Mobile Information Systems

Research Article

Using Trajectory Sub-clustering to Improve Destination Prediction

James Van Hinsbergh[1], Nathan Griffiths[1], Zhou Xu[2] and Alex Mouzakitis[2]

[1]Department of Computer Science, University of Warwick, Coventry, UK

[2]Jaguar Land Rover, Engineering Centre, Coventry, UK

Correspondence should be addressed to James Van Hinsbergh: j.van-hinsbergh@warwick.ac.uk

Abstract

Destination prediction is an active area of research, especially in the context of intelligent transportation systems. Intelligent applications, such as battery management in electric vehicles and congestion avoidance, rely on accurate prediction of the future destinations of a vehicle. Destination prediction methods can utilise mobility patterns, and can harness the latent information within vehicle trajectories. Existing approaches make use of the spatial information contained within trajectories, but this can be insufficient to achieve an accurate prediction at the start of an unfolding trajectory, since several destinations may share a common start to their trajectories. To reduce the prediction error in the early stages of a journey, we propose the Destination Prediction by Trajectory Sub-clustering method (DPTS) for iteratively clustering similar trajectories into groups using additional information contained within trajectories, such as temporal data. We show in our evaluation that DPTS is able to reduce the mean distance error in the first 40-60% of journeys. The implication of reducing the distance error early in a journey is that location-aware applications could provide more accurate functionality earlier in a journey. In this paper, we (i) propose the Destination Prediction by Trajectory Sub-clustering method (DPTS) by extending an existing destination prediction method through incorporating an iterative clustering stage to decompose groups of similar trajectories into smaller groups, and (ii) evaluate DPTS against the baseline performance of the existing method.

Keywords: Clustering; Classification; Vehicle GPS Data; Trajectory Mining; Destination Prediction

1 Introduction

Intelligent transportation systems can assist drivers and can benefit from having an accurate prediction of the destination in advance. This is a key motivation for research into methods for robust destination prediction which utilises patterns learnt from daily life. Destination prediction is also linked with traffic assessment, where intelligent vehicles or road-side units report the amount of congestion, [1, 2] and traffic flow efficiency, where cooperative routing occurs to minimise the congestion encountered on route [3–5]. Techniques in these areas can have mutual benefit when used together, such as to facilitate real-time re-routing [6, 7]. Besse *et al.* proposed a method for destination prediction [8], which we refer to as BDP (denoting Besse *et al.*'s Destination Prediction method), that uses

trajectory similarity classification. This is a technique that tries to predict which group of trajectories an unfolding trajectory is most likely to match. To calculate the trajectory groupings, Besse *et al.* use hierarchical agglomerative clustering with the Symmetrized Segment-Path-Distance (SSPD), an instance-based trajectory distance metric [9]. Their method uses either a simple unweighted score based on the GMM likelihood, or a weighted score that uses auxiliary variables (such as the hour-of-day and the day-of-week) and weighting functions to modify the score of each cluster. In this paper, we opt for using the unweighted BDP method as a baseline, so that the GMM likelihood score can be used directly without needing to define a weighting function for each auxiliary variable. The unweighted BDP method suffers from poor performance at the start of a journey, where limited spatial information is available. When only a small proportion of a journey has been completed there is only spatial information for the completed section, and since multiple journeys may originate at a single location and share an initial route, this makes it difficult to distinguish the destination early on. Other destination prediction methods exist in the literature, but either use external information from outside the vehicle, such as ground cover data or road type information, to improve predictive performance [10, 11], require knowledge of the identity of the driver [12, 13], or use a complex representation of road network [12–14]. In this paper, we assume that such information is not available and that the identity of the driver is unknown.

In this paper, we (i) propose the Destination Prediction by Trajectory Sub-clustering (DPTS) method, which extends BDP [8] by using additional data and an iterative sub-clustering approach to decompose trajectory clusters into more specific groupings, and (ii) we evaluate DPTS against the baseline performance of BDP (with the unweighted score). A key difference of DPTS from BDP is the use of iterative sub-clustering that can take multiple metrics and their respective parameters, performing iterations of clustering. In contrast, BDP uses a single iteration of clustering with the SSPD metric alone. DPTS can be easily extended by adding additional metrics and iterations to the clustering process, and by varying the order of iterations.

This paper is organised as follows. Section 2 reviews the related work, Section 3 presents the DPTS method, and Section 4 introduces our experimental methodology and the datasets used for evaluation. Section 5 presents the results of applying DPTS to vehicle trajectories, and compares the performance to that achieved by the baseline unweighted BDP. Finally, Section 6 outlines future work and concludes the paper.

2 Related Work

Human mobility is a broadening field of research [15–18], which provides useful analysis that can influence areas such as urban transportation planning. Understanding human mobility patterns can have a significant impact on numerous applications, including destination prediction. Schneider *et al.* note that the average person only visits a small number of locations on a daily basis, and that 90% of the population visit less than 7 distinct locations per day, according to the surveys analysed in their work [18]. Patterns in human mobility can be modelled as human mobility motifs, which are abstractions of activity patterns, such as a home-to-work based tour [18–22]. By analysing mobile phone data, Schneider *et al.* identified 17 daily motifs which account for 90% of the recorded trips in their data [18]. Büscher *et al.* investigated the stability of the most common motifs over time [21], and Li *et al.* investigated infrequent motif detection [23]. Travel diaries, and the trends or behaviours learnt from analysing them have also been widely researched [24–27]. Multiple studies claim that there are common days for various activities or tasks [24, 26, 28], with an increased stability of people's travel behaviours on work days [25]. Seasonal, geographical, economic and cultural factors all have an impact on people's activity patterns [26–28], in addition to access to and availability of public transport and personal vehicles [29, 30].

Destination prediction has been the subject of much research, with recent work using historical GPS trajectories in order to predict an individual's next location or final destination. Markov models are widely used for destination prediction [31–33], with some methods considering multiple transport modes [13, 34–36] while others focus on vehicle trajectories [12, 37]. Other approaches include Bayesian inference [10, 11, 14, 38] to predict the intended destination of an individual, Gaussian mixture models [8, 39], decision tree learning [35, 40, 41], and support vector machines [6]. Existing research into destination prediction has shown that consideration of temporal aspects, such

as the day-of-week or the hour-of-day, can improve predictive performance [10, 12]. Research into predicting an individual's next location is similar to destination prediction, but focusses on predicting the next intermediate location, rather than the final destination. Ziebart *et al.* provide a good example of next location prediction, where they predict individual turns along a trajectory [11].

Several destination prediction methods use a grid-based approach [10, 11, 37] or have a graph representation of the road network [12, 14, 42]. However, this is more computationally expensive than solely using the raw GPS instances, and often requires external data to enable pre-processing, such as map matching and GIS data [42]. Clustering is frequently used as an initial step in destination prediction, translating stay points, which are instances of low mobility, into places [8, 13, 43, 44].

There has been some research that attempts to address the data sparsity problem. For example, Xue *et al.* propose a method called Sub-Trajectory Synthesis (SubSyn), that decomposes trajectories into smaller segments and connects these to adjoining segments, creating synthetic trajectories [37]. This greatly increases the number of possible trajectories that can be modelled from an input dataset, since it is rare to have an exhaustive set of input trajectories available. The available data can also be increased by considering multiple individuals, and how individuals complement each other, with similar trajectories increasing the value of the training data [45].

Krumm *et al.* [10] and Xue *et al.* [37] both use a 1km grid-based approach, and Ziebart *et al.* evaluate their PROCAB algorithm on multiple grid sizes [11]. A coarse grid improves the matching performance, but also increases destination prediction error, since the grid squares span a larger area. The opposite is seen with a fine grid, and therefore an appropriate grid size should be selected to achieve an acceptable trade-off between trajectory matching performance and destination prediction error. A poor choice of grid size may cause separate destinations to be grouped. The grid representation is extended in work by Chen *et al.* in which grid cells are merged together where adjacent cells have similar routes [28, 46]. The WhereNext algorithm uses a similar approach, in which Monreale *et al.* propose T-Patterns, which are sequences of regions [41].

Alternatives to grid-based approaches include map matching or generating local graph representations of the road network [12, 14, 42]. Simmonds *et al.* use a mapping database to provide a road graph, in which link-goal pairs can be formed [12]. Their model can predict the next link, and subsequently infer the final destination [12]. Karimi *et al.* construct a tree-based structure using additional road-network data, alongside amenity information [42]. Similarly, Patterson *et al.* also propose a graph-based representation, which is constructed from a street map provided by the US Census Bureau [14].

Clustering techniques are used in many approaches as a means of extracting locations, with k-means [13, 32], hierarchical [8], and density-based [43] clustering being used. Choi *et al.* adopt the k-means approach to cluster trajectory segments [32], similar to Ashbrook and Starner who map significant locations into clusters [13]. Ashbrook and Starner use a graph comparing the number of clusters to the number of locations, locating the knee point in order to select a suitable number of clusters [13, 47]. This has proven to be a popular method, and similarities are seen in several related approaches [10, 12, 14, 34, 38, 48]. Cho *et al.* extract intermediate instances by using a more computationally expensive Gaussian-means approach [35]. Conversely, Gambs *et al.*, use a density-based approach to generate the corresponding locations from their input data [36].

To train methods for destination prediction the first step is to separate the training data into distinct trips or trajectories, such that the first instance of a trajectory is the start location and the final instance is the destination. Time thresholding is often used for this task [10, 13, 34], where the threshold is a minimum duration between two consecutive recorded instances (and instances are not recorded if an individual stays in the same place). Chen *et al.* [40] use a threshold of 2 minutes, Krumm *et al.* [10] and Alvarez *et al.* [34] use a threshold of 5 minutes, while Ashbrook and Starner opt for a 10 minute threshold [13]. The threshold value used varies, implying that it is non-trivial to find a suitable value. In the dataset collected and used in this paper, we avoid the need to use time thresholding, since trips are naturally segmented when the vehicle from which trajectories are collected is powered down.

Approaches to destination prediction also have varying input data, with some only using spatial information from within trajectories [8], while others use multiple external sources [10, 42]. For example, Krumm *et al.* use

ground cover data [10], vehicle speed is used by Fukano *et al.* [48], Karimi *et al.* use data on local amenities [42], and others use temporal data [6, 10, 12]. Temporal data, such as the day-of-week or the hour-of-day, can act as an indicator of the next location, and have been shown to improve predictive performance [12]. In this paper, we aim to avoid the dependency on external data and, since temporal data is implicitly available within a trajectory record, our approach will focus on the use of data that is naturally contained within trajectory data.

Besse *et al.* propose a destination prediction method (which we refer to as BDP), which uses distribution-based models to match similar trajectories [8]. The training trajectories are grouped using hierarchical agglomerative clustering, with the distance between trajectories computed by the Symmetrized Segment-Path-Distance (SSPD) [9]. Using the clustered trajectories, Besse *et al.* train 2D Gaussian Mixture Models over each cluster, using the latitude and longitude to fit a distribution to a sample of training coordinates. Once these models are trained, a likelihood can be assigned for each cluster in an unfolding trajectory, and its destination is predicted using the centroid of the most likely cluster. Besse *et al.* also propose using a weighted score, which uses auxiliary variables, such as the hour-of-day, with each variable associated with a weighting function to modify the GMM likelihood. Our proposed method avoids the need for defining such weighting functions and is easily extensible in terms of adding additional variables. Our method also results in smaller clusters that naturally take the auxiliary variables into account, which can be beneficial for interpreting predictions. Since our focus is on identifying suitable clusters from which to make predictions, we evaluate DPTS against BDP with the unweighted simple score. BDP has several benefits over other methods since it does not rely on external data [10, 42], it does not require a mapping of the road network, which is computationally expensive to process [12, 14], and it does not discretise the space into a grid representation [10, 11, 37]. While Besse *et al.* propose the use of auxiliary variables, these variables do not segregate the trajectories into more specific clusters, unlike the proposed DPTS method. Gaps in the literature also exist where trajectories could be clustered into more specific groupings, using criteria such as spatio-temporal attributes. While not the main aim of this paper, our proposed method, DPTS, is extensible by design and allows multiple attributes to be used to narrow down specific trajectory groupings.

