Model-based segmentation and classification of trajectories (Extended abstract)

Model-based Segmentation and Classification of Trajectories[*]

Sander P.A. Alewijnse[†] Kevin Buchin[*] Maike Buchin[‡] Stef Sijben[†] Michel A. Westenberg[*]

Abstract

We present efficient algorithms for segmenting and classifying a trajectory based on a parameterized movement model like the Brownian bridge movement model. Segmentation is the problem of subdividing a trajectory into parts such that each part is homogeneous in its movement characteristics. We formalize this using the likelihood of the model parameter. We consider the case where a discrete set of m parameter values is given and present an algorithm to compute an optimal segmentation with respect to an information criterion in $O(nm)$ time for a trajectory with n sampling points. Classification is the problem of assigning trajectories to classes. We present an algorithm for discrete classification given a set of trajectories. Our algorithm computes the optimal classification with respect to an information criterion in $O(m^2 + mk(\log m + \log k))$ time for m parameter values and k trajectories, assuming bitonic likelihood functions.

1 Introduction

Recent advances in tracking technology lead to increasing amounts of movement data being collected. For instance, animals are tracked to understand their movement behavior, vehicles are tracked for analyzing traffic situations, and sports players for analyzing their play. Movement data is typically recorded as a sequence of time-stamped positions, called *trajectory*. The analysis of large amounts of these data requires efficient algorithms. Computational movement analysis has emerged as a research field addressing this need [6].

Here we study two fundamental analysis tasks on trajectory data: segmentation and classification. A *segmentation* of a trajectory is a partition of a trajectory into *subtrajectories*, i.e., contiguous subsequences, called *segments*. These segments are disjoint –except for their endpoints, which are called *splitting points*– and cover the whole trajectory. A *classification* of a set of trajectories $\mathcal{T}$ is a partition of $\mathcal{T}$ into disjoint *classes* that cover $\mathcal{T}$.

Previous work on trajectory segmentation in computational geometry has focussed on *criteria-based segmentation*, where each segment fulfills given spatio-temporal criteria. An optimal criteria-based segmentation is one with a minimal number of segments. For this setting several algorithms have been proposed. Buchin *et al.* [4] developed a framework that computes a segmentation given a *decreasing monotone* criterion, that is a criterion which if it holds on a certain segment, also holds on every subsegment of that segment. In this framework a segmentation can generally be computed in $O(n \log n)$ time, where n is the number of sampling points. Although many natural criteria are decreasing monotone, not all are, and for this case Aronov *et al.* [2] developed an algorithm that runs in $\Theta(n^2)$ time. Recently, Alewijnse [1] proposed a framework that can efficiently handle both decreasing and increasing monotone (defined analogously to decreasing monotone) criteria in $O(n \log n)$ time. Criteria-based segmentation can also be used for classification, by using multiple criteria, one for each class. This setting has been successfully applied to a data set of migrating geese [5]. Recently, Sankararamana *et al.* [9] proposed to segment a trajectory by detecting similar subtrajectories[1].

Criteria-based segmentation and classification partition data based on pre-specified criteria. The motivation for this –and other movement analysis tasks– is in many cases to make inferences about the underlying movement process. In the light of this objective it seems only natural to take a more statistical perspective on these analysis tasks: As we describe in more detail below, trajectory segmentation and classification can be seen as fitting a parameterized movement model to the data.

Taking such an approach is essential when designing algorithms for applications –as in ecology– that use movement data in a statistical analysis. Therefore we now discuss movement models used in ecology. Movement models are used to infer a continuous motion from discrete samples of the movement path. In ecology, mostly random movement models [8], like the Brownian bridge movement model (BBMM) [7], are used. Recently, the BBMM has been introduced to computational movement analysis [3]. In these movement models, a link l has an associated log-likelihood function $L_l(x)$ as a function of the model parameter x. The log-likelihood of a parameter value for a set of links B (e.g. a segment or trajectory) is given by $L_B(x) = \sum_{l \in B} L_l(x)$. We emphasize that our methods do not apply only to the BBMM, but to any parameterized movement model that defines a likelihood as a function of the parameter value.

[*]This is an extended abstract of a presentation given at EuroCG 2014. It has been made public for the benefit of the community and should be considered a preprint.

[†]Dept. of Mathematics and Computer Science, TU Eindhoven, The Netherlands, s.p.a.alewijnse@student.tue.nl, {k.a.buchin,m.a.westenberg}@tue.nl

[‡]Dept. of Mathematics, Ruhr-Universität Bochum, Germany, {Maike.Buchin,Stef.Sijben}@rub.de

[1]For this they introduce a new model for similarity. The term *model* is used in our paper in a different sense, namely as referring to statistical models.

