Machine learning for vessel trajectories using compression, alignments and domain knowledge

Gerben Klaas Dirk de Vries *, Maarten van Someren

Informatics Institute, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands

A R T I C L E  I N F O

Keywords:
Vessel trajectories
Moving object trajectories
Piecewise linear segmentation
Alignment measures
Geographical domain knowledge

A B S T R A C T

In this paper we present a machine learning framework to analyze moving object trajectories from maritime vessels. Within this framework we perform the tasks of clustering, classification and outlier detection with vessel trajectory data. First, we apply a piecewise linear segmentation method to the trajectories to compress them. We adapt an existing technique to better retain stop and move information and show the better performance of our method with experimental results. Second, we use a similarity based approach to perform the clustering, classification and outlier detection tasks using kernel methods. We present experiments that investigate different alignment kernels and the effect of piecewise linear segmentation in the three different tasks. The experimental results show that compression does not negatively impact task performance and greatly reduces computation time for the alignment kernels. Finally, the alignment kernels allow for easy integration of geographical domain knowledge. In experiments we show that this added domain knowledge enhances performance in the clustering and classification tasks.

© 2012 Published by Elsevier Ltd.

1. Introduction

The past decade, tracking of people and objects in geographical space has become ubiquitous. Smart phones have GPS sensors and cars are equipped with navigation systems. All of this tracking data can easily be stored, generating a type of data called the moving object trajectory.

In the maritime domain, vessels can be tracked with GPS or radar. Consider all the vessel movements that occur around a major port, like the busy harbors of Singapore and Rotterdam. Grouping these movements into clusters of similar behavior can help to get an overview of the general movement patterns. This overview can assist the operator to better spot irregular movements. Another example are trajectories made by fishing vessels. These trajectories are different from the paths generated by tankers or cargo ships, which follow regular routes. Identifying whether a trajectory belongs to one or the other class can help in determining whether a vessel exhibits unwanted behavior and thus should be further investigated. Performing tasks like these is part of a Maritime Safety and Security (MSS) system.

The overall aim of an MSS system is to acquire, store and analyze data and information from different sources and enable users to analyze and operate on this data and information. The main source of data is vessel tracking information. Potentially, the analysis of vessel trajectories can be further enhanced by integrating information from other sources such as geographical domain knowledge about harbors, anchoring areas, etc.

One solution to perform analysis of vessel trajectories is to create models by applying machine learning techniques to (large) sets of vessel trajectory data. Different types of analysis using machine learning are useful. To gain insight into different vessel behaviors, groups of similar movements need to be created, which is a clustering task. Predicting existing properties, such as the vessel type, is another form of analysis, which is a classification task. Identifying strange trajectories, i.e. irregular behavior, among a large group of normal trajectories is an outlier detection task.

A major source of vessel tracking information is the Automatic Identification System (AIS). AIS is an automatic tracking system onboard large vessels, which sends out position updates frequently. Sequences of AIS messages can be transformed into vessel trajectories. These vessel trajectories have the typical properties of moving object trajectories. They are sequences of differing number of samples, temporal length and distance traveled.

The behavior of vessels is very regular relative to the number of AIS messages that are transmitted. This regularity provides the opportunity to apply compression to the vessel trajectories. We use trajectory compression based on line-simplification, which we adapt such that the important stop and move information is retained for trajectories.

For the clustering, classification and outlier detection tasks we take a similarity based approach, which fits well with the nature of vessel trajectories, and moving objects in general. We define
similarity measures based on sequence alignment techniques. Trajectory compression speeds up the computation of these similarities. However, compression can have a potentially negative impact on task performance.

The space in which vessel trajectories exist contains places and regions such as, harbors, shipping lanes and anchoring areas. Integrating this geographical domain knowledge into the trajectories and the similarity measures can potentially improve the clustering, classification and outlier detection tasks.

The rest of this paper is structured as follows. Section 2.1 introduces some preliminaries needed for the rest of the paper. Compression for vessel trajectories is discussed in Section 2, with experimental results showing that our adaptation is better at retaining stop information. Similarity measures based on alignments are introduced in Section 3. The experiments in this section show that compression does not negatively influence the performance in the tasks of clustering, classification and outlier detection for vessel trajectories. In Section 4 we add geographical domain knowledge to the similarity measures and show the influence in clustering, classification and outlier detection. We end with conclusions and suggestions for future work.

2. Compression

Because of the high rate at which AIS data is collected, the amount of AIS data is large. For example, one week of monitoring vessels in front of the Dutch west coast results in approximately 4 gigabyte of data for around 3000 different vessels. Vessels usually move in a rather regular way. Because of the large amount of data and the regularity we can and need to compress the data to a much smaller volume without losing important information. The compressed trajectories can be used for clustering, classification and outlier detection. In fact, the ultimate criterion for the quality of compression is in the extent to which the data can be compressed without damaging the use of the trajectory data for further processing.

2.1. Preliminaries

The vessel trajectory data that we use for the experiments in this paper are collected using the Automatic Identification System (AIS).

Using AIS messages a moving object trajectory of a vessel can be constructed. A moving object trajectory \( T \) in 2-dimensional space is represented by a sequence of vectors: \( T = \langle x_1, y_1, t_1 \rangle, \langle x_2, y_2, t_2 \rangle, \ldots, \langle x_n, y_n, t_n \rangle \), where \( x_i \) and \( y_i \) represent the position of the object at time \( t_i \) and \( t_{i+1} > t_i \). The number of vectors of a trajectory is denoted \( n \). Furthermore, let \( T(i) = \langle x_i, y_i, t_i \rangle \) and \( T(j) = \langle x_j, y_j, t_j \rangle \). The sample rate of trajectories is not always fixed, thus the difference between consecutive values \( c_i, t_{i+1} \) is not the same. In some tasks, there are more dimensions to trajectories that can be derived from the \( x,y,t \) information, such as speed and direction.

These dimensions can just be added to the \( \langle x,y,t \rangle \) vector as extra variables. In the following we refer to a vector \( \langle x_i, y_i, t_i \rangle \) as a trajectory point. In the analysis of trajectories the concepts of stop and move are essential (Saccapietra et al., 2008). What should be considered stops and moves is application dependent. A stop is the time interval of a trajectory for which the traveling object does not move from the application’s perspective, the rest of the trajectory is considered a move.

2.2. Piecewise linear segmentation

In the field of moving object databases different techniques have been studied to compress trajectory data (Cao, Wolfson, & Trapecevski, 2006; Frentzos & Theodoridis, 2007; Gudmundsson, Katajainen, Merrick, Ong, & Wolle, 2009; Meratnia & de By, 2004; Ni & Ravishankar, 2007; Potamias, Patroumpas, & Sellois, 2006). The most common method is Piecewise Linear Segmentation (PLS). It has been invited multiple times in different fields (Douglas & Peucker, 1973; Ramer, 1972; Keogh, Chu, Hart, & Pazzani, 2001). This method is intuitive, easy to implement, relatively fast (compared to optimal methods (Cao et al., 2006)) and gives good results.

The PLS-algorithm compresses a trajectory \( T \) into linear segments by recursively keeping the points that have maximum error higher than a fixed threshold \( \epsilon \). The pseudo code for this algorithm is given in Algorithm 2.1. It works in the following way. The first and last points, \( T(1) = \langle x_1, y_1, t_1 \rangle \) and \( T(n) = \langle x_n, y_n, t_n \rangle \), of a trajectory \( T \) of length \( n \) are selected, and for all intermediate points we compute the error, using the function \( E \), with respect to these points. If the maximum of these errors is greater than a certain threshold, then we apply the procedure again, with the corresponding point \( T(j) \) both as start and end-point, i.e. we recursively apply the procedure to the trajectory from \( T(1) \) to \( T(j) \) and \( T(j) \) to \( T(n) \). When there is no error greater than the given threshold, then we just keep the points \( \langle x_1, y_1, t_1 \rangle \) and \( \langle x_n, y_n, t_n \rangle \). Thus, the goal of the algorithm is to reduce the number of points in a trajectory while keeping the maximum deviation, or error, from the original trajectory within the threshold \( \epsilon \).

