Kernel methods for vessel trajectories

de Vries, G.K.D.

Citation for published version (APA):
This chapter is an extension of the work described by de Vries et al. (2010).

In this chapter we investigate similarity measures that combine low-level trajectory information with geographical domain knowledge to compare vessel trajectories. These similarity measures are largely based on the alignment techniques of Chapter 4. In a clustering experiment we show how these measures can be used to discover behavior concepts in vessel trajectories that are dependent both on the low-level trajectories and the domain knowledge. We also apply these measures in a classification and outlier detection task. In the classification experiment we show that significantly better classification accuracy can be achieved by combining trajectory information and geographical domain knowledge. The outlier detection experiment shows no significant increase in performance due to the added domain knowledge.

6.1 INTRODUCTION

The vessel trajectories that we are concerned with in this thesis, as well as other types of moving object trajectories, exist in spaces where places and regions have semantics of their own. In the case of vessels there are concepts such as anchoring areas, sea lanes and harbors. If we add this information to the trajectories and incorporate it in the similarity measure, then we can potentially discover more complex and interesting behavior patterns.

A simple example of such a pattern is the trajectories of large cargo ships and tankers that use a shipping lane to sail north. Using the shipping lane information the trajectories of these vessels can be discriminated from other vessels moving in that direction, such as the fishing vessels that do not use a shipping lane to go fishing up north. Also different types of ports, e.g. petrol docks and cargo terminals, can help us discriminate between the trajectories of tankers and those of cargo vessels, which look really similar in terms of movement.

In this chapter we present alignment based similarity measures that combine low-level vessel trajectories with geographical domain knowledge, such as the name and type of the regions that vessels pass through and where they stop. As a basis for this we use the alignment kernels from Chapter 4, because they outperform the measures in Chapter 5 and their integration of domain knowledge is also more intuitive. We use the similarity measures in a clustering, classification and outlier detection task. The clustering experiment shows
a number of interesting vessel behaviors that are discovered based on the combination of trajectories and geographical domain knowledge. In the classification task we try to predict the type of the vessel. The combination of trajectories and domain knowledge gives the best average classification accuracy. Integration of domain knowledge can lead to a similar performance in outlier detection as using only trajectory data, but it does not significantly increase performance.

Our work has similarities with the work by Andrienko et al. (2007), where the authors use a series of clusterings with different similarity measures to discover interesting movement patterns. In this chapter we incorporate more domain knowledge and also provide the combination of trajectory and domain knowledge as one measure where weights can be used to determine the influence of each component.

The rest of this chapter is organized as follows. In Section 6.2 we will describe the geographical domain knowledge that we used and how we enriched trajectories with this knowledge. Our similarity measure for low-level trajectories labeled with geographical information is described in Section 6.3. The clustering, classification and outlier detection experiments are presented in Section 6.4. We end with some conclusions and suggestions for future work.

6.2 GEOGRAPHICAL DOMAIN KNOWLEDGE

In this section we introduce the geographical domain knowledge that we use and how we enrich vessel trajectories with this knowledge.

6.2.1 Domain Ontologies

Our geographical domain knowledge comes in the form of two simple ontologies. Both ontologies are stored as RDF; see Section 3.4 for more details. One ontology contains the definition of different anchorages, clearways and other areas at sea, which we call Anchorages-AndClearways. 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.\(^1\) 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 Figure 12.

The modeling of the concepts follows the GeoNames ontology, with the exception of the positioning properties (wgs84:{lat|long}) and the type property (geo:featureCode). GeoNames specifies feature types with

