hsieh C-Y (2013) Kernel-based representation for 2d/3d motion trajectory retrieval and classification. Pattern Recogn 46:662–670
13. Liua L, Liua F, Ky B (2019) Data mining-based model for motion target trajectory prediction. J Intell Fuzzy Syst 37:371–379
14. Sbalzarini IF, Theriot J, Koumoutsakos P ( 2002) Machine learning for biological trajectory classification applications. In: Proceedings of the CTR Summer Program
15. Sharma LK, Vyas OP, Schieder S, Akasapu AK ( 2010) Nearest neighbour classification for trajectory data. In: ICT: International Conference on Advances in Information and Communication Technologies
16. Xu W, Zhang Y, Lu J, Wang J (2011) Hdp-hmm-scfg: a novel model for trajectory representation and classification. Procedia Eng 15:629–633
17. Zhou F, Gao Q, Trajcevski G, Zhang K, Zhong T, Zhang F (2018) Trajectory-user linking via variational autoencoder. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence
18. Gao Q, Zhou F, Zhang K, Trajcevski G, Luo X, Zhang F ( 2017) Identifying human mobility via trajectory embeddings. In: AAAI
19. Junior AS, Renso C, Matwin S (2017) An active learning system for trajectory classification. IEEE Comput Graphics Appl 37(5):28–39

20. Soleymani A, Cachat J, Robinson K, Dodge S, Kalueff AV, Weibel R (2014) Integrating cross-scale analysis in the spatial and temporal domains for classification of behavioral movement. J Spatial Inf Sci 8:1–25
21. Patel D, Sheng C, Hsu W, Lee ML ( 2012) Incorporating duration information for trajectory classification. In: 28th International Conference on Data Engineering
22. Lee J-G, Han J, Li X, Cheng H (2011) Mining discriminative patterns for classifying trajectories on road networks. IEEE 9th Int Conf Inf Fusion 23(5):713–726
23. Dodge S, Weibel R, Forootan E (2009) Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. Comput Environ Urban Syst 33(6):419–434
24. Lee, J.-G., Han, J., Li, X., Gonzalez, H.: Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. In: Proceedings of the VLDB Endowment. ICPR'06 ( 2008)
25. Murray B, Perera LP (2022) Ship behavior prediction via trajectory extraction-based clustering for maritime situation awareness. J Ocean Eng Sci 7:1–13
26. Dabiri S, Heaslip K (2018) Inferring transportation modes from gps trajectories using a convolutional neural network. Transp Res Part C 86:360–371
27. Dabiri S, Lu C-T, Heaslip K, Reddy CK (2020) Semi-supervised deep learning approach for transportation mode identification using gps trajectory data. IEEE Trans Knowl Data Eng 32:1010–1023
28. Endo Y, Toda H, Nishida K, Kawanobe A (2016) Identifying different transportation modes from trajectory data using tree-based ensemble classifiers. Pacific-Asia Conf Knowl Discov Data Min 6:54–66
29. Fang S-H, Liao H-H, Fei Y-X, Chen K-H, Huang J-W, Lu Y-D, Tsao Y (2016) Transportation modes classification using sensors on smartphones. Sensors 16:1324
30. Wang H, Liu G, Duan J, Zhang L ( 2017) Detecting transportation modes using deep neural network. IEICE Trans Inf & Syst 100, 1132– 1135
31. Zheng Y, Chen Y, Li Q, Xie X, Ma W-Y ( 2008) Understanding mobility based on gps data. Proceedings of the 10th International Conference on Ubiquitous Computing. ACM 100, 312– 321
32. Zheng Y, Xie X ( 2008) Learning transportation mode from raw gps data for geographic applications on the web. Proceedings of the 17th World Wide Web Conference 86, 247– 256
33. Etemad M, Júnior AS, Matwin S ( 2018) Predicting transportation modes of gps trajectories using feature engineering and noise removal. In: Advances in Artificial Intelligence
34. Varlamis I ( 2015) Evolutionary data sampling for user movement classification, in evolutionary computation. In: IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan
35. Tragopoulou S, Varlamis I, Eirinaki M ( 2014) Classification of movement data concerning user's activity recognition via mobile phones. In: Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14), p. 42
36. Phillips JM, Tang P( 2019) Simple distances for trajectories via landmarks. In: ACM GIS SIGSPATIAL
37. Phillips JM, Pourmahmood-Aghababa H ( 2021) Orientation-preserving vectorized distance between curves. In: Mathematical and Scientific Machine Learning (MSML)
38. Cruz MO, Macedo H, Barreto R, Guimaraes A (2016) GPS Trajectories Data Set
39. Zheng Y, Fu H, Xie X, Ma W-Y, Li Q (2011) Geolife GPS Trajectory Dataset - User Guide
40. Duan H, Ma F, Miao L, Zhang C (2022) A semi-supervised deep learning approach for vessel trajectory classification based on ais data. Ocean Coast Manag 218:106015
41. Meng L, Zhang S (2020) Inferring travel modes from trajectory data based on hidden markov model. Int Conf Trans Dev 2020(7):95–103
42. Papadopoulos AN ( 2008) Trajectory retrieval with latent semantic analysis. In: Proceedings of the 2008 ACM Symposium on Applied Computing, pp. 1089– 1094
43. Pourmahmood-Aghababa, H., Phillips, J.M.: Classifying Spatial Trajectories (Python Implementation). https://github.com/aghababa/Classifying-Spatial-Trajectories
44. Alt H, Godau M (1995) Computing the fréchet distance between two polygonal curves. JCG Appl 5:75–91
45. Guillouet, B., Hinsbergh, J.V.: A Python Package for Computing Distance Between 2D Trajectories. https://github.com/bguillouet/traj-dist
46. Eiter T, Mannila H (1994) Computing discrete Frechet distance. Technical report, Christian Doppler Laboratory for Expert Systems
47. Hausdorff F (1914) Grundzüge der mengenlehre. Leipzig
48. Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. KDD Workshop 10:359–370
49. Salvador S, Chan P (2007) Fastdtw: Toward accurate dynamic time warping in linear time and space. Intell Data Anal 11:561–580
50. Tanida K Python Implementation of FastDTW. https://pypi.org/project/fastdtw