3 Destination Prediction by Trajectory Sub-clustering (DPTS)

The motivation behind Destination Prediction by Trajectory Sub-clustering (DPTS) is to reduce the distance error in destination prediction using vehicle data, specifically when making predictions in the early stages of a journey. We define the distance error as the Haversine (or spherical) distance between the actual and predicted destination. Reducing the distance error improves confidence in the correctness of the predicted destination, which can in turn improve location-aware applications, such as recommendations for which routes to avoid [3, 4], locations of electric vehicle charging points [49, 50], and so on. In this paper, we define a trajectory, t, as a strictly ordered sequence of instances $[x_1, ..., x_{|t|}]$, in which an instance is a latitude, longitude, and a timestamp. We hypothesize that the distance error in prediction can be reduced by using the additional data that is contained within the trajectories, such as temporal data or vehicle sensor data, including the vehicle speed and status of doors. Using this additional data, we can group trajectories into more specific clusters than those of BDP, enabling us to (i) lower the average distance between the trajectories within a cluster (and since destination prediction uses cluster centroids, a lower average distance has the potential to reduce the average error), and (ii) improve the prediction of which cluster an unfolding trajectory belongs to. In this paper, we focus on using temporal data from within the trajectories. However, our method is data agnostic, and can be used with different input data depending on the application and the data available. For example, the number of passengers in a vehicle (obtained from seatbelt status data) could be used as an input to help separate and predict trajectory clusters. We evaluate DPTS using the temporal properties of trajectories to decompose the clusters, in addition to the spatial information, using data that is implicitly available in the time signal associated with each instance in a trajectory. The evaluation in this paper assumes that the raw time signal in a trajectory can be translated into a suitable format, e.g., from a unix timestamp to a date and time.

Table 1: Notation used in this paper.

Notation	Description
$\mathcal{T} = \{t_1, ..., t_n\}$	A set of n trajectories
$\mathcal{T}_{c_j}$	A set of trajectories in cluster c_j
$t = [x_1, ..., x_{\|t\|}]$	A trajectory within $\mathcal{T}$, a strictly ordered sequence of $\|t\|$ instances
$x_i = \langle lat, long \rangle$	An instance x_i is a latitude and longitude position, $lat, long$, at time i
D	A dissimilarity matrix for the input set of trajectories
$\mathcal{C} = \{c_1, ..., c_n\}$	A set of n clusters extracted from trajectories in $\mathcal{T}$
$\Phi = \{\phi_1, ..., \phi_n\}$	A set of n GMMs trained for each cluster
ϕ_j	A pre-trained GMM for the cluster c_j
P	A probability value for a trajectory, t, fitting the most likely GMM in Φ
M	A sparse matrix containing the clustering parameters, with an implicit ordering
α	A parameter value for current iteration of clustering
d	The distance function for clustering, which is used to calculate D
κ	The maximum number of components for a GMM
μ	The maximum number of instances for training a GMM
ψ	The best (lowest) Bayesian Information Criterion value for a GMM
θ	The parameter value to use for the decision threshold, i.e., a probability value in the static mode and a multiplier in the dynamic mode
$\theta_{dynamic}$	A boolean flag indicating whether to use the dynamic (true) or static (false) mode for the decision threshold

3.1 Overview & Definitions

DPTS begins by performing an initial clustering of the trajectories, akin to that in BDP. The trajectories are clustered using hierarchical agglomerative clustering, using pairwise dissimilarity matrices. For spatial dissimilarity, we adopt the approach taken by BDP, which uses the Symmetrized Segment-Path Distance (SSPD) to generate the dissimilarity matrices [9]. SSPD uses the Segment-Path distance, which is calculated as the mean of all distances from the points composing the trajectory, t_1, to the trajectory, t_2 [9]. SSPD is calculated as the mean of the sum of the Segment-Path distance from t_1 to t_2 and the Segment-Path distance from t_2 to t_1. For temporal similarity, we focus on two properties, the day-of-week and the hour-of-day. These temporal properties are only considered for the first instance in a trajectory, unlike the spatial similarity which considers each instance within a trajectory. This is done to minimise the required computation, since the start instance is a key temporal indicator. We define two functions, $encodeDay(t)$ and $encodeHour(t)$, which when given an input trajectory, t, convert the time of the first instance into encoded values for the day-of-week and hour-of-day respectively. Since our approach uses hierarchical agglomerative clustering, dissimilarity matrices are required for both the day-of-week and hour-of-day. To create these dissimilarity matrices, we use the following definitions of how the differences in the day-of-week and hour-of-day are calculated.

Definition 1. *The difference in day-of-week between trajectories t_1 and t_2 is defined as:*

$$diff_{day}(t_1, t_2) = min(abs(encodeDay(t_1) - encodeDay(t_2)), \\ 7 - abs(encodeDay(t_1) - encodeDay(t_2))) \quad (1)$$

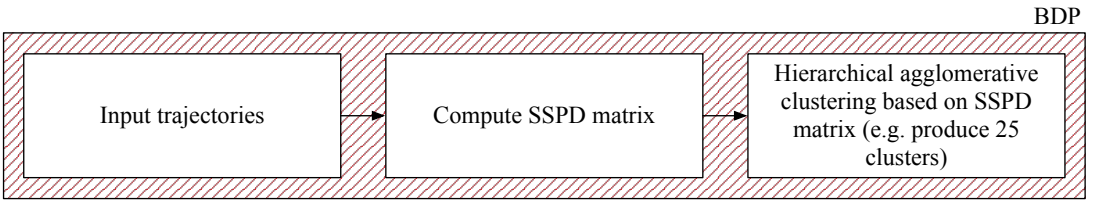
Definition 2. *The difference in hour-of-day between trajectories t_1 and t_2 is defined as*

$$diff_{hour}(t_1, t_2) = min(abs(encodeHour(t_1) - encodeHour(t_2)), \\ 24 - abs(encodeHour(t_1) - encodeHour(t_2))) \quad (2)$$

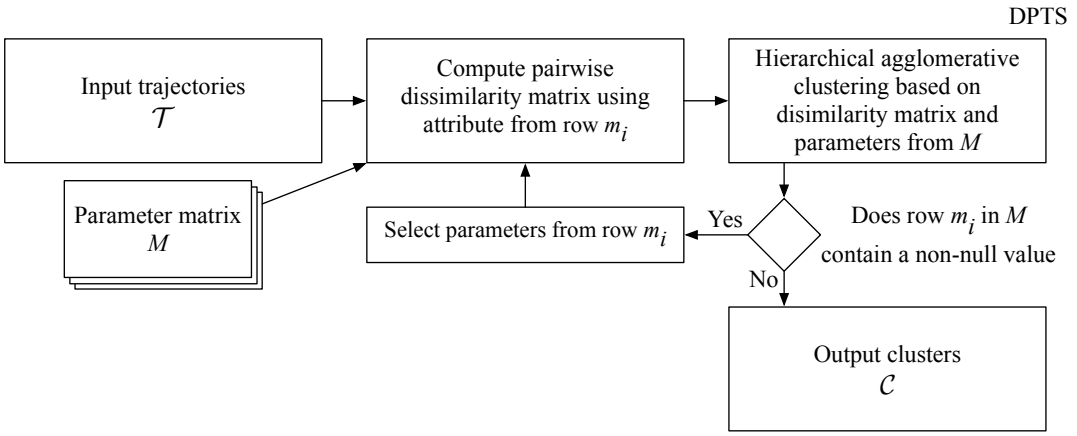
Figure 1 gives a high-level overview of the proposed DPTS methodology, showing how clusters are generated, and highlighting the differences between DPTS and BDP (using the unweighted score). In particular, BDP clusters

Table 2: Functions used when defining DPTS.

Function	Description
$generateDissimilarityMatrix(\mathcal{T}, d)$	Computes a pairwise dissimilarity matrix for all trajectories, $\mathcal{T}$, according to the distances calculated by the distance function, d
$hierarchical(\mathcal{T}, D, \alpha)$	Performs hierarchical agglomerative clustering on trajectories, $\mathcal{T}$, using the dissimilarity matrix, D, according to the clustering criteria, α
$getTrajectoriesFromCluster(c_j)$	Returns the set of trajectories within cluster c_j
$likelihood(x_i, \phi_j)$	Calculates the log-likelihood for instance, x_i, to fit the GMM, ϕ_j
$softmax(likelihood)$	Performs the softmax function on the log-likelihood, $likelihood$
$clusterCentroid(cluster)$	Returns the centroid of the cluster, $cluster$
$BIC(\phi_j)$	Obtains the Bayesian Information Criterion for GMM, ϕ_j
$trainGMM(comp, instances)$	Trains a GMM using $comp$ components with the given $instances$
$extractInstances(c_j, features)$	Extracts all instances, containing the selected $features$ from cluster c_j
$chooseRandom(instances, n)$	Selects n random instances from $instances$
$getDistanceFunction(m_i)$	Returns the distance function for the non-null entry in row m_i of the matrix of clustering parameters, M
$getClusteringParameter(m_i)$	Returns the clustering parameter for the non-null entry in row m_i of the matrix of clustering parameters, M
$isEmpty(m_i)$	Returns true if there are only null entries in row m_i of the matrix of clustering parameters, M
$encodeDay(t)$	Returns the encoded day-of-week, [0..6], on which the trajectory t started, s.t. 0 is a Monday and 6 is a Sunday.
$encodeHour(t)$	Returns the encoded hour-of-day, [0..23], on which the trajectory t started, s.t. 0 is 12am and 23 is 11pm.



(a) An overview of the BDP algorithm on which DPTS is based.



(b) Overview of the DPTS methodology for generating sub-clusters.

Figure 1: Overview of the BDP algorithm and the DPTS methodology.

the input trajectories on spatial distance using SSPD, whereas in DPTS there is an iterative process, in which clustering occurs according to the rows of a parameter matrix, M.

Definition 3. *A DPTS parameter matrix, M, is a sparse matrix in which each row corresponds to a single clustering iteration, each column corresponds to a clustering attribute, and each entry the parameter used.*

$$M = \begin{array}{c} \\ iteration_1 \\ iteration_2 \\ \vdots \\ iteration_{i-1} \\ iteration_i \end{array} \begin{array}{c} \begin{array}{cccc} attribute_1 & attribute_2 & ... & attribute_j \end{array} \\ \begin{bmatrix} & m_{1,2} & \dots & \\ m_{2,1} & & \dots & \\ \vdots & \vdots & \ddots & \vdots \\ & & \dots & m_{i-1,j} \\ & & \dots & \end{bmatrix} \end{array} \tag{3}$$

We define a DPTS parameter matrix, M, as a sparse matrix in which the column headings correspond to the available attributes on which to cluster, and the rows implicitly indicate which attribute is used for clustering in a given iteration and the parameter value to be used (see Definition 3). An attribute is comprised of two parts, namely the signal that is used, such as SSPD, and the measure to be used, such as the maximum cluster criterion. Each row corresponds to a single iteration of the hierarchical clustering process (ordered $1 \dots i$), such that a row contains at most a single non-null entry, $m_{i,j}$ denoting the parameter value, m, to be used in iteration i of the hierarchical clustering using the attribute corresponding to column j. The number of columns correspond to the number of clustering attributes considered, and the number of rows corresponds to the number of iterations, plus one null row. The final row only contains null entries, which is interpreted as being the termination criteria for clustering. We define two functions to access entries in a parameter matrix, namely, $getDistanceFunction(m_i)$ and $getClusteringParameter(m_i)$. Both functions take as input a single row of the matrix, m_i, and identify a non-null column, such that $getDistanceFunction(m_i)$ returns the distance function corresponding to this non-null column and $getClusteringParameter(m_i)$ returns the entry in the column. Both of these functions are undefined for a row containing only null entries.