Algorithm 2.1: pls(\( T, \epsilon \))

1. We use end to indicate the index of the last element of a trajectory.
2. \( d_{max} = 0 \)
3. \( l_{max} = 0 \)
4. for \( i = 2 \) to \( \text{end} - 1 \) do
5. \( d = E(T(i), T(1), T(\text{end})) \)
6. if \( d > d_{max} \) then
7. \( l_{max} = i \)
8. \( d_{max} = d \)
9. if \( d_{max} > \epsilon \) then
10. \( A = \text{pls}(T(1, l_{max}), \epsilon) \)
11. \( B = \text{pls}(T(l_{max}, \text{end}), \epsilon) \)
12. \( T_c = A, B, (\text{end}) \)
13. else
14. \( T_c = T(1), T(\text{end}) \)
15. return \( T_c \)

For trajectories, a number of different error functions \( E \) (Algorithm 2.1, line 5) are possible. Most recently, (Gudmundsson et al., 2009) proposed an error function \( E_p \) that generalizes most of the earlier defined error functions, so we consider this one: see Eq. (1) below.

\[
E_p(x_i, y_i, t_i, \langle x_1, y_1, t_1 \rangle, \langle x_n, y_n, t_n \rangle) = \| (x_i, y_i, t_i) - (x'_i, y'_i, t'_i) \|.
\]
where \((x_i', y_i', t_i')\) is the closest point to \((x_i, y_i, t_i)\) on the line-segment \((x_1, y_1, t_1), (x_0, y_0, t_0)\).

The parameter \(\mu\) determines the ratio between the space and time dimensions. Different settings of \(\mu\) lead to different previously defined error functions, \(E_2 = E_{\mu=0}\) (Douglas & Peucker, 1973), \(E_2 = E_{\mu=1}\) (Cao et al., 2006) and \(E_2 = E_{\mu=\infty}\) (Cao et al., 2006), which treat time differently. With \(E_2\), a trajectory is treated as a line in a 2-dimensional space, ignoring the time dimension. The function \(E_3\) treats time as just another dimension, i.e. a trajectory is a line in a 3-dimensional space. Finally, \(E_4\) puts the biggest emphasis on time, because we take the difference between a point and its linear interpolation based on time on the line-segment. We will use a subscript to indicate which error function is used with PLS, for example \(\text{pls}_{E_2}\).

Compression with PLS using any of these error measures can lead to problems with regards to retaining stops in trajectories. It is possible for a trajectory to be reduced in such a way that in the compressed trajectory it appears as if the vessel moves slowly, whereas in the original uncompressed trajectory the vessel stopped moving for a period of time. Knowing whether a vessel stops or not is important in the behavior analysis of vessels, thus we would like to retain this information during compression.

To deal with this problem we extend the standard algorithm. The intuition behind this extension is that the stopping and moving behavior of moving objects is more apparent in the derivative, i.e. the speed of the moving object, of the trajectory than in the trajectory itself. Therefore we propose a simple extension to the earlier defined trajectory compression, which we give in Algorithm 2.2. The idea is that first piecewise linear segmentation is applied to the speed time-series of the trajectory, using essentially a one-dimensional variant of the \(E_\mu\) error measure. We call this error measure \(E_l\) and define it below for trajectories \(T_n\) i.e. speed is added to the trajectory points. Formally, \(T_n = (x_1, y_1, t_1, v_1), \ldots, (x_n, y_n, t_n, v_n)\), where \(v_i\) is the speed at \(t_i\).

\[
E_l((x_i, y_i, v_i, t_i), (x_{i+1}, y_{i+1}, t_{i+1}, v_{i+1}), (x_n, y_n, v_n, t_n)) = \sqrt{(t_{i+1} - t_i)^2} \quad (2)
\]

where \((x_i', y_i', v_i', t_i')\) is the point on the line-segment \((x_i, y_i, t_i, v_i), (x_{i+1}, y_{i+1}, t_{i+1}, v_{i+1})\) with time \(t_i\). Essentially we take the difference between the actual speed \(v_i\) and the linearly interpolated speed \(v_i'\).

To the resulting subtrajectories, which are created in this speed-only segmentation step, we apply regular PLS with an error measure of our choosing: \(E_2, E_3, E_\mu\) or \(E_{\mu=0}\). Should we skip the second step, or alternatively just set \(s = \infty\), then we have trajectory compression purely based on speed. As we did for PLS we indicate with a subscript which error measure is used in the second step, e.g. \(2\text{-stage-pls}_{E_2}\).

**Algorithm 2.2: 2stage-pls**

1. We use \(\text{end}\) to indicate the index of the last element of a trajectory.
2. \(A_p = \text{pls}_{E_\mu}(T_n, \epsilon_p)\)
3. \(A = \emptyset\)
4. for all consecutive points \(p_i, p_j\) in \(A_p\) do
5. select from \(T_n\) the subtrajectory \(T(i, j)\) as \(p_i, \ldots, p_j\)
6. \(A_p = \text{pls}(T(i, j), \epsilon_p)\)
7. \(A = A(A, \text{end} - 1), A_p\)
8. return \(A\)

An important point in research into trajectory compression techniques are the error bounds that different error measures give on a number of standard moving object database queries. Regarding these bounds, (Gudmundsson et al., 2005) generalizes the work of Cao et al. (2006). It shows the error bounds for the \(E_\mu\) measure for the Where-At (where is the object at time \(t\)) and When-At (when is the object at position \(x, y\)) queries and how these are dependent on \(\mu\). It is easy to see that if we use \(2\text{-stage-pls}_{E_2}\), we have the same error bounds, because for every subtrajectory created by the speed compression, these error bounds hold since these subtrajectories are compressed using \(E_2\).

The error bounds for \(E_\mu\) vanish when \(\mu = 0\). Consequently, the bounds for \(2\text{-stage-pls}_{E_{\mu=\infty}}\) are not good. The error for the first speed compression step is bounded, however, an error in speed leads to a quadratic error in position, which is not further bounded for \(\mu = 0\).

The worst case running time of standard piecewise linear segmentation is \(O(n^2)\). The \(2\text{-stage-pls}\) algorithm is in the same order of complexity, because it is essentially two applications of PLS. The first (speed) step is just a regular full recursion of PLS and we can consider the second step as if we are already at a certain level of the recursion of PLS. Thus, the worst case running time of \(2\text{-stage-pls}\) is \(O(n^2)\).

### 2.3. Experiments

The aim of our two-stage PLS algorithm is to provide better stop retention at the same compression rate. Thus, for the two algorithms, the different error measures and a range of parameter settings we compute the stop retention on a hand-labeled dataset. This set contains, for instance, a trajectory from a ferry between two ports and a ship docking in a harbor. However, this set is biased towards trajectories with stops. To get a more general compression result for our data we compute the compression rate for the different settings on a regular unbiased dataset containing 600 randomly selected trajectories. To reduce the influence of noise in the AIS data, caused by for instance GPS, we apply standard moving average smoothing to the trajectories.

The measure of successful stop retention is defined as follows. If a compressed trajectory \(T_c\) contains a segment \(T_c(i, i + 1)\), such that

\[
\frac{\text{dist}(x_i, y_i, (x_{i+1}, y_{i+1}))}{(t_{i+1} - t_i)} \leq \epsilon
\]

then that segment is a stop. In other words, a vessel is stopped if the average speed in a segment is below a certain threshold. If the temporal interval \(t_i, t_{i+1}\) overlaps with the temporal interval of a hand-labeled stop (i.e. they have at least one time-point in common), then we say that the compression has retained that stop.

Apart from stops that are not retained, the opposite can also occur. The average speed in a segment can be below the stop threshold, but it is not labeled as a stop. We do not consider this option in the stop retention ratio definition, since it did not occur in our experiments.

As baseline methods we will look at the regular PLS algorithm with the error measures: \(E_2\), \(E_3\), \(E_\mu\), \(E_{\mu=1}\), \(E_{\mu=\infty}\). We recall that the settings \(E_{\mu=0,1,\infty}\) are equal to \(E_2, E_3, E_{\mu=0}\) respectively. \(E_{\mu=1}\) is in between \(E_2\) and \(E_3\), and \(E_{\mu=2}\) is in between \(E_3\) and \(E_{\mu=\infty}\).

Before any of these measures can be applied, the time dimension needs to be scaled to the spatial dimensions. Typically the time is in seconds, and the spatial dimension in arc lengths or kilometers. The intuition between the measure \(E_2/E_{\mu=1}\) is that differences in the temporal and spatial dimensions can be equally compared. Thus, we have scaled the temporal dimension such that the average temporal difference between samples, in the original scale of the temporal dimension, equals the average difference in position, under the \(E_2/E_{\mu=1}\) setting.

All of the above baselines are compared to our two-stage method \(2\text{-stage-pls}\). For this comparison we use the \(E_{\mu}\) setting that scores the best among our baselines.

For all experiments, the \(\epsilon\) settings that we range over are the same. For the error measures \(E_2, E_3, E_\mu\) and \(E_{\mu=\infty}\), \(\epsilon\) ranges from 0.01
to 0.08 km with increments of 0.01. And for \( E_v \), \( \epsilon_v \) ranges from 0.5 to 4 knots, with increments of 0.5.

Results. From the large number of different settings we tested, we present the most important results. For a complete overview of all the results we refer the reader to de Vries (2012).