We use the term *partition* to refer to either a segmentation or classification, which additionally assigns a value $x(P)$ of the model parameter to each part P (i.e. a segment or class). A partition $\mathcal{P}$ has an associated log-likelihood

$$L_{\mathcal{P}} = \sum_{P \in \mathcal{P}} \sum_{e \in P} L_e(x(P)). \qquad (1)$$

That is, the log-likelihood of each part P (i.e., segment or class) is the sum over the log-likelihoods of its elements e (i.e., single links or trajectories), while the log-likelihoods of the parts are added to obtain the log-likelihood of $\mathcal{P}$. We could now define an optimal partition as one that maximizes the log-likelihood, but then it would be optimal to put each link or trajectory into its own part, since that allows optimizing the likelihood for each element separately. One solution is to compute the optimal partition with a fixed number of parts, but typically the number of parts is not known beforehand. To determine a good number of segments or classes an information criterion like Bayesian information criterion can be used.

To facilitate multi-scale analysis, we use a more general notion of an *information criterion* (IC) to define the optimal partition. An IC assigns a value to each partition based on its likelihood and the complexity of the model (that is, the number of parts). We consider ICs of the form

$$\mathrm{IC}(\mathcal{P}) = -2L_{\mathcal{P}} + |\mathcal{P}| \cdot p, \qquad (2)$$

where $L_{\mathcal{P}}$ is the log-likelihood of the model instance and $|\mathcal{P}|$ is the number of parts of the partition. The number p is a penalty factor for adding complexity to the model that counteracts overfitting.

We now define an *optimal segmentation* or *classification* to be one that minimizes the value of the IC among all segmentations or classifications and selections of model parameter values for the given input.

In this paper we develop efficient algorithms to compute such optimal segmentations and classifications. To avoid further assumptions on the log-likelihood functions and to simplify the problems we consider a discrete version of both problems. We assume that the parameter values $x(P)$ are drawn from a finite set of candidate values $X = \{x_1, \ldots, x_m\}$ (in sorted order) and that the log-likelihood functions are given by listing the values they take on $x_1, \ldots, x_m$. As an indication for the effectiveness of our methods, Figure 1 shows an example segmentation resulting from our algorithm.

Overview. In Section 2 we give an efficient algorithm for computing an optimal segmentation, and in Section 3 for computing an optimal classification.

Notation. We assume a *trajectory* τ is given by a sequence of n triples (x_i, y_i, t_i) $(1 \leq i \leq n)$, where $(x_i, y_i) =: \tau(i)$ is the location of a moving entity at time t_i. A subtrajectory of τ starting at time t_i and ending at time t_j we denote by $\tau[i,j]$. We use k to denote the number of segments in a segmentation, and $S_1, \ldots, S_k$ to denote the segments. Furthermore we denote with $x(S_i)$ the model parameter for each segment S_i. We use ℓ to denote the number of classes and $C_1, \ldots, C_\ell \subseteq \mathcal{T}$ to denote the classes. Again, each class C_i has an associated value of the model parameter $x(C_i)$. Each trajectory $\tau_i \in \mathcal{T}$ has an associated log-likelihood function $L_i(x)$, which is the sum of the log-likelihood functions of the links in τ_i.

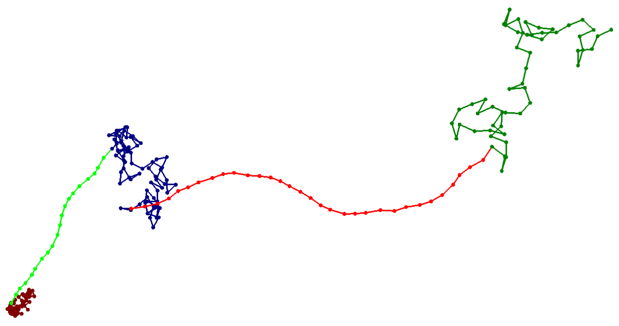


Figure 1: Segmentation of a randomly generated trajectory, with segments indicated by color.