\(^1\) [http://www.portofrotterdam.com/en/Port/port-maps/Pages/branches.aspx](http://www.portofrotterdam.com/en/Port/port-maps/Pages/branches.aspx)
Figure 12: Visualization of the geographical domain knowledge in KML. All the clearways and approach areas (translucent, A), anchorages (dark red, B), restriction zones (dark blue, C), and separation zones (yellow, D) are shown in the picture on the left. The various harbor types and the deep water lane (light blue, E) for large vessels are shown in the picture on the right, which corresponds to the rectangular area outlined with a white line in the picture on the left.
the geo:featureClass property for general classes, like geo:P for populated feature types and geo:H for hydrographical feature types. Specific types, like geo:H.HBR for harbors, are specified with the geo:featureCode property. For our experiments we require more specific types than just harbor, e.g. dry bulk harbor. We assigned these specific types as extra types to the features. The specific types are modeled as subclasses of the original geo:featureCodes. To allow RDFS reasoning over the featureCodes and their new subclasses we temporarily asserted that geo:featureCode is a subproperty of rdf:type, which makes each featureCode an rdfs:Class containing all the features of that type as instances. GeoNames uses WGS84 latitude longitude coordinates, while we use polygons of WGS84 coordinates which we express in the GeoRSS Simple vocabulary2.

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. An example of the representation of a harbor with a specific type and polygon shape can be found in Figure 13.

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 actual range used in our experiments will be discussed in Section 6.4. The label and most specific type of this harbor is then returned, e.g. ‘place_DryBulk4’ and ‘ptype_DryBulkHarbor’. Similarly, the other service, Intersection, returns a set of (label, most

---

2 See http://www.georss.org/simple and http://www.georss.org/rdf_rss1
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 et al. 2010)*.

We represent the geographical domain knowledge in RDF using the GeoNames and GeoRSS ontologies. Besides these ontologies, the SWI-Prolog Space package also supports other geolocation representations in RDF, like those used by DBpedia,3 Freebase4 and LinkedGeoData (OpenStreetMap).5 Therefore the services that match the trajectories to geographical features could just as well use RDF from these sources by directly loading them over the web. The main reason to use RWS sea maps and manually converted Port of Rotterdam harbor types is that currently there are hardly any maritime polygons to be found on the web. Moreover, the maritime features that do exist are not of the suitable level of abstraction. In the case of GeoNames, the lowest level of abstraction is often not low enough, as discussed before, while in LinkedGeoData the existing levels of abstractions are too low or inappropriate. For example, harbors are categorized as leisure areas for angling, and each separate trash bin is listed as such.

### 6.2.2 Enriching Trajectories

In this chapter we consider trajectories \( T \) as sequences of points, as defined in Definition 6.2.1. The trajectories are taken from the datasets described in Section 2.3.1. These trajectories have already been cut at the stops and are therefore all moves. They are delimited by the vessel entering the area of observation or starting (from being stopped) and the vessel leaving the area of observation or stopping.

Using the *Intersection* service we create a sequence of sets of geo-labels \( T_{\text{lab}} \) as defined in Definition 6.2.1.

**Definition 6.2.1.** A sequence of sets of geo-labels \( T_{\text{lab}} \) for a trajectory \( T \) is defined as: \( T_{\text{lab}} = L_1, \ldots, L_{|T|} \), where a set of geo-labels \( L_i = \{ (\text{name}_1, \text{type}_1), \ldots, (\text{name}_m, \text{type}_m) \} \). Each \( L_i \) is a set of pairs where \( \text{name}_j \) is the label of the region such that \( T(i) \) is contained in the polygon that defines that region and \( \text{type}_j \) is the type of that region. Let \( T_{\text{lab}}(i) = L_i \).

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 \( T_{\text{lab}} \) as a separate object, we also define the combination of the sequence of points \( T \) and \( T_{\text{lab}} \) as \( T_{\text{traj,lab}} \) in Definition 6.2.2.

---

3 http://dbpedia.org
4 http://freebase.com
5 http://linkedgeodata.org
**Definition 6.2.2.** The combination of a trajectory $T$ and its corresponding sequence of sets of geo-labels $T_{lab}$ is defined as $T_{traj,lab} = (T(1), T_{lab}(1)), \ldots, (T(n), T_{lab}(n))$. Let $T_{traj,lab}(i) = (T(i), T_{lab}(i))$.

We treat the start $T(1)$ and end $T(|T|)$ of a trajectory with special interest and define special start and end objects for these in Definition 6.2.3.