From the baselines we show the results for the \( E_\mu = 2 \) setting, the best performing setting in terms of stop retention versus compression rate. Furthermore, we give the results for \( 2stage-pls \) with \( E_\mu = 2 \). However, \( 2stage-pls \) has two parameters, thus there are 8 lines, representing the different settings of \( E_v \). For each subfigure it holds that the highest line in the graph is for \( E_v = 0.5 \), the lowest line for \( E_v = 4 \) and the rest of the settings are in between.

First, we note from Fig. 1 that using piecewise linear segmentation leads to a large data reduction with compression rates over 90% for the realistic \( \epsilon_v \) settings that we used.

Furthermore, we see that this variant of \( 2stage-pls \) outperforms regular \( pls_{E_\mu = 2} \). This is the case for almost all parameter settings. Thus, what we do with two stage PLS, adding an initial speed compression step to the best performing baseline, retains more stops at the same compression rate.

We also see that the performance of \( 2stage-pls_{E_\mu} \) does not degrade much when we lower the stop threshold. This suggest that the added speed compression step of \( 2stage-pls \) adds robustness when it comes to stop retention.

3. Trajectory similarities

There are multiple approaches for performing the clustering, classification and outlier detection tasks on vessel trajectory data. We decided to take a similarity measure based approach. We use similarities defined on alignments between the points of two trajectories. Alignment measures are flexible in dealing with trajectories of different length in terms of the number of points, time, and distance traveled. The compression detailed in the previous section reduces the number of points substantially, which makes alignment computation faster. However, compression can have a potentially negative influence on the quality of the alignments. We investigate this influence, and the performance of alignment measures in general, in this section.

Different alignment measures exist: Dynamic Time Warping (DTW) (Vlachos, 2004), and various forms of Edit Distance (ED), such as edit distance with real penalties (Chen et al., 2004), edit distance on real sequences (Chen et al., 2005) and Longest Common SubSequence (LCSS) (Vlachos, Gunopoulos, & Kollios, 2002; Vlachos, Kollios, & Gunopulos, 2005). The DTW similarity measure has its origin in the time-series literature, whereas edit distance measures were originally defined for strings of characters.

Another class of trajectory similarities is based on taking the integral over time of the distance function between trajectories (e.g. (Buchin, Buchin, van Kreveld, & Luo, 2009; Nanni & Pedreschi, 2006; van Kreveld et al., 2007)). For a comparison with these measures we refer to de Vries (2012).

The similarities that we used are defined as kernels, so that they can be used in kernel methods for clustering (kernel k-means), classification (Support Vector Machines) and outlier detection (one-class Support Vector Machines). For more details on their implementation as kernels we again refer to de Vries (2012).
3.1. Trajectory alignment kernels

We investigate two alignment measures, dynamic time warping and a type of edit distance, very similar to edit distance with real penalties.

Dynamic Time Warping (DTW) is a very popular alignment method. A DTW alignment $\pi$ of $p$ positions, where $p > 0, p < |S|$ and $p \in \mathbb{T}$, between two trajectories $S$ and $T$ is a pair of $p$-tuples:

$$\pi = ((\pi_1(1), \ldots, \pi_1(p)), (\pi_2(1), \ldots, \pi_2(p))) \in \mathbb{N}^{2p},$$

(4)

with the constraints that there are unitary increments and no simultaneous repetitions, thus $\forall 1 \leq i \leq p - 1$,

$$\pi_1(i + 1) \leq \pi_1(i) + 1, \quad \pi_2(i + 1) \leq \pi_2(i) + 1$$

(5)

and

$$((\pi_1(i + 1) - \pi_1(i)) + (\pi_2(i + 1) - \pi_2(i)) \geq 1.$$  

(6)

Furthermore,

$$1 = \pi_1(1) \leq \pi_1(2) \leq \cdots \leq \pi_1(p) = |S|,$$

(7)

$$1 = \pi_2(1) \leq \pi_2(2) \leq \cdots \leq \pi_2(p) = |T|.$$  

(8)

This means that all elements in both trajectories are aligned, which might require repeating elements from a trajectory, but in the alignment we cannot simultaneously repeat an element in both trajectories. Furthermore, the start and end of trajectories are aligned by default.

Edit distances are computed in terms of the number of substitutions, deletions and insertions that are required to transform one sequence, usually a string, into another sequence. We define a version for continuous elements similar to how (Chen et al., 2004) defines it for time-series. An edit distance alignment $\pi$ of $p$ positions, where $p > 0, p < |S|$ and $p \in \mathbb{T}$, between two trajectories $S$ and $T$ is a pair of $p$-tuples:

$$\pi = ((\pi_1(1), \ldots, \pi_1(p)), (\pi_2(1), \ldots, \pi_2(p))) \in \mathbb{N}^{2p},$$

with the constraints:

$$1 \leq \pi_1(1) \leq \pi_1(2) \leq \cdots \leq \pi_1(p) \leq |S|,$$

(10)

$$1 \leq \pi_2(1) \leq \pi_2(2) \leq \cdots \leq \pi_2(p) \leq |T|.$$  

(11)

This means that not all elements have to be aligned and there is no repetition of elements.

The score function for an alignment $\pi$ of two trajectories $S$ and $T$, using a substitution function sub and gap penalty $g$, is equal to:

$$s_{\text{sub}, \pi}(S, T, \pi) = g \cdot (||S| - |\pi_1||) + (||T| - |\pi_2||)$$

$$+ \sum_{i=1}^{p} \text{sub}(S(\pi_1(i)), T(\pi_2(i))).$$

(12)

This score sums the substitution scores computed by sub for all aligned elements, and adds a constant gap penalty for all elements that are not aligned. Considering the two types of alignments that are defined above, we note that the difference is in how they treat gaps. In the DTW case there are no gaps. What would be a gap in the edit distance case is treated in DTW by repeating a trajectory element, and thus gets a score according to the substitution function sub. In the edit distance case gaps have a fixed penalty $g$. Thus, the value of $g$ has no influence for $s_{\text{sub}, \pi}$ if $\pi$ is a DTW alignment. An edit distance alignment between two trajectories is illustrated in Fig. 2.

On these two types of alignments we will define three kinds of similarity measures. The first two measures are the common implementations of alignment measures. They use the substitution function sub, which is the negative of the $L^2$ norm:

$$\text{sub}(\langle x_i, y_i, t_i \rangle, \langle x_j, y_j, t_j \rangle) = -\|x_i - x_j, y_i - y_j, t_i - t_j\|.$$  

(13)

However, it has been argued (Cuturi, Vert, Birkenes, & Matsui, 2007; Vert, Saigo, & Akutsu, 2004) that taking the soft-max of the scores of all possible alignments leads to a better similarity, because all possible alignments are taken into account. Moreover, this soft-max version defines a proper positive semi-definite kernel, if the we use the substitution function sub2, which is the negative of the squared $L^2$ norm:

$$\text{sub}_2(\langle x_i, y_i, t_i \rangle, \langle x_j, y_j, t_j \rangle) = -\|x_i - x_j, y_i - y_j, t_i - t_j\|^2.$$  

(14)

As mentioned, using the defined alignments and score function, similarity measures can be computed in multiple ways. One option for similarity between two trajectories is to take the score of the alignment that maximizes the score function $s$. Given two trajectories $S$ and $T$, let $\Pi_{\text{max}}(S, T)$ be the set of all possible alignments between these trajectories under a certain alignment measure $\phi$ and $g \leq 0$, then

$$\text{sim}_{\text{max}}(S, T, \pi) = \max_{\pi \in \Pi_{\text{max}}(S, T)} s_{\text{sub}, \pi}(S, T, \pi).$$

(15)

This score can also be normalized by dividing by the sum of the lengths of $S$ and $T$, resulting in the average score per element. Given two trajectories $S$ and $T$, let $\phi$ be an alignment measure and $g \leq 0$, then

$$\text{sim}_{\text{maxnorm}}(S, T, \pi) = \frac{\text{sim}_{\text{max}}(S, T, \pi)}{|S| + |T|}.$$  

(16)

Finally, we can also take the soft-max of the score of all possible alignments (Cuturi et al., 2007). Given two trajectories $S$ and $T$, let $\Pi_{\text{allmax}}(S, T)$ be the set of all possible alignments between these trajectories under a certain alignment measure $\phi, g \leq 0$, and $\beta > 0$ then

$$\text{sim}_{\text{softmax}}(S, T, \pi) = \frac{1}{|S| + |T|} \sum_{\pi \in \Pi_{\text{allmax}}(S, T)} \exp(\beta s_{\text{sub}, \pi}(S, T, \pi)).$$

(17)

This similarity function has scaling parameter $\beta$ and uses substitution function sub2.