In our evaluation, discussed later in Section 5, we consider three different signals for clustering, namely SSPD, the difference in day-of-week (using Equation 1) and the difference in hour-of-day (using Equation 2). We use the maximum cluster criterion as the measure for the SSPD signal (adopted from BDP [8]), and the distance criterion as the measure for the difference in day-of-week and the difference in hour-of-day. These attributes are denoted *msspd*, *ddow* and *dhod* respectively. In this paper, our evaluation uses each attribute a maximum of once, meaning that a maximum of 3 clustering iterations are performed. An example parameter matrix is shown in Example 1, which will cause DPTS to perform 3 iterations of clustering. The first iteration will use SSPD with a clustering parameter of 25, followed by the hour-of-day with a parameter value of 6 and the final iteration will use the day-of-week, with a parameter value of 2. An illustration of representing the clustering performed in BDP using a DPTS parameter matrix is shown in Example 2. Since BDP only uses SSPD for clustering with a single clustering iteration, the parameter matrix, M_{BDP}, only has a single non-null entry in the top-left cell.

Example 1. *An example parameter matrix, M_{DPTS}, for DPTS.*

$$M_{DPTS} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} msspd & ddow & dhod \end{array} \\ \begin{bmatrix} 25 & & \\ & & 6 \\ & 2 & \\ & & \end{bmatrix} \end{array}$$

Example 2. *A representation of example BDP algorithm parameters in the form of a DPTS parameter matrix, M_{BDP}.*

$$M_{BDP} = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} msspd & ddow & dhod \end{array} \\ \begin{bmatrix} 25 & & \\ & & \end{bmatrix} \end{array}$$

Algorithm 1: $performCluster(\mathcal{T}, M)$

inputs : $\mathcal{T}$, a set of n trajectories, $\{t_1, ..., t_n\}$
M, a $i \times j$ parameter matrix
outputs: $\mathcal{C}$, the set of clusters extracted from all trajectories in $\mathcal{T}$

$\mathcal{C} = \emptyset$
// for each row m_i in M
for $m_i \in M$ **do**
  if $isEmpty(m_i)$ **then**
    break
  end
  $d = getDistanceFunction(m_i)$
  $\alpha = getClusteringParameter(m_i)$
  if $\mathcal{C} == \emptyset$ **then**
    $D = generateDissimilarityMatrix(\mathcal{T}, d)$
    $\mathcal{C} = hierarchical(\mathcal{T}, D, \alpha)$
  else
    $\mathcal{C}' = \emptyset$
    for $c_j \in \mathcal{C}$ **do**
      // for each initial cluster, generate a set of sub-clusters
      $\mathcal{T}_{c_j} = getTrajectoriesFromCluster(c_j)$
      $D = generateDissimilarityMatrix(\mathcal{T}_{c_j}, d)$
      $\mathcal{C}_{m_i} = hierarchical(\mathcal{T}_{c_j}, D, \alpha)$
      // add each sub-cluster to the final output
      $\mathcal{C}' = \mathcal{C}' \cup \mathcal{C}_{m_i}$
    end
    $\mathcal{C} = \mathcal{C}'$
  end
end
// return the set of clusters
return $\mathcal{C}$

3.2 The training stage of DPTS

Algorithm 1 details the approach used to generate the clusters. Given a set of input trajectories, $\mathcal{T}$, and a parameter matrix, M, the algorithm starts by selecting the distance function, d, and hierarchical clustering parameter, α, from the first row, m_1, in M. The distance function, d, is then used to compute a dissimilarity matrix, D, over the trajectories, $\mathcal{T}$. Hierarchical agglomerative clustering is then performed using the dissimilarity matrix, D, and the clustering parameter, α, to generate a set of clusters. For example, using the parameter matrix from Example 1, the initial dissimilarity matrix would be computed using the SSPD distance function, and the subsequent hierarchical agglomerative clustering would generate up to 25 clusters. For each further iteration of clustering, represented by the rows m_i in M, dissimilarity matrices are computed over the trajectories, $\mathcal{T}_{c_j}$, in each cluster, c_j, in the current set of clusters, i.e., $c_j \in \mathcal{C}$. These dissimilarity matrices are used to generate a further set of clusters, $\mathcal{C}_{m_i}$. Each new set of clusters, $\mathcal{C}_{m_i}$, generated over the current clusters, is appended to $\mathcal{C}'$, for use in the following iteration. This process is repeated for each of the clustering iterations specified in the parameter matrix, M, updating the current set of clusters $\mathcal{C}$ at the end of each iteration with the newly calculated clusters $\mathcal{C}'$. Once the current row of

Algorithm 2: $trainClassifiers(\mathcal{C}, \kappa, \mu, features)$

inputs : $\mathcal{C}$, a set of clusters containing trajectories
κ, the maximum number of components to use for each GMM
μ, the maximum number of instances to select
$features$, the features to train the GMM with

outputs: Φ, a set containing the trained GMMs for each cluster in $\mathcal{C}$

for $j \in [1, |\mathcal{C}|]$ **do**
  $\psi, \phi_j^* = \infty, null$
  `// get all features vectors for all trajectories within cluster`
  `// e.g. the vector for sspd is lat,long`
  $instances = extractInstances(c_j, features)$
  `// pick random sample of` μ `instances from trajectories`
  $instances = chooseRandom(instances, \mu)$
  `// for number of components in 1 to` κ
  for $comp \in [1, \min(\kappa, |instances|)]$ **do**
    $\phi_j = trainGMM(comp, instances)$
    `// use Bayesian Information Criterion,` ψ`, to select model`
    if $BIC(\phi_j) < \psi$ **then**
      `// update best gmm,` ϕ_j^* `for cluster` c_j `if better`
      $\phi_j^* = \phi_j$
      $\psi = BIC(\phi_j)$
    end
  end
  $\Phi = \Phi \cup \phi_j^*$
end
`// return trained GMMs`
return Φ

the parameter matrix contains only null entries, the algorithm terminates and returns the clusters resulting from the final iteration of clustering.

A set of GMMs, Φ, are trained on the resulting clusters, as specified in Algorithm 2. In DPTS, a feature vector is used to define the features for training the GMMs. In this paper, we consider the latitude, longitude, encoded day-of-week and encoded hour-of-day. In BDP, the latitude and longitude of an instance are the only features used in the GMM. When using a weighted score, Besse *et al.* use additional variables, such as encoded day-of-week and encoded hour-of-day, and weighting functions to modify the likelihood score, but these variables are not used to sub-divide clusters of trajectories. Our approach can also be extended to include additional data, for example the vehicle signals that are included in each instance, x_i. For each cluster, c_j, all instances from the trajectories within c_j are extracted, containing the features in the provided feature vector. A sample of these instances is selected uniformly at random and without replacement according to the parameter, μ, which controls the maximum number of instances to select. If the number of instances in c_j is less than μ, then all instances are selected. GMMs are built starting with a single component, up to the minimum of either the κ parameter or the number of instances, in increments of 1. Each GMM with an increased number of components, ϕ_j, trained on the selected instances of c_j is evaluated using the Bayesian Information Criterion (BIC) [51], and if it has a lower BIC than the best BIC observed so far, then the best GMM, ϕ_j^*, and its BIC, ψ, are updated with the current values. This is repeated for every cluster, c_j, in the set of clusters output from the clustering stage, $\mathcal{C}$, and the trained GMMs are returned in a set, Φ.

Algorithm 3: Training stage of DPTS

inputs : $\mathcal{T}$, a set of n trajectories, $\{t_1, ..., t_n\}$
M_{DPTS}, a $i \times j$ parameter matrix for DPTS
M_{BDP}, a $i \times j$ parameter matrix representing BDP
κ, the maximum number of components to use for each GMM
μ, the maximum number of instances to select
$features$, the features to train the GMM with

outputs: Φ_{BDP}, a set containing the trained GMMs for each cluster in $\mathcal{C}_{BDP}$
Φ_{DPTS}, a set containing the trained GMMs for each cluster in $\mathcal{C}_{DPTS}$

1 $\mathcal{C}_{BDP} = performCluster(\mathcal{T}, M_{BDP})$
2 $\Phi_{BDP} = trainClassifiers(\mathcal{C}_{BDP}, \kappa, \mu, \langle lat, long \rangle)$
3 $\mathcal{C}_{DPTS} = performCluster(\mathcal{T}, M_{DPTS})$
4 $\Phi_{DPTS} = trainClassifiers(\mathcal{C}_{DPTS}, \kappa, \mu, features)$
`// return matrices of trained GMMs`
5 **return** Φ_{BDP}, Φ_{DPTS}

The overall training stage of DPTS is detailed in Algorithm 3, in which the GMMs are trained. This method takes six parameters: (i) a set of training trajectories, $\mathcal{T}$, (ii) a parameter matrix, M_{DPTS}, (iii) a parameter matrix for BDP, M_{BDP}, (iv) the maximum number of components to consider for each GMM, κ, (v) the maximum number of instances to select when training a GMM, μ, and, (vi) the set of features to use to train the GMMs. The training stage returns two sets of GMMs, Φ_{BDP} and Φ_{DPTS}, containing the trained GMMs for each cluster in $\mathcal{C}_{BDP}$ and $\mathcal{C}_{DPTS}$ respectively.

The training stage first clusters all trajectories in $\mathcal{T}$, using the parameters defined in M_{BDP}, and trains a set of GMMs, Φ_{BDP}, for each cluster in $\mathcal{C}_{BDP}$ using only the latitude and longitude, $\langle lat, long \rangle$, in the feature vector (see Algorithm 2). This is equivalent to performing BDP (with the unweighted score) on the input trajectories. We perform this step to allow DPTS to revert to the prediction made by BDP should its expected performance be better. DPTS then generates a set of clusters, $\mathcal{C}_{DPTS}$, for all trajectories in $\mathcal{T}$ using the parameter matrix, M_{DPTS}, (see Algorithm 1). The GMMs in Φ_{DPTS} are trained with Algorithm 2, using the feature vector input to the algorithm, such as $\langle lat, long, day, hour \rangle$. Once the GMMs have been trained, the training stage of DPTS is complete, which returns two sets of GMMs, namely Φ_{BDP} trained using the parameters in M_{BDP}, and Φ_{DPTS} using the parameters in M_{DPTS}.

3.3 Trajectory Prediction

Algorithm 4 defines the process of predicting the cluster in which an unfolding trajectory belongs. The log-likelihood for each GMM in Φ is calculated for each instance, x_i, within the trajectory, t, and is used to score the GMMs. This algorithm can be run with $\Phi = \Phi_{BDP}$ and $\Phi = \Phi_{DPTS}$, to obtain the respective predictions. The log-likelihood is then translated into a probability using the softmax function. The prediction algorithm iterates through each instance, x_i, in the trajectory, maintaining a sum of the likelihood and probability over all instances. DPTS predicts the cluster for the final instance in the trajectory, where the probability is averaged. As the algorithm iterates through each GMM, $\phi_j \in \Phi$, the total likelihood is compared to the best seen so far, updating the predicted cluster and its respective probability if it exceeds the previous best. The method returns the predicted cluster and its probability.