**Definition 6.2.3.** For a trajectory $T$ we define the start and end object $T_{start}$ and $T_{end}$ as $T_{start} = (\text{stopped}, L_{start})$ and $T_{end} = (\text{stopped}, L_{end})$. stopped is a boolean value indicating whether the vessel is stopped. $L_{start}$, respectively $L_{end}$, is a set of pairs (name, 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 (name, type) to the point $T(1)$, respectively $T(|T|)$. 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 close, then we use the Intersection service to find the regions that $T(1)$, respectively $T(|T|)$, 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 et al. (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.

### 6.3 Enriched Trajectory 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 Chapter 4. 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. An overview of the kernels defined in this chapter is given in Table 14.
Table 14: List of kernels defined in this chapter.

<table>
<thead>
<tr>
<th>Shorthand</th>
<th>Parameters</th>
<th>Description</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{\text{start}}$</td>
<td>none</td>
<td>Kernel for start of trajectory domain knowledge</td>
<td>$6.3.3$</td>
</tr>
<tr>
<td>$K_{\text{end}}$</td>
<td>none</td>
<td>Kernel for end of trajectory domain knowledge</td>
<td>$6.3.3$</td>
</tr>
<tr>
<td>$K_{\text{lab}}$</td>
<td>$\beta, g$</td>
<td>Kernels for sequences of sets of geo-labels</td>
<td>$6.3.4$</td>
</tr>
<tr>
<td>$K_{\text{traj_lab}}$</td>
<td>$\beta, g, w_1, w_2$</td>
<td>Kernels for trajectories combined with sets of geo-labels</td>
<td>$6.3.5$</td>
</tr>
<tr>
<td>$K_{\text{all}1}$</td>
<td>$w_1, w_2, w_3, w_4$</td>
<td>Kernel combining $K_{\text{traj}}, K_{\text{lab}}, K_{\text{start}}$ and $K_{\text{end}}$</td>
<td>$6.3.6$</td>
</tr>
<tr>
<td>$K_{\text{all}2}$</td>
<td>$w_1, w_2, w_3, w_4, w_5$</td>
<td>Kernel combining $K_{\text{traj_lab}}, K_{\text{start}}$ and $K_{\text{end}}$</td>
<td>$6.3.7$</td>
</tr>
</tbody>
</table>

6.3.1 Substitution Functions for Sets of Geo-Labels

For sequences of sets of geo-labels $T_{\text{lab}}$, the substitution function $sub_{\text{lab}}$, given in Definition 6.3.1, expresses how many labels two sets of labels $L_i$ and $L_j$ have in common.

**Definition 6.3.1.** For two sets of labels $L_i, L_j$, we define the substitution function $sub_{\text{lab}}$ as:

$$
sub_{\text{lab}}(L_i, L_j) = -1 + \frac{|N_i \cap N_j| + |M_i \cap M_j|}{\sqrt{|N_i| + |M_i|)(|N_j| + |M_j|)},
$$

where $N_m = \{l_1 \mid l \in L_m\}$ and $M_m = \{l_2 \mid l \in L_m\}$, for $m = i, j$.

Note that $l_1$ indicates the first element of the (name, type) pair, i.e. the name, and $l_2$ 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 add $-1$ to create a function that ranges from $0$ to $-1$, which is $0$ when two objects are equal, similar to the negative of the Euclidean distance (i.e. $sub_1$). We use this range since the alignment measures in Chapter 4 are defined using the max function.

The function $sub_{\text{lab}}$ can be expressed as the dot products of vectors that have a length equal to the total number of possible name and
type labels. In this vector we use the value 1 if a particular label is in the set, and 0 otherwise. Let \( n \) be this vector representation of \( L_i \) and \( m \) the vector representation of \( L_j \). Then Definition 6.3.1 can be represented as:

\[
\text{sub}_{\text{lab}}(n, m) = -1 + \frac{n \cdot m}{\sqrt{n \cdot n \cdot m \cdot m}}. \tag{6.1}
\]

This is a positive semi-definite kernel, where \( \frac{1}{\sqrt{n \cdot n \cdot m \cdot m}} \) is the regular kernel normalization. This means that \( \exp(\text{sub}_{\text{lab}}) \), which is used in the soft-max kernels, is also a positive definite kernel. Besides a substitution function we require a gap penalty if we use edit distance, we fix this to the minimum value of \( \text{sub}_{\text{lab}} \), which is \(-1\).