We have to put these similarity measures into kernel form to be able to use them in Support Vector Machines and kernel $k$-means. For a set of trajectories $T$ and the max or maxnorm alignment similarity, we compute the kernel matrix $K$ as:

$$K(i, j) = \text{sim}(S, T),$$

(18)

where $i$ and $j$ are indexes for $S, T \in T$. Furthermore, we normalize and make a kernel out of $K$ by:

$$K = 1 - \frac{K}{\text{min}(K)}.$$  

(19)

For a set of trajectories $T$ and the soft-max alignment similarity, we compute the kernel matrix $K$ as:

$$K(i, j) = \frac{\text{sim}(S, T)}{\sqrt{\text{sim}(S, S) \text{sim}(T, T)}},$$  

(20)

where $i$ and $j$ are indexes for $S, T \in T$. 

Fig. 2. Example of an edit distance alignment between two (partial) trajectories. Points in two trajectories that are aligned are indicated with a substitution, points that are not aligned with a gap.
3.2. Experiments

The goal of the experiments below is to investigate the performance of the different alignment kernels that we defined above on three typical machine learning tasks, clustering, classification and outlier detection. We are interested in which measure performs best overall and more specifically what the influence is of applying trajectory compression on task performance. We expect that compression negatively impacts alignment similarities, since compression can remove points in one trajectory that align well with points in another trajectory. This would lead to bad alignment scores, whereas in uncompressed form the trajectories would align well.

Using the two-stage compression algorithm described in Section 2 we cut the trajectories at the stops for each of the datasets that we describe for the three different tasks below. Thus the trajectories that we consider are moves in terms of the stop-move model of Spaccapietra et al. (2008). For all trajectories, $t_i$ is set to 0, thus we consider only the relative time of trajectories.

To investigate the influence of trajectory compression on the performance of the three tasks we used 6 compression settings for Piecewise Linear Segmentation (PLS) with Algorithm 2.1 and error measure $E_{p,2}$, i.e. $\text{pls}_{p,2}$. Since the datasets only consist of moves, we use regular PLS in the following experiments. We set the $\mu$ parameter to the best stop retention versus compression rate value from Section 2 and did not vary it for these experiments, since we found it had little effect on the overall outcome. The first of the compression settings is $\epsilon = 0$, thus, we apply no compression. In the other settings we applied $\text{pls}_{p,2}$ to each trajectory with $\epsilon = 10, 50, 100, 1000, 10000$ m, respectively. The high epsilon settings are not realistic application settings, since they reduce trajectories that we consider are moves in terms of the stop-move model. As weights we took $w = 0, 1, 2$. With $w = 0$ we ignore the time dimension and with the other weights we increasingly weigh the time dimension more heavily. The $w = 1$ setting means that the average difference between two points in the space dimension is roughly equal to the average difference in the time dimension, cf. $E_{2}$ in Section 2.2.

As parameters we use $\beta = 1, 1.6, 16, 64, 256, 1024$ and $g = -0.01, -0.025, -0.05, -0.075, -0.1$. For the soft-max kernels we use $g = -0.01^2, -0.025^2, -0.05^2, -0.075^2, -0.1^2$.

Table 1 gives some results on running times. These computation times are for running our Matlab Mex implementation of the DTW alignment kernels in the clustering experiment on an AMD Phenom II X6 1055 (3.2 Gzh) cpu system with 8 GB of ram.

<table>
<thead>
<tr>
<th>$\epsilon$ = 0 m</th>
<th>10 m</th>
<th>50 m</th>
<th>100 m</th>
<th>1000 m</th>
<th>10000 m</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{\max,DTW}$</td>
<td>172 s</td>
<td>1.4 s</td>
<td>0.9 s</td>
<td>0.9 s</td>
<td>0.9 s</td>
</tr>
<tr>
<td>$K_{\maxnorm,DTW}$</td>
<td>172 s</td>
<td>1.4 s</td>
<td>0.9 s</td>
<td>0.9 s</td>
<td>0.9 s</td>
</tr>
<tr>
<td>$K_{\softmax,DTW}$</td>
<td>239 s</td>
<td>1.7 s</td>
<td>1.0 s</td>
<td>0.9 s</td>
<td>0.9 s</td>
</tr>
</tbody>
</table>

All similarities are computed using a dynamic programming approach, as is standard for the max and max-norm alignment similarities. This approach can also be used for the soft-max alignments (Cuturi et al., 2007).

3.2.1. Clustering

The goal of the clustering task is to investigate which trajectory kernel best reconstructs a manually created gold standard clustering. We created a dataset of a 25 km radius consisting of 714 vessel trajectories around the Texel island in the Netherlands. These trajectories are partitioned into 8 different clusters, creating a gold standard $G = g_1, \ldots, g_8$. The clusters are very different in size, ranging from 8 to 348 trajectories. The average length of a trajectory is a sequence of $300 (x, y, t)$ vectors. We illustrate the clustering in Fig. 3.

The general direction of movement for each of the 8 clusters is indicated with an arrow. These clusters are unlabeled, but they can be given descriptive labels, such as “leaving port to the north” for 3 and “from Den Helder to Texel” for 8. Together, these 8 clusters give a summary, in terms of the different types of movements/behavior, of what goes on around Texel island.

The kernels defined above are used as input for the weighted kernel $k$-means algorithm (Dhillon, Guan, & Kulis, 2007), computing the ‘normalized graph cut’. This algorithm is a kernel variant of the standard $k$-means algorithm. It produces a clustering $C = c_1, \ldots, c_k$ of the input data into $k$ clusters. For each kernel, clustering is done 100 times with random initializations and the clustering with the lowest intra cluster spread is kept. This process is repeated 10 times. We set $k = 8$, i.e. the same number of clusters as in the gold standard. The function $\text{FT}$ (Liao, 2005), defined as:

$$\text{FT}(C, G) = \frac{1}{k^2} \sum_{i=0}^{k-1} \max_{l=0}^{k-1} \frac{2|g_i \cap c_l|}{|g_i| + |c_l|},$$

is used to evaluate the quality of a clustering. This function takes the best $F1$-score for each cluster and averages over these scores.

Results. Table 2 presents the results per kernel type. For each kernel, we give the mean, minimum and maximum scores computed over all the parameter settings. The top row of the tables indicate the different compression settings that were used. The mean score of 10 random clusterings is 0.14.

The best score (0.82) for a dynamic time warping kernel is achieved by the max norm kernel. For some of the $K_{\max,DTW}$ kernel settings the performance under compression is better than the $K_{\maxnorm,DTW}$ kernel.
performance with no compression. The same holds for all the DTW
soft-max kernels. However, we must remark here that the soft-
max kernels are outperformed by their regular (max and max
normal) counterparts.

The best score was achieved by the \(K_{\text{max,E}}\) kernel. Scores almost
as good as the best score can be achieved by reducing the trajecto-
ries to just two points, i.e. the start and end, and using these two to
calculate similarity. Inspecting the dataset suggests that, to deter-
mine the cluster of a trajectory indeed this information is almost
sufficient.

3.2.2. Classification

In the classification task we use the trajectory kernels in combi-
nation with a Support Vector Machine (SVM) (Cortes & Vapnik,
1995; Schölkopf, Smola, Williamson, & Bartlett, 2000) to predict
the type of vessel from a trajectory. To do classification experi-
ments we use vessel trajectories in a 50 km radius around the Port
of Rotterdam. The AIS system provides a number of different vessel
types. For our classification experiments we use the four most
common types: cargo ship, tanker, tug and law-enforcement ves-

sels. We use a set of 400 trajectories, with 100 trajectories per vessel
type. The average sequence length for this set is 689 elements.

The trajectories for the four types are illustrated in Fig. 4. From
the figure we see that cargo ships and tankers have similar trajec-
tories, as do law enforcement vessels and tugs. Thus, separating
cargo ships from tankers and tugs from law enforcement will be
the hardest classification task. The vessel type is already available
via AIS. But, predicting the vessel type is still useful to check
whether the vessel type field is filled in correctly, and hence deter-
mines the trustworthiness of the vessel. Moreover in situations
where vessels are tracked by different means than AIS, such as ra-
dar, vessel type information might not be available.

For the classification algorithm we use the C-SVC implementa-
tion of the Support Vector Machine algorithm in LibSVM (Chang
et al., 2001). To evaluate the performance of the different kernels
we use a 10-fold cross validation set-up. For each fold the classifi-
cation accuracy is computed as the performance measure. Within
each fold the C parameter is optimized by using, again, 10-fold
cross validation.

Results. The presentation of the results below has the same
structure as used for the clustering experiment. With 4 equally
sized classes, the prior classification accuracy is 25%.

Table 3 presents the results for the different types of kernels.
The best score (76.25%) is achieved by the \(K_{\text{max,E}}\) kernel. For
all kernel types we see that compression has a positive effect on
classification, since the best scores are achieved under high com-
pression settings. We also must remark again that the soft-max
kernels suffer from numerical machine precision problems in the
uncompressed case.