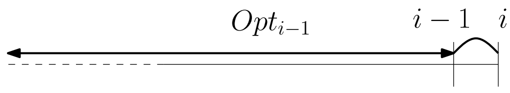
2 Segmentation

We present a new algorithm for trajectory segmentation based on a parameterized movement model, which computes an optimal segmentation with respect to a given information criterion IC. We solve the problem using dynamic programming. Let Opt_i denote the optimal segmentation of $\tau[0,i]$ with respect to IC. We compute all Opt_i in increasing order of i. In order to compute Opt_i we maintain a two-dimensional table O. Each entry $O_{i,x}$ stores the optimal segmentation (minimizing the IC) of subtrajectory $\tau[0,i]$ that ends with a segment S_k with parameter value $x(S_k) = x$.

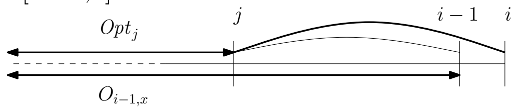
Given $O_{i,x}$ for all $x \in X$, it is straightforward to compute Opt_i: since every segment, in particular the last, has a parameter value in X, Opt_i is the entry $O_{i,x}$ that has minimal IC among all $x \in X$. The table O is computed using the following greedy property.

Lemma 1 *$O_{i,x}$ is equal to one of the following options:*

Append: Opt_{i-1} appended with the one-link segment $\tau[i-1,i]$.



Extend: $O_{i-1,x}$ with the last segment extended by $\tau[i-1,i]$.



Lemma 1 implies that we can compute $O_{i,x}$ in a dynamic programming fashion, looping over i and x. We compute each new entry $O_{i,x}$ using a comparison between two already computed table entries:

$$\begin{aligned}&\mathrm{IC}(O_{i,x})\\&\quad=\min\big(\mathrm{IC}(O_{i-1,x}),\mathrm{IC}(\mathit{Opt}_{i-1})+p\big)-2L_i(x).\end{aligned}$$

If $\mathrm{IC}(\mathit{Opt}_{i-1})+p$ is smaller, $O_{i,x}$ is set according to the append option. Otherwise it is set according to the extend option.

In our algorithm we store segmentations by storing only the length l and parameter value x for the last segment ending at $\tau(i)$. The previous segments can be worked out by looking at Opt_{i-l}. Furthermore, we store for each segmentation the information according to IC. Using this storage format computing $O_{i,x}$ given $O_{i-1,x}$ and Opt_{i-1} takes only constant time. The actual segmentation can be retrieved from the tables in $O(n)$ time. This leads to the following result.

Theorem 2 *The optimal segmentation of a trajectory τ with respect to an information criterion* IC *can be computed in $O(nm)$ time, and $O(n+m)$ space, where m is the number of candidate values for the model parameter and n the number of points on the trajectory.*

Fixed number of segments. It is straightforward to change the algorithm in such a way that it computes the segmentation with a fixed number k of segments and maximal likelihood. All tables get an extra dimension: the number of segments. Those tables can be computed using a greedy property that is similar to Lemma 1. To compute $O_{i,x,k}$ (with k the number of segments) we choose between either appending one link to $\mathit{Opt}_{i-1,k-1}$ and extending the last segment of $O_{i-1,x,k}$ by one link. Each table entry is hence computed in constant time, which results in $O(nmk)$ time and $O((n+m)k)$ space. Note that this does not only give the best segmentation with k segments, but also those with $k-1, k-2, \ldots, 1$ segments.

Table compression. When studying the DP tables (for BBMM) we made the following important observation: Let $\mathrm{last}(\mathcal{S})$ denote the starting index of the last segment of segmentation $\mathcal{S}$. Given an ordered set of parameter values $x_1 < x_2 < \cdots < x_m$ there seems to be only a constant (with respect to m and n) number of values x_j for which $\mathrm{last}\left(O_{i,x_j}\right) \neq \mathrm{last}\left(O_{i,x_{j+1}}\right)$.

We improve the algorithm by storing the DP table in a compressed way. With compressed tables and under some additional assumptions on the likelihood functions, the optimal segmentation can be computed in $O(nw)$ time and $O(n+w)$ space, where w is the maximum number of parameter values at which $\mathrm{last}(O_{i,x})$ changes for any i.

Continuous model parameter. So far, we have considered the situation where the model parameter takes values from a discrete set of candidate values. We now consider the case where the parameter can take any value from a continuous domain (e.g. an interval on the real line). Assume that the optimum model parameter $x(\tau[i,j])$ for a segment $\tau[i,j]$ can be computed in $F(j-i)$ time.

In this case, Opt_1 is empty, since $\tau[1,1]$ contains only a single point. Opt_j for $j>1$ consists of an optimal segmentation of $\tau[1,i]$ for some $i \in \{1,\ldots,j-1\}$, combined with a single segment for $\tau[i,j]$. The IC for each of the $j-1$ candidates for Opt_j can be computed in $O(F(j))$ time, since $\mathrm{IC}(\mathit{Opt}_i)$ is stored in the table for Opt and computing the optimum likelihood value for $\tau[i,j]$ takes $O(F(j))$ time.