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 of Definition 6.2.2. This leads to Definition 6.3.2.

**Definition 6.3.2.** Let \( S_{\text{traj}_{\text{lab}}} \) and \( T_{\text{traj}_{\text{lab}}} \) be two combinations of trajectories and sequences of sets of geo-labels. Furthermore, \((S_i, L_i) = S_{\text{traj}_{\text{lab}}}(i)\) and \((T_j, L_j) = T_{\text{traj}_{\text{lab}}}(j)\). Then

\[
\text{sub}_{\text{traj}_{\text{lab}}}(w_1, w_2)(S_i, L_i, T_j, L_j) = w_1 \gamma \text{sub}_m(S_i, T_j) + w_2 \text{sub}_{\text{lab}}(L_i, L_j),
\]

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 \( \text{sub}_2 \) from Chapter 4 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 \( \text{sub}_{\text{traj}_{\text{lab}}} \) 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.

### 6.3.2 Kernels

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, given in Definition 6.3.3.

**Definition 6.3.3.** For all \((\text{stopped}_i, L_i)\) and \((\text{stopped}_j, L_j)\) in a set of start/end objects, we compute a kernel matrix as:

\[
K_{\Phi}(i, j) = 1 + [\text{stopped}_i = \text{stopped}_j] + \text{sub}_{\text{lab}}(L_i, L_j),
\]
where \( \phi = \text{start} \) for a set of start objects and \( \phi = \text{end} \) for a set of end objects.

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. Furthermore we add +1 to compensate for the −1 in the definition of \( \text{sub}_{\text{lab}} \). Using Definition 6.3.3 we get a kernel \( K_{\text{start}} \) for the start objects and a kernel \( K_{\text{end}} \) for the end objects. These kernels are positive semi-definite, since the function \( \text{sub}_{\text{lab}} \), the boolean function \([\text{stopped}_i = \text{stopped}_j]\) and the constant 1 are PSD.

Chapter 4 defines different alignment kernels in which we use the substitution functions defined above to get kernels for the sequence of sets of geo-labels \( T_{\text{lab}} \) and the combined version \( T_{\text{traj_lab}} \). To avoid unnecessary reduplication of definitions, we give an abridged definition for kernels for sequences of sets of geo-labels in Definition 6.3.4.

**Definition 6.3.4.** \( K_{\text{lab}} \) kernels for sequences of sets of geo-labels \( T_{\text{lab}} \) are computed by using the \( \text{sub}_{\text{lab}} \) substitution function in \( \text{sim}_{\text{max}} \) (Definition 4.2.7), \( \text{sim}_{\text{maxnorm}} \) (Definition 4.2.8) and \( \text{sim}_{\text{softmax}} \) (Definition 4.2.9). Then, Definitions 4.2.10 and 4.2.11 are used to compute kernels.

An example of such a kernel is \( K_{\text{lab}},\text{max,DTW} \) which is the kernel that uses the regular non normalized dynamic time warping measure applied to the sets of labels sequence.

For the kernels for \( T_{\text{traj_lab}} \) we do almost the same as for the \( K_{\text{lab}} \) kernels above; see Definition 6.3.5.

**Definition 6.3.5.** \( K_{\text{traj_lab}} \) kernels for sequences of sets of geo-labels combined with normal trajectories \( T_{\text{traj_lab}} \) are computed by using the \( \text{sub}_{\text{traj_lab}} \) substitution function in \( \text{sim}_{\text{max}} \) (Definition 4.2.7) and \( \text{sim}_{\text{maxnorm}} \) (Definition 4.2.8). The \( \text{sub}_{\text{traj_lab}} \) substitution function is used in \( \text{sim}_{\text{softmax}} \) (Definition 4.2.9). Then, Definitions 4.2.10 and 4.2.11 are used to compute kernels. The superscripts \( w_1,w_2 \) are used for the weights of the substitution function.