Discussion. From the above results we see that compression has
a positive result on the performance in the classification task. How-
ever, the performance in case of very high compression (\(\epsilon = 10000\)),
which amounts to reduction to just a few points, is just slightly
worse than the best performance. However, there is still room
for almost 24% performance gain. When we inspect the confusion
matrix for the highest performance, we see that the most misclas-
sifications are made when separating tankers from cargo ships.
These two types have almost identical trajectories, and hence are
nearly impossible to distinguish on the basis of their trajectories
alone. Similar to the clustering experiment, we see that the soft-
max kernels are outperformed by their regular (max and max
normal) counterparts.

3.2.3. Outlier detection

For outlier detection we use trajectory kernels with one-class
SVMs (Schölkopf, Platt, Shawe-Taylor, Smola, & Williamson,
2001) to find outlying trajectories among a set of vessel trajectories
of cargo ships and tankers. In this set 39 trajectories were labeled
as outliers because they show strange behavior compared to the
747 trajectories that were labeled as regular. For this set, the aver-
age sequence length is 918 vectors. The outliers are plotted against
the regular trajectories in Fig. 5. From the figure we see that trajecto-
ries can be outliers for different reasons: vessels can sail to an
irregular place, have extra turns, or stop unexpected, etc.

We use the one class version of Support Vector Machines in Lib-
SVM (Chang et al., 2001) as outlier detector. To evaluate the perfor-
ance of the different kernels we randomly split the 747 normal
trajectories in the dataset into a training set containing \(\frac{3}{4}\) of
the dataset and a test set containing the other \(\frac{1}{4}\). To the test set we
add the 39 outlier trajectories. Per kernel we do this split 10 times.

Please cite this article in press as: de Vries, G. K. D., & van Someren, M. Machine learning for vessel trajectories using compression, alignments and domain
knowledge. Expert Systems with Applications (2012), http://dx.doi.org/10.1016/j.eswa.2012.05.060
For each split we train a one class SVM model on the training set, and then use the decision values generated by the model to compute as performance score the fraction of labeled outliers among these 39, i.e. the precision@39.

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>0 m</th>
<th>10 m</th>
<th>50 m</th>
<th>100 m</th>
<th>1000 m</th>
<th>10000 m</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{\text{max}}$</td>
<td>0.31</td>
<td>0.46</td>
<td>0.41</td>
<td>0.39</td>
<td>0.39</td>
<td>0.39</td>
</tr>
<tr>
<td>$K_{\text{maxnorm}}$</td>
<td>0.21</td>
<td>0.17</td>
<td>0.18</td>
<td>0.20</td>
<td>0.19</td>
<td>0.26</td>
</tr>
<tr>
<td>$K_{\text{softmax}}$</td>
<td>0.37</td>
<td>0.74</td>
<td>0.72</td>
<td>0.60</td>
<td>0.59</td>
<td>0.59</td>
</tr>
<tr>
<td>$K_{\text{soft-max}}$</td>
<td>0.41</td>
<td>0.47</td>
<td>0.44</td>
<td>0.45</td>
<td>0.54</td>
<td>0.54</td>
</tr>
<tr>
<td>$K_{\text{max}}$</td>
<td>0.38</td>
<td>0.45</td>
<td>0.43</td>
<td>0.40</td>
<td>0.39</td>
<td>0.39</td>
</tr>
<tr>
<td>$K_{\text{maxnorm}}$</td>
<td>0.36</td>
<td>0.59</td>
<td>0.55</td>
<td>0.47</td>
<td>0.55</td>
<td>0.55</td>
</tr>
<tr>
<td>$K_{\text{softmax}}$</td>
<td>0.36</td>
<td>0.59</td>
<td>0.55</td>
<td>0.47</td>
<td>0.55</td>
<td>0.55</td>
</tr>
</tbody>
</table>

**Table 4**

For each split we train a one class SVM model on the training set, using 10-fold cross validation to determine the optimal $\epsilon$. In this experimental set-up we assume that in an application setting we have cleaned the training set from outliers such that they have no influence on the model. Therefore we also do not include outliers in our training sets.

To test the performance of the different kernels in outlier detection we use the decision values generated by the one class SVM to rank the test trajectories. We select the top 39 ranked trajectory kernels and compute as performance score the fraction of labeled outliers among these 39, i.e. the precision@39.

**Results.** The mean precision@39 score for 10 random rankings of the test set is 0.16. Table 4 presents the results for the different kernels. The best performance is clearly achieved by the $K_{\text{softmax}}$ kernels, with the best score being 0.84. For the edit distance we see that the best performance is achieved by the soft-max kernels as well.

For all the types of kernels we see that most compression settings outperform the no compression setting. Again, as in the clustering and classification experiments, the soft-max kernels suffer from problems with numerical machine precision.

**Discussion.** As with the clustering and classification experiments, data reduction has a largely positive effect on outlier detection task performance. However, in the outlier detection task the soft-max kernels shine and outperform the max and max norm kernels, which are the best in the clustering and classification tasks, by a large margin.

Most outlying trajectories are outliers because part of the trajectory is strange. This strange part does not align well with normal trajectories. The soft-max kernels take into account all possible alignments, and in all these possible alignments this strange part is likely difficult to align well. Thus outlying trajectories will end up being very dissimilar from regular trajectories. This effect is the strongest in the case of the dynamic time warping soft-max kernel. For DTW badly aligned points can lead to high penalties, because there is no maximum gap penalty $g$.

**3.3. Concluding remarks**

The results of all three experiments show that applying compression, in the form of piecewise linear segmentation, to vessel trajectories before computing the alignment similarity scores does not have a negative impact on task performance. Especially in the case of outlier detection the performance is enhanced by a large margin. This is likely due to the fact that the amount of samples relative to the change in behavior of a vessel is large. Compression retains the salient points in a trajectory, i.e. the points where behavior changes. Due to geographical and legislative constraints, the behavior of vessels tends to change in the same locations, thus alignments are not negatively impacted by compression. Which alignment kernel works best depends on the task.

**4. Adding geographical domain knowledge**

Vessel trajectories exist in a space where places and locations have semantics of their own. Using this geographical domain knowledge can potentially enhance clustering, classification and outlier detection task performance. For example, two movements at different locations, with different direction and different trajectories may be very similar if it is known that both are mooring at a passenger terminal, anchoring in an anchoring area or approaching the same harbor via different sea lanes. Two similar trajectories may be considered more different if we know that one is fishing and another is just passing through.

The alignment measures of the previous section are easily adapted to deal with geographical domain knowledge. We will introduce a new substitution function to deal with the domain knowledge. For example, two trajectories that end at two different anchoring areas may now have a small distance because the type of location is now the same.

**4.1. Enriching trajectories**

Our geographical domain knowledge comes in the form of two simple ontologies. Both ontologies are stored as RDF. One ontology
contains the definition of different anchorages, clearways and other areas at sea, which we call AnchoragesAndClearways. All of these geographical features were converted to RDF from shape files from Rijkswaterstaat (RWS), part of the Netherlands Ministry of Transport, Public Works and Water Management. The other ontology has definitions for different types of harbors, such as liquid bulk and general cargo (containers), which we call Harbors. All harbors were manually copied from the harbor branches map of the Port of Rotterdam Authority. The concepts in these ontologies have a unique identifier, are assigned polygon regions, and have a type. The different concepts in these ontologies are illustrated in Fig. 6.

The modeling of the concepts follows the GeoNames ontology. The polygons that define the different regions can be overlapping. For example, an anchorage area can overlap with a harbor approach. Each of the harbor regions is assigned a polygon demarcating the land area of the harbor (the port) and not the part of the water (the dock), because the same dock can be shared by two ports of different types. For instance, there can be container cranes on one side of the basin and oil valves on the other. This is not the case for harbors found in GeoNames, because these are located by points in the middle of the dock.

We have created two web services to enrich trajectories with geographical features. One of these services, NearestHarbor, matches a latitude, longitude point to the nearest harbor in Harbors within a predetermined range. The label and most specific type of this harbor is then returned, e.g. ‘DryBulk4’ and ‘DryBulkHarbor’.

Similarly, the other service, Intersection, returns a set of (label, most specific type) pairs corresponding to the regions in AnchoragesAndClearways that intersect with a given point. Both web services were implemented in SWI-Prolog using the Space package (van Hage, Wielemaker, & Schreiber, 2010).

Using the Intersection service we create a sequence of sets of geo-labels for a trajectory as: \( T_{lab} = L_1, ..., L_l \), where a set of geo-labels \( L_i = \{ \text{name}_1, \text{type}_1 \}, ..., \{ \text{name}_n, \text{type}_n \} \). Each \( L_i \) is a set of pairs where name is the label of the region such that \( T(i) \) is contained in the polygon that defines that region and \text{type} is the type of that region. Let \( T_{lab}(i) = L_i \). Note that \( L_i \) is a set because a point can be in multiple regions. Furthermore, \( L_i \) can be empty since the defined regions do not cover everything. Next to a trajectory as a sequence of sets of geo-labels \( T_{lab} \) as a separate object, we also define the combination of the sequence of points \( T \) and \( T_{lab} \) as \( T_{traj,lab} = (T(1), T_{lab}(1)), ..., (T(n), T_{lab}(n)) \). Let \( T_{traj,lab}(i) = (T(i), T_{lab}(i)) \).