In DPTS, we introduce the notion of a decision threshold, which is the value to be exceeded by the probability of the DPTS prediction in order to use the DPTS prediction. Failing to exceed the decision threshold will cause DPTS to revert to the prediction made by BDP. In DPTS, we consider two modes of decision threshold, namely a static and dynamic mode, controlled by a boolean flag, $\theta_{dynamic}$. The static mode, $\theta_{dynamic} = False$, is where the

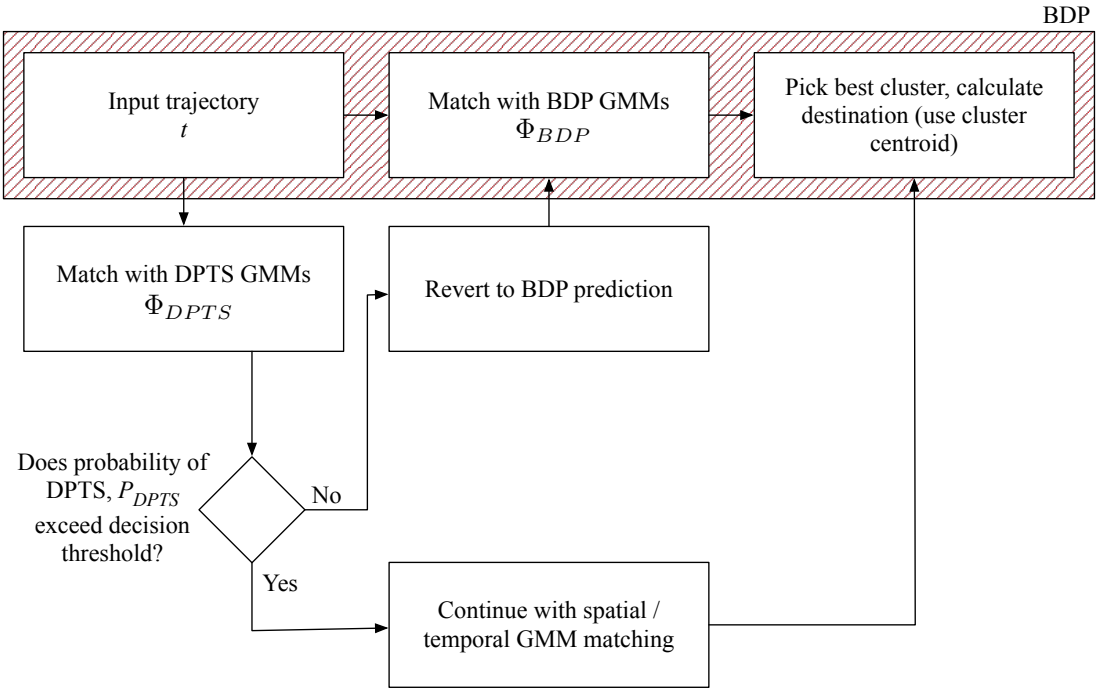


Figure 2: Overview of DPTS methodology for using GMMs on trajectory sub-clusterings for destination prediction.

prediction probability of DPTS, P_{DPTS}, is compared to a fixed predefined decision threshold, θ. In the dynamic mode, $\theta_{dynamic} = True$, the DPTS prediction probability, P_{DPTS}, is compared to the prediction probability of BDP, P_{BPD}, multiplied by the decision threshold parameter value, θ. The decision threshold parameter value, θ, is therefore used to scale P_{BPD} to increase or decrease the likelihood of the exceeding the decision threshold. Such scaling is needed since the BDP prediction probability, P_{BPD}, may be consistently higher than that of DPTS, P_{DPTS}, since the BDP clusters are less specific. Algorithm 5 defines the method to check whether the decision threshold is exceeded or not. The algorithm returns true if the DPTS prediction has exceeded the decision threshold, and therefore will be used for prediction.

The deployment stage of DPTS is illustrated in Figure 2 and detailed in Algorithm 6. This method takes five parameters: (i) the unfolding trajectory to predict, t, (ii) a set of trained GMMs using BDP, Φ_{BDP}, (iii) a set of trained GMMs using DPTS, Φ_{DPTS}, (iv) a boolean flag that indicates whether to use the dynamic or static mode for the decision threshold, $\theta_{dynamic}$, and (v) the value to use within the decision threshold calculation, θ. The algorithm begins with a given an input trajectory, t, for which cluster predictions and their corresponding probabilities, for both BDP and DPTS, are computed. Based on these probabilities, the decision threshold, θ, is evaluated, and if it is exceeded then the DPTS prediction is used, otherwise the algorithm reverts to using the prediction made by BDP. The predicted destination itself is obtained by taking the cluster centroid of the predicted cluster.

4 Data & Experimental Methodology

In this paper, we use three separate datasets to evaluate DPTS, two of which are those used by Besse *et al.* to evaluate BDP [8], on which DPTS is based. The first of these, the Caltrain dataset, contains 4,127 taxi trajectories originating from Caltrain Station, San Francisco [52]. The second, the Porto dataset, contains 19,423 taxi trajectories commencing from Sao Bento station, located in the centre of Porto [53]. The third dataset, named POL, is a pattern of life dataset, collected over a number of non-consecutive weeks for a single participant. Unlike the Cal-

Algorithm 4: $predict(t, \Phi)$, prediction stage of DPTS

inputs : t, a unfolding trajectory to predict
Φ, a set containing the trained GMMs for each cluster in $\mathcal{C}$

output: $predictedCluster, bestProb$, the predicted cluster for the last instance in t, along with the probability for the prediction

```
predictedCluster = null
bestLikelihood, bestProb = −∞, −∞
for j ∈ [1, |C|] do
    totalLikelihood = 0
    totalProbability = 0
    for x_i ∈ t do
        // calculate likelihood for instance x_i in GMM φ_j
        likelihood = likelihood(x_i, φ_j)
        // convert likelihood into probability using softmax function
        prob = softmax(likelihood)
        // increment total likelihood and probability
        totalLikelihood = totalLikelihood + likelihood
        totalProbability = totalProbability + likelihood
    end
    // calculate average probability
    averageProbability = totalProbability/|t|
    // only predict for the latest instance in the unfolding trajectory
    if totalLikelihood > bestLikelihood then
        predictedCluster = j
        bestLikelihood = totalLikelihood
        bestProb = averageProbability
    end
end
return predictedCluster, bestProb
```

train and Porto datasets, the POL dataset does not have a single starting location for all trajectories, and so allows us to evaluate the performance of DPTS when trajectories do not have a common starting location.

For all stages of the evaluation, unless explicitly stated, we explore in detail the effect of the parameters on the Caltrain dataset, and state the best results for the Porto dataset. Due to the different nature of the POL dataset, we evaluate DPTS on the POL dataset separately in Section 5.5. Unless explicitly stated, our comparison against the baseline BDP method uses the unweighted score, rather than relying on auxiliary variables and weighting functions to modify the score since, as noted in Section 2, our focus is on identifying suitable clusters from which to make predictions. We comment on the effectiveness of our method on these datasets, noting the differences. In this paper, we use a value of 10000 for μ and 20 for κ, since these parameters are not the focus of our investigation and these values were used in the original evaluation ofBDP, allowing for a direct comparison [8].

The first stage of our evaluation of DPTS investigates the order of clustering and the parameters for the day-of-week and hour-of-day clustering, to find the best performing values for each. We perform all combinations of clustering with SSPD, day-of-week and hour-of-day using two iterations. Within this parameter search, we use a decision threshold of 0 in the static mode, meaning that the DPTS prediction will always be used. Table 3 shows the set of parameters used in this evaluation. For the SSPD clustering, we use the parameter values from the work of Besse *et al.*, which are 25 and 45 for the Caltrain and Porto datasets respectively. We train the GMMs

Algorithm 5: $exceedThreshold(P_{DPTS}, P_{BDP}, \theta_{dynamic}, \theta)$, decision threshold check in DPTS

inputs : P_{DPTS}, average probability of the best GMM for the DPTS clusters, $\mathcal{C}_{DPTS}$
P_{BDP}, average probability of the best GMM for the BDP clusters, $\mathcal{C}_{BDP}$
$\theta_{dynamic}$, a boolean flag indicating whether to use dynamic or static mode
θ, the parameter value to use in the decision threshold

output: $thresholdExceeded$, whether the decision threshold to use Φ_{DPTS} was exceeded

```
if θ_dynamic then
    return P_DPTS > (θ * P_BDP)
else
    return P_DPTS > θ
end
```

Algorithm 6: Deployment stage of DPTS

inputs : t, a unfolding trajectory to predict
Φ_{BDP}, a set containing the trained GMMs for each cluster in $\mathcal{C}_{BDP}$
Φ_{DPTS}, a set containing the trained GMMs for each cluster in $\mathcal{C}_{DPTS}$
$\theta_{dynamic}$, a boolean flag indicating whether to use dynamic or static mode
θ, the parameter value to use in the decision threshold

output: $predictedDestination$, the predicted destination for trajectory, t

```
prediction_BDP, P_BDP = predict(t, Φ_BDP)
prediction_DPTS, P_DPTS = predict(t, Φ_DPTS)
if exceedThreshold(P_BDP, P_DPTS, θ_dynamic, θ) then
    predictedDestination = clusterCentroid(prediction_DPTS)
else
    predictedDestination = clusterCentroid(prediction_BDP)
end
return predictedDestination
```

with 4 different feature vectors, namely $\langle lat, long\rangle$, $\langle lat, long, day\rangle$, $\langle lat, long, hour\rangle$, and $\langle lat, long, day, hour\rangle$, resulting in 392 sets of results for each dataset. Evaluating the mean distance error for each parameter combination against the baseline performance of BDP, we discard those that are significantly outperformed by the baseline from further evaluation.

After the parameter search has been completed, the next stage evaluates the effect of our proposed decision thresholds on performance. We analyse the decision threshold in both static and dynamic modes, and compare these results to both the baseline performance of BDP and the performance of DPTS where the decision threshold is set to 0. For the static and dynamic modes of the decision threshold, we explore the parameter value, θ in the range [0,1], in increments of 0.05 and 0.1 respectively.

The third stage of our evaluation explores the impact of the clustering parameter for SSPD. In our initial analysis, we use the best performing parameter for each dataset, as reported by Besse *et al.* [8], and so we also investigate a range of values for the SSPD clustering parameter, in increments of 5. Our stopping criteria is where the supplied parameter value causes an error due to the number of clusters being too large, and therefore not giving sufficient data to properly train the GMMs.

In the next stage of our evaluation, we add a third iteration of clustering to DPTS, considering SSPD, day-of-week and hour-of-day simultaneously. The ordering of clustering iterations is evaluated, and the performance of three iterations is compared to that of using two iterations, using the mean distance error.

Table 3: Set of parameters, α, used for our initial evaluation.

Experiment #	Order of Clustering	Parameters (α) Considered
1–4	SSPD → Day-of-week	25 → [0,3]
5–13	SSPD → Hour-of-day	25 → [0,8]
14–17	Day-of-week → SSPD	[0,3] → 25
18–53	Day-of-week → Hour-of-day	[0,3] → [0,8]
54–62	Hour-of-day → SSPD	[0,8] → 25
63–98	Hour-of-day → Day-of-week	[0,8] → [0,3]

For the final stage of our evaluation we consider destination clustering, specifically on the POL dataset. The evaluation of the POL dataset is notable since, unlike the previous datasets, the POL dataset does not contain a single starting location for all journeys. To explore this aspect, we propose adding a fourth clustering approach which groups trajectories based on the trajectory destinations, using the Haversine distance between each destination to generate the dissimilarity matrix, D.

5 Results

In this section, we discuss the results of applying DPTS to the Caltrain [52], Porto [53] and POL datasets. We evaluate the effect of the clustering parameters, and analyse the impact of introducing a decision threshold using the evaluation approach outlined in the previous section. Unless stated, the results presented in this section are based on the Caltrain dataset [52]. Due to its distinct nature, the POL dataset is evaluated separately in Section 5.5. Note that for simplicity figures that have trajectory completion on the x-axis have an origin of 0%, however the data points start from the first instance of the trajectory.