In this chapter, we give the subscript \( \text{traj} \) to kernels for normal trajectories \( T \), i.e. the kernels as studied in Chapter 4.

For explanatory purposes, we work out \( \text{sim}_{\text{traj_lab,softmax,DTW}}^{\beta,w_1,w_2} \) i.e. the similarity function on which \( K_{\text{traj_lab,softmax,DTW}}^{\beta,w_1,w_2} \) is based:

\[
\text{sim}_{\text{traj_lab,softmax,DTW}}^{\beta,w_1,w_2}(S_{\text{traj_lab}}, T_{\text{traj_lab}}) = \sum_{\pi \in \Pi(S,T)} \exp(\beta \sum_{i=1}^{|\pi|} \text{sub}_{\text{traj_lab}}^{w_1,w_2}(S_{\text{traj_lab}}(\pi_1(i)), T_{\text{traj_lab}}(\pi_2(i))))
\]

\[
= \sum_{\pi \in \Pi(S,T)} \prod_{i=1}^{|\pi|} \exp(\beta \text{sub}_{\text{traj_lab}}^{w_1,w_2}(S_{\text{traj_lab}}(\pi_1(i)), T_{\text{traj_lab}}(\pi_2(i))))
\]

(6.2)
where (with $S_1$ as shorthand for $S_{\text{traj,lab}}(\pi_1(i))$ and $T_1$ as shorthand for $T_{\text{traj,lab}}(\pi_2(i)))$,

$$
\exp(\beta \text{sub}_{\text{traj,lab}_2}(w_1,w_2)(S_1,T_1)) = \exp(\beta(w_1 \gamma \text{sub}_2((S_1)_1,(T_1)_1))
+ w_2 \text{sub}_2((S_1)_2,(T_1)_2))) = \exp(\beta w_1 \gamma \text{sub}_2((S_1)_1,(T_1)_1)) \cdot \exp(\beta w_2 \text{sub}_2((S_1)_2,(T_1)_2)))
$$

(6.3)

Both $\exp(\text{sub}_2)$ and $\exp(\text{sub}_\text{lab})$ are positive semi-definite kernels. Hence we are multiplying PSD kernels, and therefore the use of the substitution function $\text{sub}_{\text{traj,lab}_2}$ leads to a positive semi-definite softmax kernel.

We combine the kernels defined above together by taking weighted sums in two ways. The first way, in Definition 6.3.6, uses separate kernels for the trajectories $T$ and the geo-labels $T_{\text{lab}}$.

**Definition 6.3.6.** Let $K_{\text{traj}}$ be an alignment kernel for trajectories, $K_{\text{lab}}$ an alignment kernel for sequences of sets of geo-labels, and $K_{\text{start}}$ and $K_{\text{end}}$ kernels for start and end objects, then

$$K_{\text{all}1} = w_1 K_{\text{traj}} + w_2 K_{\text{lab}} + w_3 K_{\text{start}} + w_4 K_{\text{end}},$$

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 $T_{\text{traj,lab}}$, which is given in Definition 6.3.7.

**Definition 6.3.7.** Let $K_{\text{traj,lab}}$ be an alignment kernel for trajectories combined with their sequence of sets of geo-labels, and $K_{\text{start}}$ and $K_{\text{end}}$ kernels for start and end objects, then

$$K_{\text{all}2} = w_3 K_{\text{traj,lab}} + w_4 K_{\text{start}} + w_5 K_{\text{end}},$$

with $w_1 + w_2 = 1$ and $w_3 + w_4 + w_5 = 1$.

Depending on the alignment kernels that are used for $T$, $T_{\text{lab}}$ and $T_{\text{traj,lab}}$, 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 $K_{\text{traj}}$, $K_{\text{lab}}$ and $K_{\text{traj,lab}}$ we use the dynamic programming approach, mentioned earlier in this thesis.