We treat the start \( T(1) \) and end \( T(n) \) of a trajectory with special interest and define special start and end objects for these. For a trajectory \( T \) we define the start and end object \( T_{start} \) and \( T_{end} \) as \( T_{start} = (\text{stopped}, T_{start}) \) and \( T_{end} = (\text{stopped}, T_{end}) \). stopped is a Boolean value indicating whether the vessel is stopped. \( L_{start} \), respectively \( L_{end} \), is a set of pairs \( (\text{name}, \text{type}) \). To add domain knowledge about whether a vessel is docked at a port we use the NearestHarbor service to find the geographically closest harbor \( (\text{name}, \text{type}) \) to the point \( T(1) \), respectively \( T(n) \). If this service returns a harbor and the vessel is also stopped, then we put this pair in \( L_{start} \), respectively \( L_{end} \). If the vessel is stopped but there is no harbor nearby, then we use the Intersection service to find the regions that \( T(1) \), respectively \( T(n) \), is in, and add those to \( L_{start} \), respectively \( L_{end} \). We do the same if the vessel is not stopped. Thus, we are interested in harbors if a vessel is docked there, where docked is defined as being close to that harbor and stopped. We could have also added the start and stop harbors to the sequence of geo-labels \( T_{lab} \). However, by treating these separately we have more flexibility in weighing the importance of the start and stop places.

So, for each trajectory we have five objects, the trajectory itself, \( T \), a sequence of sets of geo-labels, \( T_{lab} \), the combination of \( T \) and \( T_{lab} \) into \( T_{traj,lab} \), and start and end information, \( T_{start} \) and \( T_{end} \).

The above labeling process has similarities with the work by Alvares, Kuipers, de Macêdo, Moelans, and Vaisman (2007). However, we label not only the stops (or starts and ends in our terminology), but also the moves. Furthermore we use RDF based web services instead of a geographic database.

4.2. Enriched Trajectory Similarity Kernels

Like trajectories \( T \), sequences of sets of geo-labels \( T_{lab} \) and their combinations \( T_{traj,lab} \) can be compared using the same alignment methods as those used in Section 3. However, they require different substitution functions, which we will define below. After these definitions we look into how we create kernels for the different objects defined above.

For sequences of sets of geo-labels \( T_{lab} \), the substitution function \( \text{sub}_{lab} \) expresses how many labels two sets of labels \( L_i \) and \( L_j \) have in common. For two sets of labels \( L_i, L_j \),

\[
\text{sub}_{lab}(L_i, L_j) = -1 + \frac{|N_i \cap N_j| + |M_i \cap M_j|}{\sqrt{|N_i| + |M_i|}|N_j| + |M_j|},
\]

where \( N_m = \{ l \mid l \in L_m \} \) and \( M_m = \{ l \mid l \in L_m \} \); for \( m = i, j \). Note that \( N_1 \) indicates the first element of the \( (\text{name}, \text{type}) \) pair, i.e. the name, and \( l_1 \) the second element, i.e. the type. Thus, we count the number of names and type labels that both sets have in common and divide this by the square root of the multiplied lengths of the sets.
We can combine the substitution functions for position, \( \text{sub}_1 \) and \( \text{sub}_2 \), with the above substitution function for sets of labels such that we can do alignments on the combined trajectories \( T_{\text{traj}} \) and \( T_{\text{tra}} \). Let \( T_{\text{traj}} \) and \( T_{\text{tra}} \) be two combinations of trajectories and sequences of sets of geo-labels. Furthermore, \((S_i, L_i) = S_{\text{traj}}(i) \) and \((T_j, L_j) = T_{\text{tra}}(j) \). Then
\[
\text{sub}_{m,n}^{w_{1},w_{2}}((S_i, L_i), (T_j, L_j)) = w_{1}\gamma \text{sub}_{m}(S_i, T_j) + w_{2}\text{sub}_{lab}(L_i, L_j),
\]
where \( m = 1, 2 \). This function combines the substitution functions for trajectories and for sequences of set of labels into one function, where the index \( m \) determines if \( \text{sub}_1 \) or \( \text{sub}_2 \) is used. The weights \( w_1 \) and \( w_2 \) are used to determine the influence of the position and domain knowledge information. We furthermore use the parameter \( \gamma \) to get the position substitution function and the label substitution function on the same scale. This can also be done directly by incorporating \( \gamma \) into \( w_1 \). However, this would result in the weights \( w_1 \) and \( w_2 \) not summing to 1, which we consider to be less clear. We experiment with \( w_1 \) and \( w_2 \) and fix \( \gamma \) to an appropriate value. In the case of the \( \text{sub}_{m,n}^{w_{1},w_{2}} \) functions, the gap penalty \( g \) is also \(-1\), since this is the minimum value that these substitution functions are designed to take, which is achieved by tuning the \( \gamma \) parameter.

We have not discussed similarity or substitution functions for start and end objects yet. This similarity is straightforward; it can immediately be put into kernel form. For all \((\text{stopped}_1, L_1) \) and \((\text{stopped}_2, L_2) \) in a set of start/end objects, we compute a kernel matrix as:
\[
K(i, j) = 1 + [\text{stopped}_1 = \text{stopped}_2] + \text{sub}_{lab}(L_i, L_j).
\]
Thus the similarity between two start/end objects is determined by whether the vessel is stopped or not and how much labels there are in common. Using this definition we get a kernel \( K_{\text{start}} \) for the start objects and a kernel \( K_{\text{end}} \) for the end objects.

In Section 3 we defined different alignment kernels. Essentially, we plug the substitution functions defined above in the definitions of Section 3 to get kernels for the sequence of sets of geo-labels \( T_{\text{lab}} \) and the combined version \( T_{\text{traj}} \). An example of such a kernel is \( K_{\text{lab, maxnorm,ED}} \), which is the kernel that uses the regular non-normalized dynamic time warping measure applied to the sets of labels sequence. In this section, we give the subscript \( \text{traj} \) to kernels for normal trajectories \( T \).

We combine the kernels defined above together by taking weighted sums in two ways. Let \( K_{\text{traj}} \) be an alignment kernel for trajectories, \( K_{\text{lab}} \) an alignment kernel for sequences of geo-labels, and \( K_{\text{start}} \) and \( K_{\text{end}} \) kernels for start and end objects, then
\[
K_{\text{all}1} = w_1K_{\text{traj}} + w_2K_{\text{lab}} + w_3K_{\text{start}} + w_4K_{\text{end}},
\]
with \( w_1 + w_2 + w_3 + w_4 = 1 \). The other kernel uses the alignment on the trajectories combined with sequences of geo-labels \( T_{\text{tra}} \). Let \( K_{T_{\text{traj}} \text{lab}} \) be an alignment kernel for trajectories combined with the their sequence of sets of geo-labels, and \( K_{\text{start}} \) and \( K_{\text{end}} \) kernels for start and end objects, then
\[
K_{\text{all}2} = w_1K_{\text{traj}} + w_2K_{T_{\text{traj}} \text{lab}} + w_3K_{\text{start}} + w_4K_{\text{end}},
\]
with \( w_1 + w_2 + w_3 + w_4 + w_5 = 1 \). Depending on the alignment kernels that are used for \( T_{\text{tra}}, T_{\text{lab}} \) and \( T_{\text{traj}}, T_{\text{lab}} \) these kernels are positive semi-definite (PSD). The weighted kernels are inspired by work in computational biology on combined kernels for comparing protein sequences and DNA (Cuturi, 2010).

The \( K_{\text{start}} \) and \( K_{\text{end}} \) sub-kernels are cheap to compute. For \( K_{\text{traj}} \) and \( K_{\text{lab}} \) we use the dynamic programming approach, mentioned in Section 3.

4.3. Experiments

In Section 3 we saw that for none of the tasks the best performance was achieved in the no compression setting. Therefore, we will not investigate this no compression setting in the experiments here. Furthermore, the number of repeated labels for each sequence of sets of geo-labels \( T_{\text{lab}} \) would be very large in this setting, which seems useless. All trajectories are compressed using \( \text{PS}_\epsilon \), with \( \epsilon = 50 \) m, which is the parameter setting that achieves the highest performance in the clustering and outlier detection task. Furthermore, visual inspection of some of the labeled trajectories suggests that no regions are missed under this setting in the labeling process.