Thus, Opt_n, the optimal segmentation of τ, can be computed in $O(n^2F(n))$ time. We only need to store the Opt table of size $O(n)$, so the algorithm uses $O(n)$ space, or what is needed to optimize the parameter value for a candidate segment.

For many practical log-likelihood functions like the one used in the BBMM, $x(\tau[i,j])$ and the corresponding log-likelihood can be computed to a fixed precision in linear time and space, i.e. $F(n)=O(n)$, leading to $O(n^3)$ time and $O(n)$ space. If the log-likelihood functions allow a closed-form expression of constant complexity, we can optimize each subtrajectory in amortized constant time and thus compute the optimal segmentation in $O(n^2)$ time.

3 Classification

In this section we present an algorithm for classifying a set of trajectories by grouping those that exhibit similar behaviour according to the movement model. For example, we can classify the segments resulting from the segmentation algorithm to detect recurring patterns in the data.

We now assume that the log-likelihood functions L_i are bitonic (first increasing to a maximum, then decreasing). We also assume without loss of generality that the functions are given in increasing order by the value at which they reach their maximum. That is, if we define $M_i = \arg\max_{x\in X}(L_i(x))$, then $i<j \Rightarrow M_i \le M_j$.

We represent a classification $\mathcal{C}$ for $L_1,\ldots,L_k$ by an array of length k where $\mathcal{C}[i]$ is the value that the classification assigns to the class of L_i, or $\mathcal{C}[i]=$ NIL if $\mathcal{C}$ is a partial classification that does not classify L_i.

We compute the optimal classification by dynamic programming. A natural approach would be to process the segments in the order they are given. However it is not necessarily the case that if $M_i < M_j$ that this order is reflected in the classes they are associated with. Figure 2 shows an example. However, we can use the following property to efficiently compute the optimal classification.

Observation 1 *Assume an optimal classification assigns the parameter values $x(C_1) < \cdots < x(C_\ell)$ to the classes. Then a trajectory that reaches its maximum likelihood in the interval $[x(C_j), x(C_{j+1}))$ will be either assigned to C_j or C_{j+1} by the bitonicity of the likelihood function. In particular if we know that some $x(C_i)$ is selected then $x(C_j)$ with $j<i$ does not depend on any of the trajectories with maximum larger or equal to $x(C_i)$.*

Consider a set of log-likelihood functions $\{L_1,\ldots,L_k\}$ and candidate parameter values $\{x_1,\ldots,x_m\}$. We add

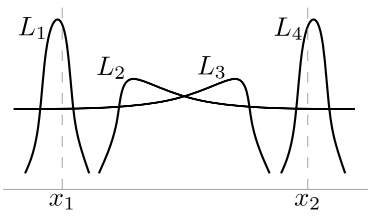


Figure 2: The optimal classification does not respect the order in which trajectories obtain their maximum likelihood. If the penalty factor is low, there will be two classes $C_1 = \{L_1, L_3\}$ and $C_2 = \{L_2, L_4\}$ with $x(C_1) = x_1$ and $x(C_2) = x_2$.

dummy values $x_0 := -\infty$ and $x_{m+1} := \infty$. Let $\mathcal{L}_{i,j} := \{L_l \mid x_i \leq M_l < x_j\}$ with $0 \leq i < j \leq m+1$ denote the set of functions that reach their maximum value between x_i and x_j. We now show that Algorithm 1 computes the optimal classification with respect to IC.

We define Opt_i as the optimal classification of $\mathcal{L}_{0,i}$, i.e. the input functions that reach their maximum likelihood at a value smaller than x_i, conditioned on the fact that the last class C_ℓ of Opt_i has parameter value $x(C_\ell) = x_i$.

We can iteratively construct Opt_i for $i = 1, 2, \ldots, m$ by observing that the second to last class (if Opt_i contains at least two classes) has a parameter value $x_j \in \{x_1, x_2, \ldots, x_{i-1}\}$. Thus, using Observation 1, Opt_i consists of Opt_j extended with a new class C_ℓ with $x(C_\ell) = x_i$. The functions that were already classified in Opt_j stay in the class they were assigned to, and the functions in $\mathcal{L}_{i,j}$ are assigned to either $C_{\ell-1}$ or C_ℓ, whichever results in the highest likelihood. The algorithm computes Opt_i by trying all possible values of j and selecting the resulting classification that has the lowest IC.