### 6.4 Experiments

Like we did in the previous two chapters, the goal of our experiments is to investigate the performance of the defined kernels, which incorporate geographical domain knowledge, on the three typical machine learning tasks of clustering, classification and outlier detection. We are interested to see if performance increases over using only trajectory information and under what settings.
6.4 Experiments

6.4.1 Experimental Set-Up

In Chapter 4 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 $T_{lab}$ would be very large in this setting, which seems useless. All trajectories are compressed using $\text{pls}_{E_{u=2}}$ with $\epsilon = 50\text{m}$, 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. For completeness, in the classification task we will also include the $\epsilon = 1000\text{m}$ setting, since under this setting the best results are obtained in that experiment. We use the time weight setting of $w = 0$; thus time is ignored.

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

The different weight settings for the kernels from Definition 6.3.6 and 6.3.7 that we experiment with are different per task. Hence, we will mention them there. This also holds for the $\beta$ parameter when applicable.

6.4.2 Clustering

The gold standard clustering dataset that we used in the previous two chapters is not well suited to evaluate the performance of clustering using the domain knowledge kernels that we defined in this chapter, 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 2.3.3) dataset. The difference with the classification set is that all vessel types are included. As we mentioned above, for this dataset there is domain knowledge. Since we do not have a gold standard, we cannot evaluate the clustering results quantitatively. Our evaluation is qualitative in nature, and consists of show-
Adding domain knowledge

Three illustrative clustering examples for three different settings of the weights in Definition 6.3.6. After some experimentation, we chose as the $K_{\text{traj}}$ kernel the edit distance variant: $K_{\text{traj,maxnorm,ED}}$, with $\gamma = -0.01$, and as $K_{\text{lab}}$ kernel we took $K_{\text{lab,maxnorm,ED}}$. The first setting for $K_{\text{all}}$ is $w_1 = \frac{1}{2}, w_2, w_3, w_4 = \frac{1}{7}$, 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, $w_1 = 1, w_2, w_3, w_4 = 0$, $K_{\text{traj}}$, and one for just the ontological information $w_1 = 0, w_2, w_3, w_4 = \frac{1}{3}$, $K_{\text{onto}}$. 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 of Section 2.1.1.2. For each kernel, clustering is done 100 times with random initializations. This process is repeated 10 times. 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. We could have also used a clustering algorithm that has no parameter $k$, e.g. density based ones are popular in combination with moving object trajectories. However, these algorithms have other parameters that need to be determined, which also is a manual process. Moreover, we are not sure that the combined kernel that we have defined induces a feature space in which density based algorithms work well.

Examples

The three 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 ($K_{\text{traj}}$ and $K_{\text{onto}}$) 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.

Figure 14 illustrates the behavior of vessels anchoring in a specified anchoring area. In Figure 14A 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 Figure 14B. 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 (Figure 14C). 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”.


Figure 14: Example of a cluster of trajectories showing anchoring behavior. The start of a trajectory is indicated by a dot, the end by an asterisk. Figure A shows a cluster generated with the combined kernel. Figure B shows the most similar cluster from clustering with the trajectory information only kernel, and figure C shows the most similar cluster from clustering with the domain knowledge only kernel.

The cluster in Figure 15A shows the docking behavior of vessels in a certain part of the harbor. There is some noise in the cluster; not all trajectories go to that part. This cluster is a result of clustering with the combined kernel \( K_{\text{comb}} \). We see something similar in Figure 15B. However, here we have used the kernel \( K_{\text{traj}} \), which uses only the trajectory information. The result is that the cluster also contains trajectories starting from anchoring areas. This differs from the combined setting, where we only have trajectories coming from outside the observation area, i.e. the open sea. In Figure 15C, the ontology only setting, \( K_{\text{onto}} \), we also have trajectories going to the harbor from outside the observation area, but, all trajectories stop in the deep water lane, not on a dock. This is somewhat odd and is the result of only considering ontological information. In the combined case we have stopped in the deep water lane, but also in the adjacent docks. Thus, the combined case shows the behavior of “docking in a certain part of the harbor, coming directly from the open sea”.

The trajectories in Figure 16A, on which we zoom in in Figure 16B, 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 Figure 16B we see that none of them pass through the deep water lane. Figure 16C 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, Figure 16D, 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 be-
Figure 15: Example of a cluster of trajectories showing docking behavior. The start of a trajectory is indicated by a dot, the end by an asterisk. Figure A shows a cluster generated with the combined kernel. Figure B shows the most similar cluster from clustering with the trajectory information only kernel, and figure C shows the most similar cluster from clustering with the domain knowledge only kernel.