Contrary to the previous section, the datasets used are from the same area. In this area the ‘AnchorageAndClearways’ ontology contains the names and polygons for approximately 50 regions of 6 different types. There are around 90 different harbors in ‘Harbors’ that are divided into 7 different types. For each trajectory in the respective dataset we created a sequence of sets of geo-labels \( T_{\text{lab}} \), the combination with the raw trajectory \( T_{\text{traj}} \) a start object \( T_{\text{start}} \) and an end object \( T_{\text{end}} \). The threshold used in the ‘NearestHarbor’ service is set to 100 m. This threshold range was determined manually and is suitable given the size of the vessels, docks and clearways, and is of an order of magnitude larger than the GPS errors.

4.3.1. Clustering

The gold standard clustering dataset that we used in the previous section is not well suited to evaluate the performance of clustering using the domain knowledge kernels that we defined in this section, since the gold standard is located in an area for which not a lot of geographical domain knowledge exists. We use another dataset of 1917 trajectories here, which contains data from the same area as the classification (Section 2.1) dataset. The difference with the classification set is that all vessel types are included. Since we do not have a gold standard, we cannot evaluate the clustering results quantitatively. Our evaluation is qualitative in nature, and consists of showing illustrative clustering examples for three different settings of the weights in \( K_{\text{all}} \). After some experimentation, we chose as the \( K_{\text{all}} \) kernel the edit distance variant: \( K_{\text{lab, maxnorm,ED}} \), with \( g = -0.01 \), and as \( K_{\text{lab}} \) kernel we took \( K_{\text{lab, maxnorm,ED}} \). The first setting for \( K_{\text{all}} \) is \( w_1 = \frac{1}{3}, w_2, w_3 = \frac{1}{3} \), the resulting kernel being \( K_{\text{comb}} \). This setting weights the trajectory information and the ontological information equally. There is also a setting for just the trajectory information, \( w_1 = 1, w_2, w_3, w_4 = 0, K_{\text{tra}} \), and one for just the ontological information \( w_1 = 0, w_2, w_3, w_4 = \frac{1}{3}, K_{\text{lab}} \). We define these three distinct settings to investigate the effect of using low-level trajectory information and domain knowledge in clustering.

These kernels are used as input for the weighted kernel k-means algorithm (Dhillon et al., 2007), analog to the previous section. Because we have no gold standard clustering that we wish to achieve or a specific criterion that we want to optimize, it is difficult to determine a good value for the number of clusters parameter \( k \). Therefore, we manually experimented with different values and finally selected \( k = 40 \), which gave cluster examples that show the differences between the three kernels well.

Examples. The two examples we give below illustrate behavior clusters of vessels that arise in the combined setting, i.e. using kernel \( K_{\text{comb}} \). For each example we will also show the clusters from the other two settings \( (K_{\text{tra}} \) and \( K_{\text{lab}} \)) that resemble the behavior the most. All figures show the trajectories in one cluster in black against a background of all trajectories in gray. For the trajectories in a cluster, the start of a trajectory is indicated by a dot and the end by an asterisk.

Fig. 7 illustrates the behavior of vessels anchoring in a specified anchoring area. In Fig. 7 a we show a cluster resulting from the
combined information kernel $K_{\text{comb}}$. We see that all the tracks end up in one anchoring area. If we use only the trajectory information, i.e. $K_{\text{traj}}$, we get the result in Fig. 7B. In this case there are a number of other trajectories that do not end in the anchoring area. For the clustering with only the ontological information, $K_{\text{onto}}$, we see something different (Fig. 7C). Here there is another track of a vessel anchoring in another anchoring area. So, the combination of trajectory and ontological information results in the discovery of the behavior “anchoring in a specific anchoring area”.

The trajectories in Fig. 8A, on which we zoom in in Fig. 8B, are a result of clustering with the combined kernel. The figures show trajectories that do not stop and continue on the river to the land behind. These vessels are smaller, and in Fig. 8B we see that none of them pass through the deep water lane. Fig. 8C is a cluster from clustering with the trajectory only kernel $K_{\text{traj}}$. The trajectories in this cluster both stop and go on, and some of them go through the deep water lane and some of them do not. Because no domain knowledge is used the deep water lane and non-deep water lane trajectories are difficult to separate, since they do not differ much in shape. The comparable cluster for the $K_{\text{onto}}$ kernel, Fig. 8D, shows trajectories that go in different directions and some noise. Using only domain knowledge does not guarantee that trajectories that go in different directions are not clustered. The combined kernel discovers the behavior of “smaller ships coming from sea and continuing directly to the land behind”.

The above examples show that a combination of low-level trajectory information and geographical domain knowledge in one similarity measure can lead to the discovery of interesting vessel behavior patterns that are indeed due to a combination of these two information sources.

4.3.2. Classification

In the classification task we use the dataset described in Section 3. We use a larger set than in the previous section, it contains 800 trajectories, with 200 trajectories per vessel type. Due to the fact that we do not have a ‘no compression’ setting, we can easily run the experiments with more data. The classification algorithm set-up is the same as in the classification experiment of the previous section. We use the C-SVC Support Vector Machines from LibSVM, with a 10-fold cross validation set-up to evaluate the kernel performance, with inner 10-fold cross validation to optimize the C parameter.
The first stage of the experiments is to discover the best performance settings for each of the basic kernels: $K_{lab}$ and $K_{traj,lab}$. For $K_{traj}$ we already know these settings from the previous section. The kernels $K_{start}$ and $K_{end}$ do not have any settings.

For $K_{lab}$ we test the different variants of dynamic time warping and edit distance. The soft-max kernels parameter $\beta$ is varied over: $\frac{1}{10}, \frac{3}{10}, \frac{4}{10}, \frac{5}{10}, 1$. Besides the parameters of the different variants of DTW and edit distance, the $K_{traj,lab}$ kernel has the parameters $w_1$ and $w_2$, for which we test the combinations: $(0.9,0.1), (0.7,0.3), (0.5,0.5), (0.3,0.7)$ and $(0.1,0.9)$. The $\gamma$ parameter is set to 100, which is the ratio between the best gap penalty ($-0.01$) for the $K_{traj}$ kernel and the gap penalty ($-1$) in the $K_{lab}$ kernel. Furthermore, for the soft-max alignments we vary $\beta$ over: $\frac{1}{10}, \frac{3}{10}, \frac{4}{10}, \frac{5}{10}, 1$.

With the best performing basic kernels we create variants of $K_{lab}$ and $K_{all}$ with different weight settings, to cover different combinations of the basic kernels. We give the specific settings in the Results section.

Results. Table 5 presents the results for the different basic kernels. If a kernel has parameter settings we give the mean, maximum and minimum accuracies. For reference we give the performance of the best $K_{traj}$ kernels.

The performance of the $K_{start}$ and $K_{end}$ kernels is almost the same. The $K_{lab}$ kernels are relatively close together in performance, with the DTW max and normalized max performing the worst. The best average performance is for the $K_{lab,maxnorm,ED}$ kernel. In case of the $K_{traj,lab}$ kernels, the best performance is clearly for the $K_{traj,lab,maxnorm,ED}$ kernel.

Based on Table 5 we take the kernel $K_{lab,maxnorm,ED}$ for the sequences of geo-labels. We combine this kernel with $K_{traj,lab,maxnorm,ED}$ with $\beta = -0.01$ and the start and end object kernels to create a kernel incorporating domain knowledge $K_{all}$. The results for different weight settings are shown in Table 6. To statistically compare accuracies for two kernels we use a two-tailed paired t-test with $p < 0.05$. For each score we indicate whether there is a significant difference with the accuracy achieved by the $K_{traj}$ kernel for that $\epsilon$ setting. A * indicates a significant positive difference, whereas a - indicates a significant negative difference. We see that for both $\epsilon$ settings there are weight settings that significantly outperform the $K_{traj}$ kernel.

From Table 5 we see that $K_{traj,lab,maxnorm,ED}$ has the best performance of the $K_{traj,lab}$ kernels. Thus we take this kernel with its best settings and combine it with the start and end object kernels to get $K_{all}$. Results are shown in Table 7. The combined kernels show better performance than the $K_{traj}$ kernel. However, the difference is only significant for one setting.