51. Cuturi M, Blondel M ( 2017) Soft-dtw: a differentiable loss function for time-series. In: Proceedings of ICML
52. Blondel M, Python Implementation of soft-DTW. https://github.com/mblondel/soft-dtw
53. Vlachos M, Gunopulos D, Kollios G ( 2002) Discovering similar multidimensional trajectories. In: ICDE
54. Chen L, Özsu MT, Oria V (2005) Robust and fast similarity search for moving object trajectories. In: SIGMOD, pp. 491– 502
55. Chen L, Ng R (2004) On the marriage of lp-norms and edit distance. Proceedings of the 30th International Conference on Very Large Data Bases (VLDB) 30, 792– 803
56. Khaiiate-Ajami H, Pourmahmood-Aghababa H, Phillips JM (2021) Trjtrypy. https://pypi.org/project/trjtrypy
57. Databases, in University of Illinois at Chicago MCL (2006) Real Trajectory Data. https://www.cs.uic.edu/~boxu/mp2p/gps_data.html
58. Yuan J, Zheng Y, Zhang C, Xie W, Xie X, Sun G, Huang Y (2010) T-drive: driving directions based on taxi trajectories. In: 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS
59. Yuan J, Zheng Y, Xie X, Sun G ( 2011) Driving with knowledge from the physical world. In: The 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining
60. Chen T, Guestrin, C ( 2016) Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785– 794
61. Meade J, Biro D, Guilford T( 2005) Homing pigeons develop local route stereotypy. In: Proceedings of the Royal Society B, vol. 272, pp. 17– 23
62. Mann R, Freeman R, Osborne M, Garnett R, Armstrong C, Meade J, Biro D, Guilford T, Roberts S (2011) Objectively identifying landmark use and predicting flight trajectories of the homing pigeon using gaussian processes. J R Soc Interface 8(55):210–219
63. Mann RP, Armstrong C, Meade J, Robin F, Biro D, Guilford T (2014) Landscape complexity influences route-memory formation in navigating pigeons. Biol Let 10:1020130885



**Hasan Pourmahmood-Aghababa** is a PhD candidate in Data Science in the School of Computing at the University of Utah. His current research focuses on machine learning, data mining and geometric data analysis. Hasan got his first PhD in mathematics (harmonic analysis) from Kharazmi University in Iran in 2010. He worked as an associate professor of mathematics at the University of Tabriz for 9 years.

**Jeff M. Phillips** is an Associate Professor in the School of Computing at the University of Utah. He is also the Director of the Utah Center for Data Science. He also directs the Data Science Educational. He works in geometric data analysis and has published a book called the Mathematical Foundations for Data Analysis.