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.

6.4.3 Classification

In the classification task we use the dataset from Section 2.3.3, which contains trajectories of the four most common vessel types. From this set we randomly select 200 trajectories for each type, creating a 800 trajectories dataset. Note that this set is larger than the one we used in Chapter 4. Due to the fact that we do not have a ‘no compression’ setting, we can easily run the experiments with more data.

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 Chapter 4. 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: $\frac{1}{1024}, \frac{1}{256}, \frac{1}{64}, \frac{1}{16}, \frac{1}{4}, 1$.

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: $\frac{1}{1024}, \frac{1}{256}, \frac{1}{64}, \frac{1}{16}, \frac{1}{4}, 1$. 
Figure 16: Example of a cluster of trajectories continuing through the harbor, not going through the deep water lane. The start of a trajectory is indicated by a dot, the end by an asterisk. Figures A and B shows a cluster generated with the combined kernel. Figure C shows the most similar cluster from clustering with the trajectory information only kernel, and figure D shows the most similar cluster from clustering with the domain knowledge only kernel.
With the best performing basic kernels we create variants of $K_{all_1}$ and $K_{all_2}$ with different weight settings, to cover different combinations of the basic kernels. We give the specific settings in the Results section.

The classification algorithm set-up is the same as in the classification experiment of Chapter 4, we use the C-SVC Support Vector Machines from LibSVM; see Section 2.1.1. We use a 10-fold cross validation set-up to evaluate the kernel performance, with inner 10-fold cross validation to optimize the C parameter.

### Results

In the results we present the mean classification accuracy over 10 folds. To statistically compare accuracies for two kernels we use a two-tailed paired t-test with $p < 0.05$. The tables give results for two different compression settings: $\epsilon = 50,1000m$. If a kernel has parameter settings we give the results for the best parameter settings and also give the mean, maximum and minimum accuracies.

Table 15 presents the results for the different basic kernels. 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, and obviously the same for both $\epsilon$ settings.

The $K_{lab}$ kernels are relatively close together in performance, with the DTW max and normalized max performing the worst. The best performance overall (60.5%) is for the $K_{lab,softmax,ED}$ kernel, this however does not differ significantly from the best performing DTW kernel (60.13%).

In case of the $K_{traj,lab}$ kernels, the best performance (75.25%) is clearly for the $K_{traj,lab,maxnorm,ED}$ kernel. It does not differ significantly from the best $K_{traj}$ score (75.63%). Also, there is no significant difference from the score (73.38%) for the $K_{traj,lab,max,ED}$ kernel. However, there is a significant difference with the best DTW variant score (65.25%) and the best edit distance soft-max score (67.38%).

Based on Table 15 we take the kernel $K_{lab,softmax,ED}$ for the sequences of geo-labels, with $\beta = \frac{1}{16}$. We combine this kernel with $K_{traj,maxnorm,ED}$ with $g = -0.01$ and the start and end object kernels to create a kernel incorporating domain knowledge using Definition 6.3.6.