The optimal classification $\mathcal{C}$ of the whole input has a last class C_ℓ with $x(C_\ell) = x_i$ for some $i \in \{1, \ldots, m\}$. Then, $\mathcal{C}$ consists of an optimal classification of $\mathcal{L}_{0,i}$, conditioned on $x(C_\ell) = x_i$, and the remaining functions in $\mathcal{L}_{i,m+1}$ are assigned to C_ℓ by Observation 1. This optimal classification of $\mathcal{L}_{0,i}$ is exactly Opt_i, and we compute a candidate classification $\mathcal{C}_i$ from it by assigning all remaining functions to the last class. Then, $\mathcal{C}$ is the classification among the $\mathcal{C}_i$ that has minimum IC, which is what the algorithm returns.

Theorem 3 *Algorithm 1 computes the optimal classification of k trajectories with respect to an information criterion in $O(km^2)$ time and $O(km)$ space, where m is the number of candidate parameter values.*

Improved running time. By reusing the values of O_j and the corresponding IC computed in previous iterations of the outer loop, the running time of the classification algorithm can be reduced to $O(m^2 + km(\log m + \log k))$.

Continuous model parameter. We recently showed, that when the parameter values are taken from a continuous domain, finding the optimal classification is NP-hard

```
DISCRETECLASSIFICATION($(L_1, \ldots, L_k), (x_1, \ldots, x_m)$)
 1  $Opt_0 \leftarrow$ An array with all $k$ elements set to NIL
 2  $\mathcal{C} \leftarrow$ An arbitrary classification of $L_1, \ldots, L_k$
 3  for $i \leftarrow 1$ to $m$
 4      do for $j \leftarrow 0$ to $i-1$
 5             do $O_j \leftarrow Opt_j$
 6                for $L_l \in \mathcal{L}_{j,i}$
 7                    do $O_j[l] \leftarrow \arg\max_{x \in \{x_j, x_i\}} (L_l(x))$
 8         $Opt_i \leftarrow \arg\min_{\{O_j \mid 0 \leq j < i\}} \mathrm{IC}_i(O_j)$
 9         $\mathcal{C}_i \leftarrow Opt_i$ with all NILs replaced by $x_i$
10         if $\mathrm{IC}(\mathcal{C}_i) < \mathrm{IC}(\mathcal{C})$
11           then $\mathcal{C} \leftarrow \mathcal{C}_i$
12  return $\mathcal{C}$
```

Algorithm 1: Discrete classification.

for arbitrary bitonic functions. However, in many cases of practical interest, such as the BBMM, the optimal classification can be computed in polynomial time.

References

[1] S. P. Alewijnse. A framework for trajectory segmentation by stable criteria and Brownian bridge movement model. Master's thesis, Eindhoven University of Technology, 2013.

[2] B. Aronov, A. Driemel, M. J. van Kreveld, M. Löffler, and F. Staals. Segmentation of trajectories for non-monotone criteria. In *Proc. 24th ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 1897–1911, 2013.

[3] K. Buchin, S. Sijben, T. J. Arseneau, and E. P. Willems. Detecting movement patterns using Brownian bridges. In *Proc. 20th Internat. Conf. Advances in Geogr. Inf. Syst.*, pages 119–128. ACM, 2012.

[4] M. Buchin, A. Driemel, M. van Kreveld, and V. Sacristán. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *Journal of Spatial Information Science*, 3:33–63, 2011.

[5] M. Buchin, H. Kruckenberg, and A. Kölzsch. Segmenting trajectories based on movement states. In *Proc. 15th Internat. Sympos. Spatial Data Handling (SDH)*, pages 15–25. Springer-Verlag, 2012.

[6] J. Gudmundsson, P. Laube, and T. Wolle. Computational movement analysis. In W. Kresse and D. M. Danko, editors, *Springer Handbook of Geographic Information*, pages 423–438. Springer, 2012.

[7] J. Horne, E. Garton, S. Krone, and J. Lewis. Analyzing animal movements using Brownian bridges. *Ecology*, 88(9):2354–2363, 2007.

[8] N. E. Humphries *et al.* Environmental context explains Lévy and Brownian movement patterns of marine predators. *Nature*, 465:1066–1069, 2010.

[9] S. Sankararaman, P. K. Agarwal, T. Mølhave, J. Pan, and A. Boedihardjo. Model-driven matching and segmentation of trajectories. In *Proc. 21st Internat. Conf. Advances in Geogr. Inf. Syst.* ACM, 2013.