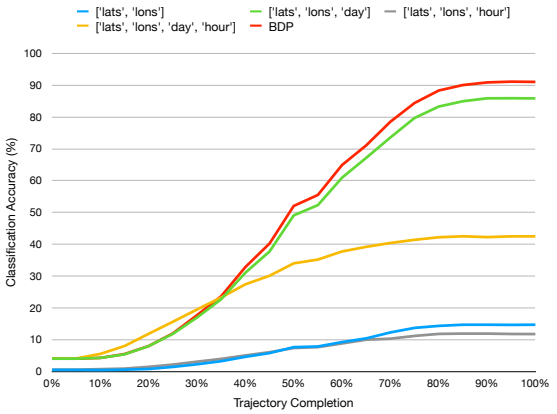
5.1 Clustering Parameter Search

This section evaluates our novel iterative clustering approach, and the impact of altering the parameters within the parameter matrix, M_{DPTS}. In this analysis, we discuss in detail the effect of altering the parameters on the Caltrain dataset, and simply report the best performing parameters on the Porto dataset.

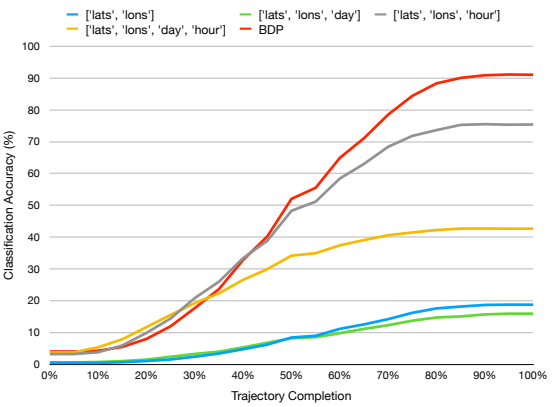
We first give an overview the classification performance for each of the 6 parameter combinations outlined in Table 3. Note that there is a strong correlation between the features used in the GMM and the clustering criteria. For example, if the hour-of-day is used to cluster the trajectories but is not present in the feature vector provided to the GMM, then the performance is be severely degraded. The exception to this is that the $\langle lat, long \rangle$ features are always needed in the feature vector to achieve a reasonable performance, even if SSPD was not included in the clustering stage. The classification performance for the top performing parameters for each combination are shown in Figure 3, in addition to the baseline performance.

Clustering with SSPD followed by the day-of-week achieves a peak performance of 85.90% at 95% trajectory completion. This was obtained by setting the clustering parameter for the day-of-week to $\alpha = 2$, and $\langle lat, long, day \rangle$ as our feature vector. If the day-of-week is omitted from the feature vector, then the performance falls to a maximum of 14.73%. Interestingly, if the hour-of-day is also included, i.e., $\langle lat, long, day, hour \rangle$, the performance sees a notable drop, with a maximum of 42.52% at 85% trajectory completion. These results are shown in Figure 3a.

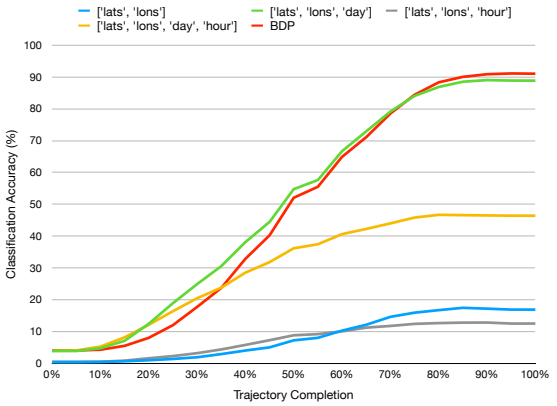
Conversely, if we cluster using the day-of-week followed by SSPD (see Figure 3c), then the clustering parameter, α, does not make any difference to the performance. Slightly decreased performance is observed in the first 10% of trajectory completion, but after this the performance exceeds that of having SSPD followed by the day-of-week. The peak performance is 89.02%, achieved at 90% trajectory completion. Similar to SSPD followed by the



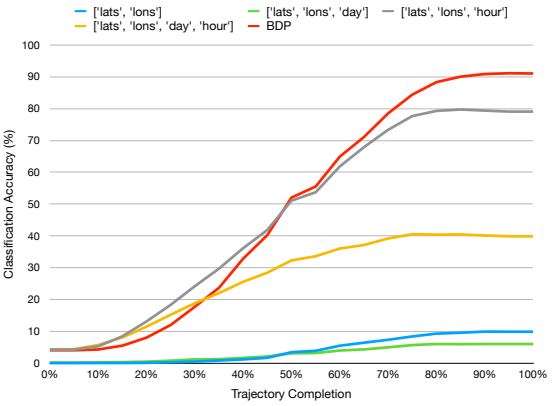
(a) SSPD → Day-of-week



(b) SSPD → Hour-of-day



(c) Day-of-week → SSPD



(d) Hour-of-day → SSPD

Figure 3: Classification performance for DPTS on the Caltrain dataset.

day-of-week, omitting day-of-week from the feature vector causes a noticeable drop in performance, as does the addition of the hour-of-day.

When clustering by the hour-of-day followed by SSPD, we observe a peak performance of 79.21% at 85% trajectory completion, as illustrated in Figure 3d. There is a noticeable degradation in performance when not using the hour-of-day in the feature vector, as seen in the previous results. If we reverse the order of clustering to have SSPD followed by hour-of-day, a peak performance of 75.58% is achieved at 90% trajectory completion (see Figure 3b). From these results, we can see that higher performance is achieved when the temporal component (day-of-week or hour-of-day) is clustered prior to the spatial component, SSPD.

If we consider both temporal components, the day-of-week and the hour-of-day, without SSPD, the classification performance is misleading. The day-of-week and the hour-of-day are taken from the start of the trajectory, and therefore their respective values are constant throughout. These combinations are unsuitable due to the little information they provide.

We take the best performing parameters from each of the 6 clustering combinations, using the classification percentage at 100% trajectory completion. The parameters, and the feature vector used for each of the top combinations is shown in Table 4. The temporal-only combinations are included for reference, but show a misleading classification accuracy as noted above. Figure 4 illustrates each of the top combinations from Table 4 against the baseline performance, BDP. Most of the performance gain can be seen in the initial 40% of the unfolding tra-

Table 4: Performance comparison between each two-iteration clustering combination.

Clustering Order	**Accuracy (%)**	**Avg. Cluster Dist. (m)**
SSPD, $\alpha = 25 \rightarrow$ Day, $\alpha = 2$	85.85	544
Day, $\alpha = 0 \rightarrow$ SSPD, $\alpha = 25$	88.85	511
Hour, $\alpha = 8 \rightarrow$ SSPD, $\alpha = 25$	79.06	405
SSPD, $\alpha = 25 \rightarrow$ Hour, $\alpha = 1$	75.41	448
Day, $\alpha = 0 \rightarrow$ Hour, $\alpha = 6$	99.18	1439*
Hour, $\alpha = 8 \rightarrow$ Day, $\alpha = 2$	99.03	1424*

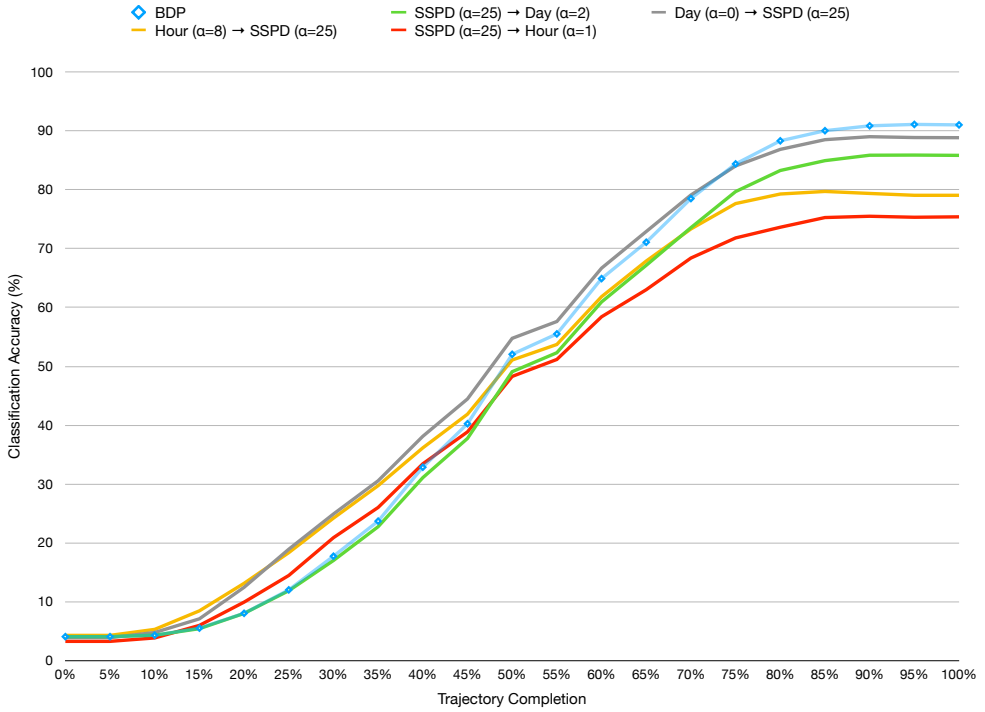


Figure 4: Comparison of the BDP performance against each of the top DPTS combinations on the Caltrain dataset.

jectories, after which BDP starts to outperform the DPTS combinations. Due to the misleading performance, the temporal-only combinations are omitted from Figure 4.

If we consider the predicted clusters and calculate the distance error from the prediction to the ground truth, we obtain the results shown in Figure 5. The first point to note is the two straight lines, which show the prediction error of both temporal-only combinations. This is expected, since the temporal values provided to the GMM do not change as the trajectory progresses, but it may not be immediately apparent as to why such high classification performance translates to a large prediction error. If we refer back to Table 4, we note the large average cluster distances for the temporal combinations. This explains the high distance error, because even though the classification performance is good, the clusters are noticeably larger, and therefore the centroid that is used for prediction is on average further from the actual destination. When clustering with SSPD and then the day-of-week, we see no improvement over the baseline. The other combinations, day-of-week to SSPD, hour-of-day to SSPD and SSPD to hour-of-day, all show reductions in distance error over the baseline from 20% to 60% of trajectory completion. After 70% of trajectory competition, the baseline performance is unbeaten. Given that we saw no improvement when clustering from SSPD to day-of-week, and that the temporal combinations have such large cluster distances, we omit these combinations from further evaluation.