Table 5

<table>
<thead>
<tr>
<th>$\epsilon = 50$ in</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{traj,maxnorm,ED}$</td>
<td>73.50</td>
</tr>
<tr>
<td>$K_{lab}$</td>
<td>63.63</td>
</tr>
<tr>
<td>$K_{end}$</td>
<td>62.88</td>
</tr>
<tr>
<td>$K_{lab,maxDTW}$</td>
<td>45.38</td>
</tr>
<tr>
<td>$K_{lab,maxnorm,DTW}$</td>
<td>50.13</td>
</tr>
<tr>
<td>$K_{lab,softmax,DTW}$</td>
<td>58.67</td>
</tr>
<tr>
<td>$K_{lab,ED}$</td>
<td>52.75</td>
</tr>
<tr>
<td>$K_{lab,maxnorm,ED}$</td>
<td>58.75</td>
</tr>
<tr>
<td>$K_{lab,softmax,ED}$</td>
<td>58.52</td>
</tr>
<tr>
<td>$K_{traj,lab,maxDTW}$</td>
<td>46.89</td>
</tr>
<tr>
<td>$K_{traj,lab,maxnorm,DTW}$</td>
<td>54.45</td>
</tr>
<tr>
<td>$K_{traj,lab,softmax,DTW}$</td>
<td>62.14</td>
</tr>
<tr>
<td>$K_{traj,lab,ED}$</td>
<td>71.30</td>
</tr>
<tr>
<td>$K_{traj,lab,maxnorm,ED}$</td>
<td>72.10</td>
</tr>
<tr>
<td>$K_{traj,lab,softmax,ED}$</td>
<td>61.64</td>
</tr>
</tbody>
</table>

Table 6

<table>
<thead>
<tr>
<th>$\epsilon = 50$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>$w_2$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7

<table>
<thead>
<tr>
<th>$\epsilon = 50$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>$w_4$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Discussion. In the results above we see that kernels combining geographical domain knowledge and regular trajectory information can achieve significantly better classification accuracy than kernels that only consider the regular trajectory information. The highest performance increase is somewhat upward of 3%. We also see that kernels that only consider domain knowledge are outperformed by kernels that only consider trajectory information.

The start and end of a trajectory already contain a lot of information about the type of vessel. The kernel that just takes this type of information into account ($W_1$, $W_4 = \frac{1}{2}$) achieves a score of 71.0%. We also see that the best performing kernel type for classification with only trajectory information, the regular normalized max edit distance kernels, also performs well for the trajectories combined with domain knowledge $T_{traj,lab}$ and the sequences of geo-labels $T_{lab}$. Thus, for the three types of data used here, edit distance seems to be a good choice for classification.

4.3.3. Outlier detection

For our outlier detection experiment we use the dataset described in Section 3. Like we did in the classification task, the first stage of the experiment is to discover the best performing basic kernels. We already know from Section 3 that the best performing $K_{traj}$ kernels are the DTW soft-max variants. For the other kernels that have parameters, we use the same settings as in the classification experiment. Using the best performing kernels we create variants of $K_{lab}$ and $K_{all}$, covering different combinations of the basic kernels and hence different combinations of domain knowledge.

The algorithmic set-up for the outlier detection experiment is the same as for the experiments in Section 3. We use a one-class Support Vector Machine. The 747 normal trajectories in the dataset are randomly split into a training set of $\frac{1}{3}$ and a test set of the rest, to which we add the outlying trajectories. This split is done 10 times.
times per kernel, and for each split the one-class SVM is optimized on the train set using 10-fold cross validation.

Results. In Table 8 we see the results for the different basic kernels. The best performance is achieved by the $K_{\text{traj,softmax,DTW}}$ kernel.

For $K_{\text{all}}$, we take from Table 8 the $K_{\text{lab,softmax,DTW}}$ kernel as basic kernel, with $\beta = 1$. We combine this with the $K_{\text{traj,softmax,DTW}}$ kernel for regular trajectories, and the start and end object kernels.

The results for different weight settings for $K_{\text{all}}$ are shown in Table 9. We statistically compare the results for two kernels using a two-tailed Student t-test with $p < 0.05$. For each score a significant difference with the $K_{\text{traj,softmax,DTW}}$ kernel is indicated with $a^+$ if it is positive and $a^-$ if it is negative. All the combined kernels show a significantly worse performance than the $w_1 = 1$ kernel, i.e. $K_{\text{traj,softmax,DTW}}$.

$K_{\text{traj,softmax,DTW}}$ has the best performance of the $K_{\text{lab}}$ kernels. Thus we use this to create a kernel $K_{\text{all}}$. We give the results in Table 10. Again all combinations are significantly outperformed by the $K_{\text{traj,softmax,DTW}}$ kernel, as indicated by $a^+$.

Discussion. The only combination of trajectory information and geographical domain knowledge that performs similarly to the trajectory information only kernel $K_{\text{traj,softmax,DTW}}$ is the kernel $K_{\text{traj,softmax,DTW}}$, since performance under their best settings shows no significant difference. The incorporation of the start and end information has a clear negative effect on outlier detection performance.

Like in Section 3, we see again that soft-max kernels are well suited for outlier detection, since the soft-max variants that include domain knowledge outperform the non-soft-max versions.

4.4. Concluding remarks

Adding geographical domain knowledge leads to better clustering results and also improves performance in the classification task. The outlier detection task does not benefit from adding domain knowledge. The domain knowledge can easily be added by using different substitution functions for the same alignment kernels.

5. Conclusions and future work

We presented a solution for performing the three typical machine learning tasks of clustering, classification and outlier detection on vessel trajectory data. Our approach consists of three steps. In the first step we compress vessel trajectories using an adapted piecewise linear segmentation method. This method reduces the trajectories substantially but retains the important stop and move information. To the compressed trajectory data we apply alignment based similarity measures. The computation of these alignments benefits greatly from the previous compression step, making computations a factor 100 faster. Moreover, this compression does not negatively impact task performance. Finally, the alignment measures allow for easy integration of geographical domain knowledge in the form of types and labels of places and regions. This added domain knowledge improves the performance in clustering and classification.

The results of the outlier detection experiments illustrate the good combination of piecewise linear segmentation and alignment based similarities very well. When the dynamic time warping soft-max alignment kernel is applied to the compressed trajectories, substantially better performance is achieved than without compression. The outlying parts of the trajectories are retained by the compression algorithm, and these parts do not align well with other trajectories, to which the soft-max DTW measure seems especially sensitive.

Lee, Han, and Whang (2007), Lee, Han, Li, and Gonzalez, (2008), and Lee, Han, and Li (2008) present a partition-and-group data mining framework for trajectories that covers the same tasks as this paper. However, their approach focuses on solving clustering, classification and outlier detection for subtrajectories. We use generic machine learning algorithms for the three tasks, putting all main knowledge in the form of types and labels of places and regions. This added domain knowledge improves the performance in clustering and classification.

The results of the outlier detection experiments illustrate the good combination of piecewise linear segmentation and alignment based similarities very well. When the dynamic time warping soft-max alignment kernel is applied to the compressed trajectories, substantially better performance is achieved than without compression. The outlying parts of the trajectories are retained by the compression algorithm, and these parts do not align well with other trajectories, to which the soft-max DTW measure seems especially sensitive.

Lee, Han, and Whang (2007), Lee, Han, Li, and Gonzalez, (2008), and Lee, Han, and Li (2008) present a partition-and-group data mining framework for trajectories that covers the same tasks as this paper. However, their approach focuses on solving clustering, classification and outlier detection for subtrajectories. We use generic machine learning algorithms for the three tasks, putting all main knowledge about the application domain in the similarity measure. The Lee et al. framework integrates knowledge about the moving object domain more directly in their methods. These two issues complicate a direct comparison. First, our datasets are not yet ideally suited for a subtrajectory focused approach, but this can likely be adapted. Second, because our framework differs a lot from the framework by Lee et al. there are a lot of parameters to consider.

### Table 10
Mean precision@39 for different $K_{\text{all}}$ kernels. The gray background is used to group kernels of the same type visually.

<table>
<thead>
<tr>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
<th>$w_4$</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.82</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.67</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.08</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.15</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>0.24</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0.33</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.69</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>0.71</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.75</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0.37</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0.38</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0.42</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0.33</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0.39</td>
</tr>
</tbody>
</table>

Please cite this article in press as: de Vries, G. K. D., & van Someren, M. Machine learning for vessel trajectories using compression, alignments and domain knowledge. Expert Systems with Applications (2012), http://dx.doi.org/10.1016/j.eswa.2012.05.060
to make a good comparison, i.e. it is not just a matter of plugging in another similarity measure.

Using the proposed methods in an online system, such as a Maritime Safety and Security (MSS) system, requires existing solutions to make Support Vector Machines faster during classification time, such as work by Burgschürk and Schölkopf (1997), since this is an expensive step. Furthermore, methods to compute partial alignments could prove to be beneficial.

Acknowledgements

The authors thank Willem Robert van Hage, Véronique Malaisé and Pieter Adriaans for valuable comments and collaboration. This work has been carried out as a part of the Poseidon project in collaboration with Thales Nederland, under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

References


van Kreveld, M., & Luo, J. The definition and computation of trajectory and subtrajectory similarity. In Sanet et al. Samet and Shahabi, 2007 (p. 44).


van Kreveld, M., & Luo, J. The definition and computation of trajectory and subtrajectory similarity. In Sanet et al. Samet and Shahabi, 2007 (p. 44).