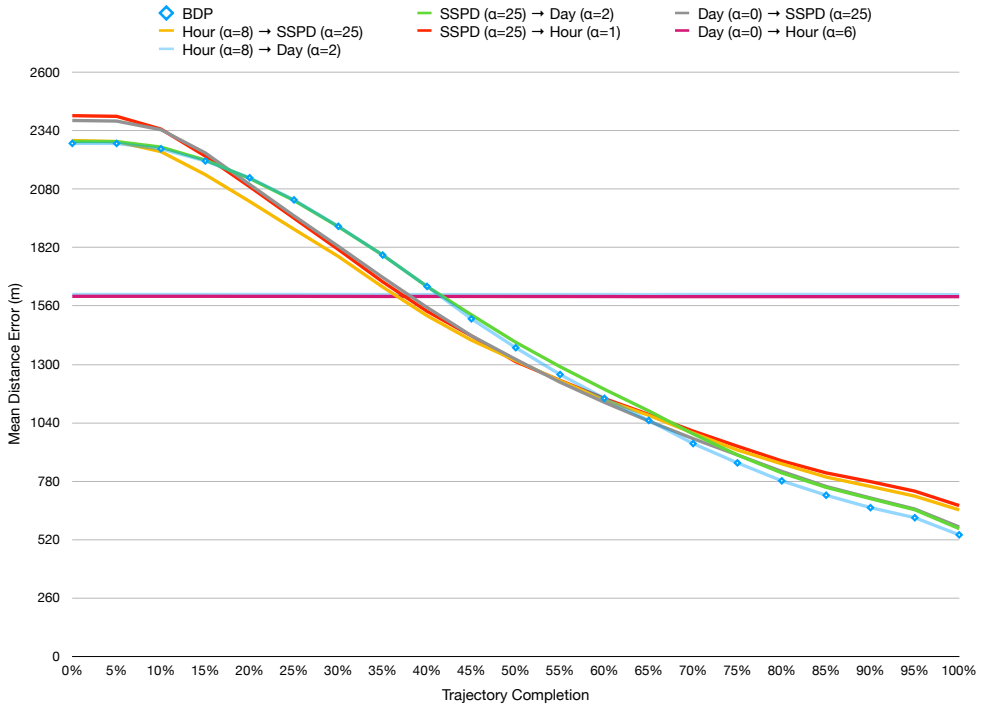


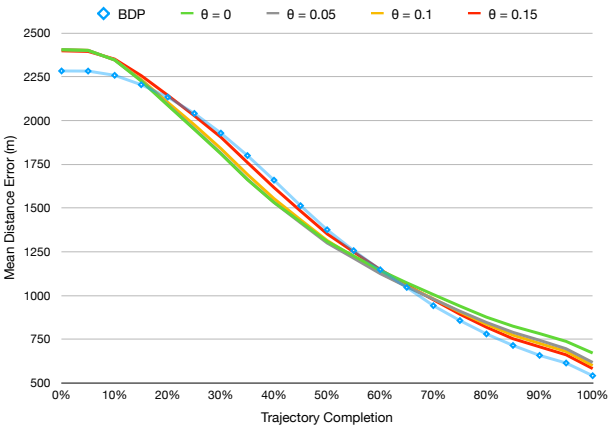
Figure 5: Comparison of the prediction error from BDP against each of the top parameter combinations for DPTS on the Caltrain dataset.

When applying DPTS to the Porto dataset, the hour-of-day ($\alpha = 0$) $\rightarrow$ SSPD ($\alpha = 45$) combination, gives the highest performance. Even though there is a slight improvement in the middle of the trajectories, the overall performance is lower than that of BDP, due to degraded performance at the start and end of the trajectories. This follows the trend seen with the Caltrain dataset. Overall, DPTS is outperformed by BDP on the Porto dataset, by an average of 7 metres.

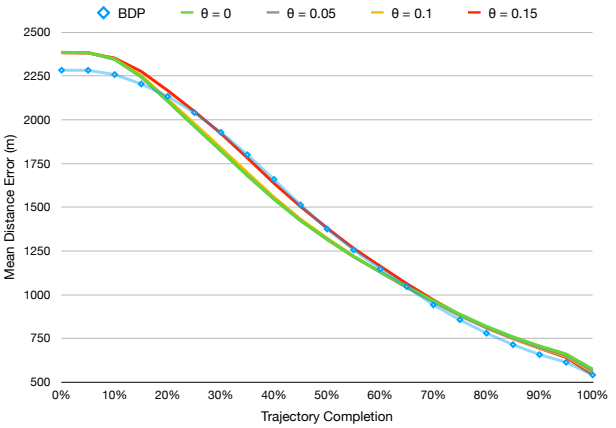
5.2 Evaluation of the Decision Threshold

Considering the results discussed in Section 5.1, we see that the baseline performance exceeds that of DPTS in the final portion of the journey. To address this, we propose using a decision threshold, that combines our novel method, DPTS, and the existing method, BDP, within a single wrapper. The decision threshold selects a prediction to use at different stages of the unfolding trajectories, according to the prediction probability of DPTS, P_{DPTS}, and BDP, P_{BDP}. As described in Section 3, we consider two modes for the decision threshold, namely a static mode ($\theta_{dynamic} = False$) and a dynamic mode ($\theta_{dynamic} = True$).

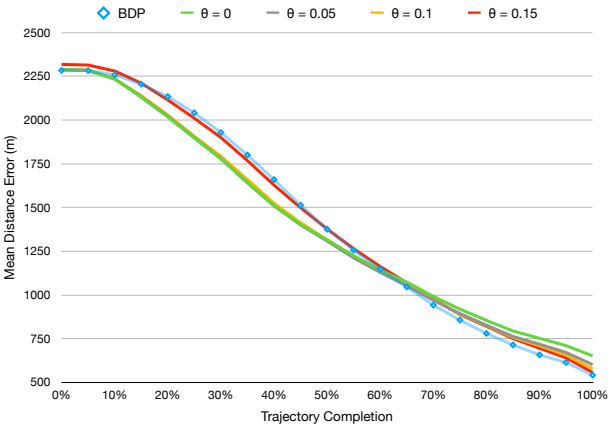
First we evaluate the effect of a decision threshold in the static mode, by considering values in the range [0,1] with increments of 0.05. The effect of the decision threshold, θ, on SSPD $\rightarrow$ Hour-of-day is shown in Figure 6a. A decision threshold of 0 in the static mode is essentially removing consideration of BDP, since all probabilities greater than 0 will pass, and therefore the result will be identical to our original results. Conversely, a decision threshold of 1 will always revert to the baseline results of BDP. We can see that setting a threshold of 0.05 improves the performance past 50% trajectory completion, with no apparent loss of performance below 50% completion. If we increase the decision threshold to 0.1, we notice a loss of performance (compared to a decision threshold of 0) from 15–50% of trajectory completion, after which the performance improves. Further increasing the decision threshold to 0.15 leads to a more significant degradation in performance from 10–65% of trajectory completion, after which a small improvement is made for the remainder of the journey. At this decision threshold, we also see



(a) SSPD → Hour-of-day



(b) Day-of-week → SSPD



(c) Hour-of-day → SSPD

Figure 6: Comparison of destination prediction performance for given decision threshold values in static mode.

a slight improvement in performance in the first 5% of trajectory completion compared to our original results. Any further increase in the decision threshold has the effect of improving the first part of the journey (0–15% trajectory completion), degrading the middle of the journey (15–65% trajectory completion) and improving the final part of the journey (65–100% trajectory completion). Overall, in static mode, a decision threshold of 0.05 gives the best trade-off, resulting in the highest average performance for SSPD $\rightarrow$ hour-of-day.

Figure 6b illustrates the effect of the decision threshold in static mode on day-of-week $\rightarrow$ SSPD. We observe a similar trend to SSPD $\rightarrow$ hour-of-day, but note that the original result (with a decision threshold of 0) performs nearer to the baseline result in the final stage of the trajectories (65–100% trajectory completion). Therefore, it seems that adding a decision threshold will have a smaller positive impact on this combination. Decision thresholds of 0.05 and 0.1 provide a good trade-off between performance in the middle and final parts of the journey. We note that a decision threshold of 0.15 gives a greater loss of performance in the middle of the journey, similar to that reported in the analysis of SSPD $\rightarrow$ hour-of-day. In the static mode, a decision threshold of 0.05 also gives the best trade-off for day-of-week $\rightarrow$ SSPD performance.

The combination of hour-of-day $\rightarrow$ SSPD, as shown in Figure 6c, appears to give the best results of the three alternatives Most notably, the early part of the journey (0–10% trajectory completion), is nearer the baseline performance than the other two combinations. As with the other results, we see that a decision threshold of 0.05 gives the optimum performance trade-off, with more apparent losses seen for decision threshold values of 0.15 and above. All three sets of results appear to provide the best overall performance when a decision threshold of 0.05 is used, with a more significant loss of performance with a decision threshold of 0.15.

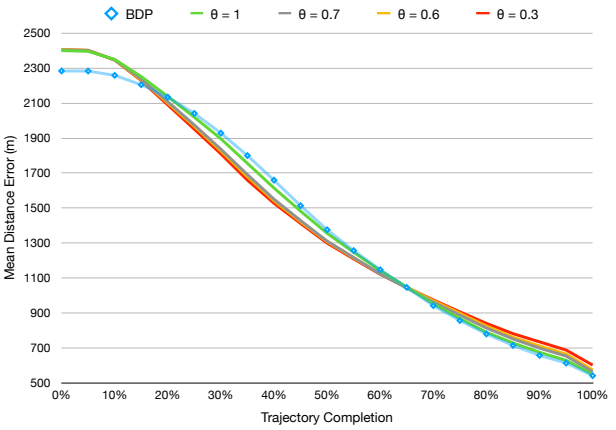
We will now consider the decision threshold in dynamic mode ($\theta_{dynamic} = True$), to investigate whether this outperforms the static mode ($\theta_{dynamic} = False$). Figure 7 shows the performance when a decision threshold is used in dynamic mode. The decision threshold in dynamic mode ($\theta_{dynamic} = True$), as explained in Section 3, is where the probability of the DPTS prediction, P_{DPTS}, is compared directly to the probability of the baseline BDP prediction, P_{BDP}. The decision threshold parameter value, θ, is used to scale the probability of the BDP prediction, P_{BPD}. For our evaluation we explored parameter values in the range [0,1] in increments of 0.1. Overall, we found that a decision threshold value of 0.7 for SSPD $\rightarrow$ hour and 0.4 for day $\rightarrow$ SSPD and hour $\rightarrow$ SSPD gave the best prediction performance. On average, using the decision threshold in dynamic mode causes a slight improvement in performance compared to the static mode. This gain, however, is minimal in terms of metres, and appears to be of little effect, but could be influenced by properties of the input dataset.

When evaluating the decision threshold on the Porto dataset, a slight improvement over the performance of BDP is seen. A decision threshold in the dynamic mode with a parameter value of $\theta = 0.9$, was used on the hour-of-day ($\alpha = 0$) $\rightarrow$ SSPD ($\alpha = 45$) combination, giving an average distance error of 10 metres lower than BDP. Figure 8 illustrates the performance comparison between BDP, DPTS ($\theta = 0$) and DPTS ($\theta = 0.9$).

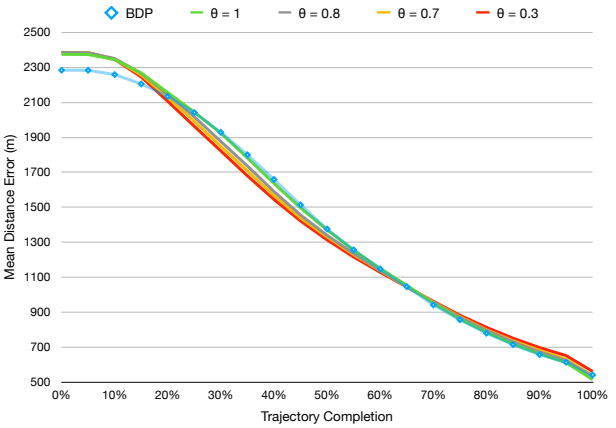
5.3 Altering the SSPD parameter values

We investigated changing the clustering parameter, α, on the highest performing combinations. In the results discussed above, this was fixed at the values used by Besse *et al.* in their investigation [8]. Intuitively, lowering the parameter value in BDP, should increase the trajectory classification but also increase the destination prediction error, since the destinations in these larger clusters will be more spread out. However, since DPTS performs iterative clustering, there may be benefits to lowering the α value for SSPD.

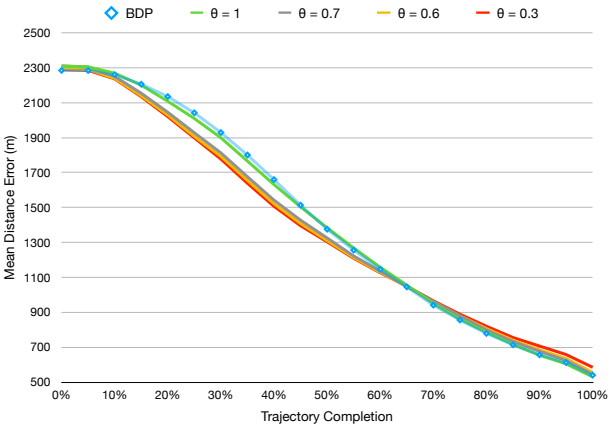
Figure 9 illustrates the comparison between BDP, DPTS (hour-of-day, $\alpha = 8 \rightarrow$ SSPD, $\alpha = 25$), DPTS (SSPD, $\alpha = 10 \rightarrow$ hour-of-day, $\alpha = 6$) with $\theta = 0.0$, and DPTS (SSPD, $\alpha = 10 \rightarrow$ hour-of-day, $\alpha = 6$) with $\theta = 1.0$. It is immediately apparent that reducing α provides a significant reduction in destination error in the first portion of the journey. After 60% of the trajectory is complete the performance degrades below the performance of BDP. When introducing a decision threshold greater than 0, the performance gains are comparable at the start of the journey, and the performance degradation is slightly reduced past 65% trajectory completion. Overall, the variant



(a) SSPD → Hour-of-day



(b) Day-of-week → SSPD



(c) Hour-of-day → SSPD

Figure 7: Comparison of destination prediction performance for given decision threshold values in dynamic mode.

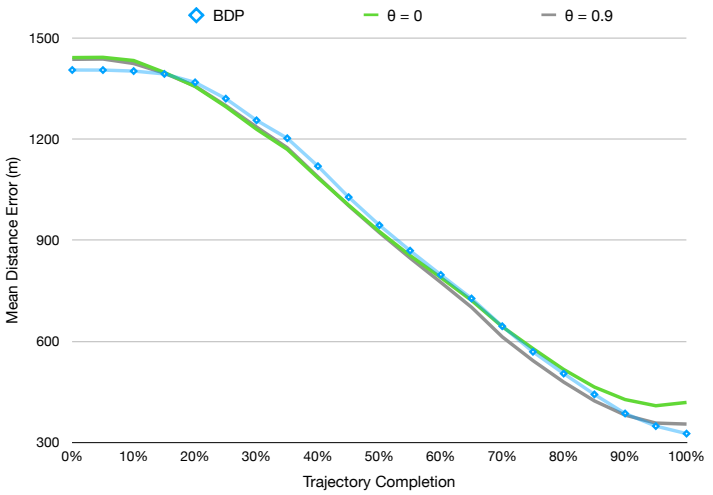


Figure 8: Comparison of the prediction error for the Porto dataset.

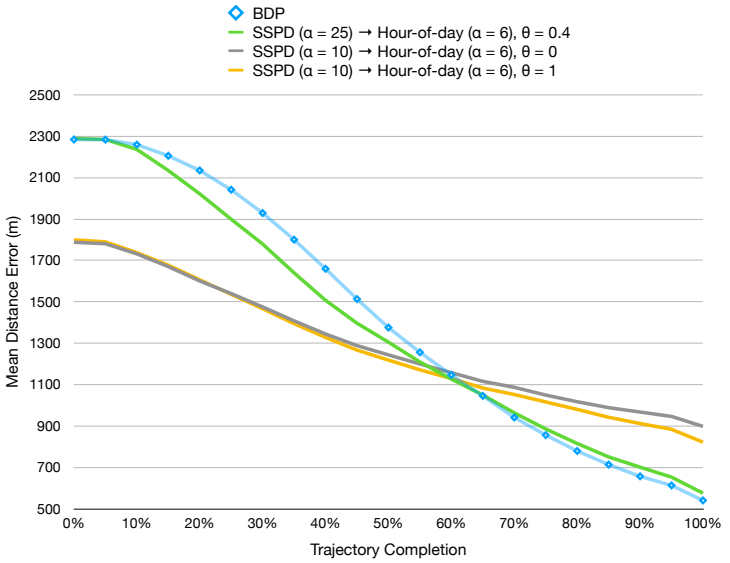


Figure 9: Comparison of the prediction error for the Caltrain dataset, when altering the clustering parameter value, α, for SSPD.

with $\alpha = 10$ and a decision threshold of $\theta = 1.0$ provides the best performance on average over the entire duration of the journey, with significant gains in the first 30–40% of the trajectory.

If we compare DPTS (SSPD, $\alpha = 10 \rightarrow$ hour-of-day, $\alpha = 6$) with $\theta = 1.0$ with the weighted version of BDP, we observe similar performance at the start of the journey. As the trajectory unfolds, there is a larger performance gap between DPTS and the weighted version of BDP, with BDP seeing a maximum of 366 metres lower prediction error at some points.

When applied to the Porto dataset, no gains in performance were observed, and the original clustering parameter for SSPD, $\alpha = 45$, produced the highest performance.

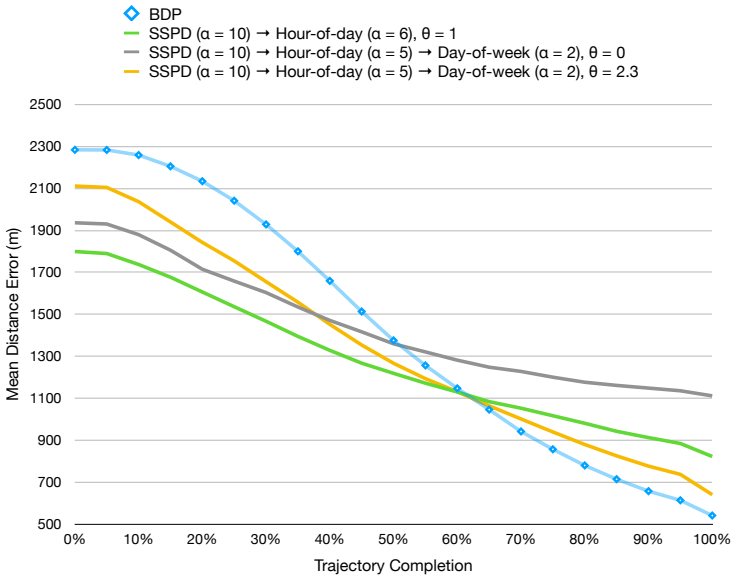


Figure 10: Comparison of the prediction error for the Caltrain dataset between two and three iterations of clustering in DPTS and BDP

5.4 Adding a third clustering iteration

We now evaluate the performance of DPTS using three iterations of clustering, and compare the performance with using two iterations. The motivation behind including an additional iteration is that we can further decompose the clusters (whilst trying to maintain a high accuracy for the trajectory classification). The drawback of adding a third iteration is that it can generate a large number of clusters, each containing only a few trajectories. If the number of clusters increases too much, there could be a situation in which some clusters only contain a single trajectory, and therefore the cluster has no training data and is not useful for prediction.

When using three iterations of clustering, we find the best combination to be SSPD $\rightarrow$ hour-of-day $\rightarrow$ day-of-week. However, Figure 10 shows that the performance of this combination is not as high as to that of two iterations with a reduced α for SSPD (as discussed above). When we add a decision threshold in dynamic mode, the performance is degraded in the initial 40% of the trajectories, but sees improved performance from 60% competition onwards, nearer to that of the BDP. Taking into consideration the average distance error throughout the trajectory, the extra computation required for the additional layer, and the increased number of GMMs required, we take the previous combination with two clustering iterations to be the better variant.

5.5 Evaluating DPTS on the POL dataset

Applying DPTS to the POL dataset provides an insight into a more general application of the algorithm, since unlike the other datasets the POL dataset contains trajectories with multiple starting locations. When applying BDP to the POL dataset, we notice a increase in distance error at around 50–80% trajectory completion. This is due to the characteristic that, unlike the taxi datasets, we do not have a single starting location, and therefore we can not assume a fixed direction of travel away from the source. To address this issue, we add another iteration of clustering, in which we generate a dissimilarity matrix of trajectories based on the destination location to be used as input to the hierarchical agglomerative clustering. For our evaluation we use 2500 metres as the clustering parameter, α, for this iteration of clustering. Further exploration of this value is outside the scope of this paper, and could be investigated in future work.

Figure 11 shows the results of applying DPTS to the POL dataset, with a comparison to the performance of BDP. When we apply DPTS, using four iterations of clustering (hour-of-day, $\alpha = 0 \rightarrow$ day-of-week, $\alpha = 0$

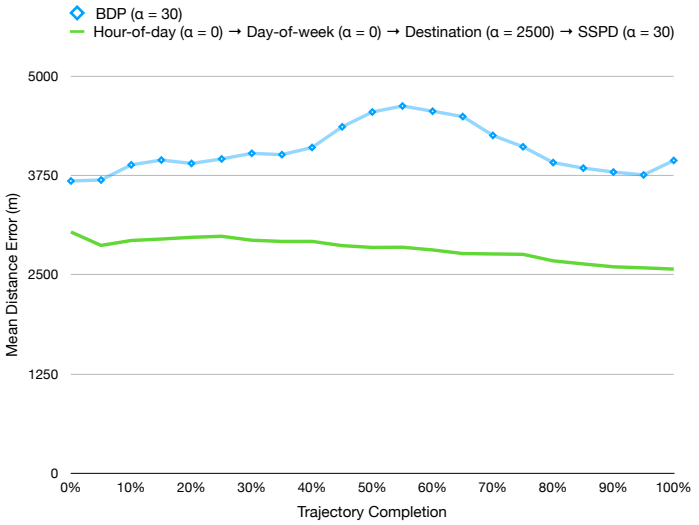


Figure 11: Comparison of the prediction error between BDP and DPTS when applied to the POL dataset.

$\rightarrow$ destination, $\alpha = 2500 \rightarrow$ SSPD, $\alpha = 30$), we see a significant improvement over BDP. This combination outperforms BDP over the entire trajectory, with an average reduction in error of over 1.2km. A small decrease in error can be seen as the trajectory unfolds, unlike the sudden rise in error as seen with BDP.

6 Conclusion

In this paper, we propose DPTS, an extension to the existing BDP method. DPTS uses an iterative clustering stage and a decision threshold to improve the destination prediction performance on vehicle trajectories. DPTS harnesses the additional properties of the trajectories, attempting to further decompose them into more meaningful clusters, rather than using these temporal properties in combination with weighting functions to modify the likelihood. For our evaluation, we use the temporal properties of day-of-week and hour-of-day.

When applying DPTS to the Caltrain dataset, we see an improvement in overall performance, where our decision threshold allows the prediction to revert back to that of BDP towards the end of the trajectories, as the performance of BDP improves. If two iterations of clustering are used, with smaller parameter values for SSPD, we see a reduction in error for the first half of the journey compared to BDP, however this is at a cost of lower performance in the final 40% of trajectories. We see severely reduced effectiveness from DPTS when used on the Porto dataset, barely matching the performance of BDP. This implies that the capability of our method is somewhat dependant on the data. However, when applied to the POL dataset, which has multiple starting locations, we see promising results. BDP struggles to accurately predict destinations, with an increase in error in the middle of the trajectories. When applying DPTS with multiple clustering iterations, we see notable gains in prediction performance over BDP, that are consistent throughout the unfolding trajectories. This implies that selecting the best approach in practice is highly dependent on the application setting and the nature of the data available. In practise, we recommend adding clustering iterations for attributes that help differentiate clusters when using DPTS for applications. We also recommend the consideration of both static and dynamic thresholds, adjusting parameter values to maximise performance and balance the trade-off between performance in the early stages of a journey against that in the latter stages.

Reducing the prediction error can provide benefits to location-aware applications, such as on-route traffic updates, intelligent parking suggestions and amenity recommendations at the destination. Without an accurate location, these applications will suffer from reduced effectiveness and ultimately poor user trust. Having a more

accurate prediction earlier in the trajectory can enable these applications to provide their location-based functionality in a more timely manner.

Future work will consider the decision threshold process and investigate whether the parameter values can be removed, in order to make DPTS more generic across datasets. Additionally, further information from the time signal can be extracted to analyse the effect of seasonality and trends in user mobility to see if this can aid performance. Finally, additional investigation will be conducted into adding the difference in selected vehicle signals in further iterations of DPTS.

Acknowledgements

This work was supported by Jaguar Land Rover and the UK-EPSRC grant EP/N012380/1 as part of the jointly funded Towards Autonomy: Smart and Connected Control (TASCC) Programme.

References

[1] R Bauza and J Gozalvez. "Traffic congestion detection in large-scale scenarios using vehicle-to-vehicle communications". *Journal of Network and Computer Applications*, vol. 36, no. 5, 1295–1307, Sep 2013.

[2] Quan Yuan, Zhihan Liu, Jinglin Li, Junming Zhang and Fangchun Yang. "A traffic congestion detection and information dissemination scheme for urban expressways using vehicular networks". *Transportation Research Part C*, vol. 47, no. Part 2, 114–127, Oct 2014.

[3] Riccardo Scarinci, Benjamin Heydecker and Andreas Hegyi. "Analysis of Traffic Performance of a Merging Assistant Strategy Using Cooperative Vehicles". *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, 2094–2103, Feb 2015.

[4] Vadim A Butakov and Petros Ioannou. "Personalized Driver Assistance for Signalized Intersections Using V2I Communication". *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, 1910–1919, Jan 2016.

[5] Behrang Asadi and Ardalan Vahidi. "Predictive Cruise Control: Utilizing Upcoming Traffic Signal Information for Improving Fuel Economy and Reducing Trip Time". *IEEE Transactions on Control Systems Technology*, vol. 19, no. 3, 707–714, May 2011.

[6] Shen Wang, Soufiene Djahel, Zonghua Zhang and Jennifer McManis. "Next Road Rerouting: A Multiagent System for Mitigating Unexpected Urban Traffic Congestion". *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 10, 2888–2899, mar 2016.

[7] A. M. de Souza, R. S. Yokoyama, G. Maia, A. Loureiro and L. Villas. "Real-time path planning to prevent traffic jam through an intelligent transportation system". In *Proc. of the IEEE Symposium on Computers and Communication*, pages 726–731, 2016.

[8] P. C. Besse, B. Guillouet, J. Loubes and F. Royer. "Destination prediction by trajectory distribution-based model". *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, 2470–2481, 2018.

[9] P. C. Besse, B. Guillouet, J. Loubes and F. Royer. "Review and perspective for distance-based clustering of vehicle trajectories". *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, 3306–3317, 2016.

[10] J Krumm and E Horvitz. "Predestination: Where do you want to go today?". *Computer*, vol. 40, no. 4, 105–107, 2007.

[11] Brian D. Ziebart, Andrew L. Maas, Anind K. Dey and J. Andrew Bagnell. "Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior". In *Proc. of the International Conference on Ubiquitous Computing*, pages 322—331, 2008.

[12] R Simmons, B Browning and Y Zhang. "Learning to predict driver route and destination intent". In *Proc. of the IEEE Conference on Intelligent Transportation Systems*, pages 127–132, 2006.

[13] Daniel Ashbrook and Thad Starner. "Using gps to learn significant locations and predict movement across multiple users". *Personal and Ubiquitous Computing*, vol. 7, no. 5, 275—-286, oct 2003.

[14] Donald J. Patterson, Lin Liao, Dieter Fox and Henry Kautz. "Inferring high-level behavior from low-level sensors". In *Proc. of the International Conference on Ubiquitous Computing*, pages 73–89, 2003.

[15] Dirk Brockmann, Lars Hufnagel and Theo Geisel. "The scaling laws of human travel". *Nature*, vol. 439, 462–465, 2006.

[16] Chaoming Song, Tal Koren, Pu Wang and Albert-László Barabási. "Modelling the scaling properties of human mobility". *Nature Physics*, vol. 6, 818–823, 2010.

[17] Marta C Gonzalez, Cesar A Hidalgo and Albert-Laszlo Barabasi. "Understanding individual human mobility patterns". *Nature*, vol. 453, 779–782, 2008.

[18] Christian M Schneider, Vitaly Belik, Thomas Couronné, Zbigniew Smoreda and Marta C González. "Unravelling daily human mobility motifs". *Journal of The Royal Society Interface*, vol. 10, 1–8, 2013.

[19] Shan Jiang, Joseph Ferreira and Marta C Gonzalez. "Activity-based human mobility patterns inferred from mobile phone data: A case study of singapore". *IEEE Transactions on Big Data*, vol. 3, 208–219, 2017.

[20] Joe Castiglione, Mark Bradley and John Gliebe. *Activity-based travel demand models: A primer*. Number SHRP 2 Report S2-C46-RR-1. The National Academies Press, 2015.

[21] Sebastian Büscher, Manuel Batram and Dietmar Bauer. "Using motifs for population synthesis in multi-agent mobility simulation models". In *Workshop on Stochastic Models, Statistics and their Application*, pages 335–349, 2019.

[22] Shan Jiang, Gaston A. Fiore, Yingxiang Yang et al. "A review of urban computing for mobile phone traces: Current methods, challenges and opportunities". In *Proc. of the ACM SIGKDD International Workshop on Urban Computing*, pages 1–9, 2013.

[23] Zhenhuan Li, Haiyang Wang, Xiaming Chen et al. "Discovering mass activities using anomalies in individual mobility motifs". In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 321–326, 2016.

[24] Yi-lin Sun, Ari Tarigan, Owen Waygood and Dian-hai Wang. "Diversity in diversification: an analysis of shopping trips in six-week travel diary data". *Journal of Zhejiang University-SCIENCE A*, vol. 18, 234–244, 2017.

[25] Robert Schlich and Kay W Axhausen. "Habitual travel behaviour: evidence from a six-week travel diary". *Transportation*, vol. 30, 13–36, 2003.

[26] Andreas Allström, Ida Kristoffersson and Yusak Susilo. "Smartphone based travel diary collection: Experiences from a field trial in stockholm". *Transportation research procedia*, vol. 26, 32–38, 2017.

[27] Nursitihazlin Ahmad Termida, Yusak O Susilo, Joel P Franklin and Chengxi Liu. "Understanding seasonal variation in individual's activity participation and trip generation by using four consecutive two-week travel diary". *Travel Behaviour and Society*, vol. 12, 52–63, 2018.

[28] Charles Raux, Tai-Yu Ma and Eric Cornelis. "Variability in daily activity-travel patterns: the case of a one-week travel diary". *European transport research review*, vol. 8, 26, 2016.

[29] Mohammad Javad Koohsari, Neville Owen, Rachel Cole et al. "Built environmental factors and adults' travel behaviors: role of street layout and local destinations". *Preventive medicine*, vol. 96, 124–128, 2017.

[30] Naznin Sultana Daisy, Mohammad Hesam Hafezi, Lei Liu and Hugh Millward. "Understanding and modeling the activity-travel behavior of university commuters at a large canadian university". *Journal of Urban Planning and Development*, vol. 144, 2018.

[31] Ramaswamy Hariharan and Kentaro Toyama. "Project lachesis: Parsing and modeling location histories". In *Geographic Information Science*, pages 106–124, 2004.

[32] Patrick Pakyan Choi and Martial Hebert. "Learning and Predicting Moving Object Trajectory: A Piecewise Trajectory Segment Approach". 2006.

[33] R. Assam and T. Seidl. "Bodyguards: A clairvoyant location predictor using frequent neighbors and markov model". In *Proc. of the IEEE Conference on Ubiquitous Intelligence and Computing and IEEE Conference on Autonomic and Trusted Computing*, pages 25–32, 2013.

[34] J.A. Alvarez-Garcia, J.A. Ortega, L. Gonzalez-Abril and F. Velasco. "Trip destination prediction based on past gps log using a hidden markov model". *Expert Systems with Applications*, vol. 37, no. 12, 8166–8171, dec 2010.

[35] Sung-Bae Cho. "Exploiting machine learning techniques for location recognition and prediction with smartphone logs". *Neurocomputing*, vol. 176, 98–106, feb 2016.

[36] Sébastien Gambs, Marc-Olivier Killijian and Miguel Núñez del Prado Cortez. "Next place prediction using mobility markov chains". In *Proc. of the Workshop on Measurement, Privacy, and Mobility*, pages 1–6, 2012.

[37] A. Y. Xue, R. Zhang, Y. Zheng et al. "Destination prediction by sub-trajectory synthesis and privacy protection against such prediction". In *Proc. of the IEEE International Conference on Data Engineering (ICDE)*, pages 254–265, 2013.

[38] Lin Liao, Donald J. Patterson, Dieter Fox and Henry Kautz. "Learning and inferring transportation routines". *Artificial Intelligence*, vol. 171, no. 5, 311–331, apr 2007.

[39] J. Wiest, M. Höffken, U. Kreßel and K. Dietmayer. "Probabilistic trajectory prediction with gaussian mixture models". In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 141–146, 2012.

[40] Ling Chen, Mingqi Lv, Qian Ye, Gencai Chen and John Woodward. "A personal route prediction system based on trajectory data mining". *Information Sciences*, vol. 181, no. 7, 1264–1284, apr 2011.

[41] Anna Monreale, Fabio Pinelli, Roberto Trasarti and Fosca Giannotti. "Wherenext: A location predictor on trajectory pattern mining". In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 637—646, 2009.

[42] Hassan A. Karimi and Xiong Liu. "A predictive location model for location-based services". In *Proc. of the ACM International Symposium on Advances in Geographic Information Systems*, pages 126—-133, 2003.

[43] R. Trasarti, R. Guidotti, A. Monreale and F. Giannotti. "Myway: Location prediction via mobility profiling". *Information Systems*, vol. 64, 350–367, mar 2017.

[44] Alasdair Thomason, Nathan Griffiths and Victor Sanchez. "Identifying locations from geospatial trajectories". *Journal of Computer and System Sciences*, vol. 82, no. 4, 566–581, Jun 2016.

[45] Y. Gong, Y. Li, D. Jin, L. Su and L. Zeng. "A location prediction scheme based on social correlation". In *Proc. of the IEEE Vehicular Technology Conference (VTC Spring)*, pages 1–5, 2011.

[46] Ling Chen, Mingqi Lv and Gencai Chen. "A system for destination and future route prediction based on trajectory mining". *Pervasive and Mobile Computing*, vol. 6, no. 6, 657–676, dec 2010.

[47] D. Ashbrook and T. Starner. "Learning significant locations and predicting user movement with gps". In *Proc. of the International Symposium on Wearable Computers*, pages 101–108, 2002.

[48] J. Fukano, T. Mashita, T. Hara et al. "A next location prediction method for smartphones using blockmodels". In *Proc. of the IEEE Conference on Virtual Reality (VR)*, pages 1–4, 2013.

[49] J. C. Ferreira, V. Monteiro and J. L. Afonso. "Dynamic range prediction for an electric vehicle". In *Proc. of the World Electric Vehicle Symposium and Exhibition*, pages 1–11, 2013.

[50] C. De Cauwer, W. Verbeke, J. Van Mierlo and T. Coosemans. "A model for range estimation and energy-efficient routing of electric vehicles in real-world conditions". *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 7, 2787–2800, Jul 2020.

[51] Gideon Schwarz. "Estimating the dimension of a model". *Annals of Statistics*, vol. 6, no. 2, 461–464, mar 1978.

[52] Michal Piorkowski, Natasa Sarafijanovic-Djukic and Matthias Grossglauser. "CRAWDAD dataset epfl/mobility (v. 2009-02-24)". Downloaded from https://crawdad.org/epfl/mobility/20090224, feb 2009.

[53] "Kaggle data set ecml/pkdd 15: Taxi trajectory prediction (1)". Downloaded from https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data, apr 2015.