The results for different weight settings are shown in Table 16. 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 15 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, as in Definition 6.3.7. Results are shown in Table 17. The combined kernels
Table 15: Mean classification accuracy (%) for different basic kernels.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>$\epsilon = 50$</th>
<th>$\epsilon = 1000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{\text{traj, maxnorm, ED}}$</td>
<td>best ($g = -0.01$)</td>
<td>73.50 75.63</td>
</tr>
<tr>
<td>$K_{\text{start}}$</td>
<td>63.63 63.63</td>
<td></td>
</tr>
<tr>
<td>$K_{\text{end}}$</td>
<td>62.88 62.88</td>
<td></td>
</tr>
<tr>
<td>$K_{\text{lab, max, DTW}}$</td>
<td>45.38 44.75</td>
<td></td>
</tr>
<tr>
<td>$K_{\text{lab, maxnorm, DTW}}$</td>
<td>50.13 51.0</td>
<td></td>
</tr>
<tr>
<td>$K_{\text{lab, softmax, DTW}}$</td>
<td>best ($\beta = \frac{1}{4}$)</td>
<td>60.13 59.38</td>
</tr>
<tr>
<td>mean</td>
<td>58.67 59.13 57.63 59.83 59.38</td>
<td></td>
</tr>
<tr>
<td>$K_{\text{lab, max, ED}}$</td>
<td>52.75 56.13</td>
<td></td>
</tr>
<tr>
<td>$K_{\text{lab, maxnorm, ED}}$</td>
<td>58.75 57.75</td>
<td></td>
</tr>
<tr>
<td>$K_{\text{lab, softmax, ED}}$</td>
<td>best ($\beta = \frac{1}{4}$)</td>
<td>58.25 60.5</td>
</tr>
<tr>
<td>mean</td>
<td>58.52 59.0 58.45 60.13 59.75</td>
<td></td>
</tr>
<tr>
<td>$K_{\text{traj, lab, max, DTW}}$</td>
<td>best ($w_1 = 0.9, w_2 = 0.1$)</td>
<td>45.88 52.0</td>
</tr>
<tr>
<td>mean</td>
<td>46.85 49.63 45.88 51.65 52.0 51.25</td>
<td></td>
</tr>
<tr>
<td>$K_{\text{traj, lab, maxnorm, DTW}}$</td>
<td>best ($w_1 = 0.5, w_2 = 0.5$)</td>
<td>54.38 57.0</td>
</tr>
<tr>
<td>mean</td>
<td>54.45 55.25 54.0 55.75 57.0 54.0</td>
<td></td>
</tr>
<tr>
<td>$K_{\text{traj, lab, softmax, DTW}}$</td>
<td>best ($w_1 = 0.5, w_2 = 0.5, \beta = \frac{1}{4}$)</td>
<td>64.88 65.25</td>
</tr>
<tr>
<td>mean</td>
<td>62.14 59.0 58.13 62.26 62.85 62.88</td>
<td></td>
</tr>
<tr>
<td>$K_{\text{traj, lab, max, ED}}$</td>
<td>best ($w_1 = 0.7, w_2 = 0.3$)</td>
<td>73.38 72.63</td>
</tr>
<tr>
<td>mean</td>
<td>71.37 73.38 68.7 71.15 72.63 68.8</td>
<td></td>
</tr>
<tr>
<td>$K_{\text{traj, lab, maxnorm, ED}}$</td>
<td>best ($w_1 = 0.7, w_2 = 0.3$)</td>
<td>74.5 75.25</td>
</tr>
<tr>
<td>mean</td>
<td>72.17 74.5 68.3 73.87 75.25 71.38</td>
<td></td>
</tr>
<tr>
<td>$K_{\text{traj, lab, softmax, ED}}$</td>
<td>best ($w_1 = 0.9, w_2 = 0.1, \beta = 1$)</td>
<td>59.38 67.38</td>
</tr>
<tr>
<td>mean</td>
<td>61.63 64.88 67.5 62.59 67.38 58.88</td>
<td></td>
</tr>
</tbody>
</table>
show better performance than the $K_{\text{traj}}$ kernel. However, the difference is only significant for one setting.

### 6.4.3.2 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.

We saw earlier in Chapter 4 that the start and end of a trajectory already contain a lot of information about the type of vessel and we see the same here. The kernel that just takes this type of information
into account \((w_3, w_4 = \frac{1}{2})\) 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 \(T_{\text{traj}, \text{lab}}\), and almost the best for the sequences of geo-labels \(T_{\text{lab}}\). Thus, for the three types of data used here, edit distance seems to be a good choice for classification.

### 6.4.4 Outlier Detection

For our outlier detection experiment we use the dataset from Section 2.3.4, which we also used in the previous two chapters. This set consists of 786 trajectories, 39 of which are outliers.

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 Chapter 4 that the best performing \(K_{\text{traj}}\) kernels are the DTW soft-max variants. For the other kernels that have parameters, we use the same settings as in the classification experiment. The weights \(w_1\) and \(w_2\) are varied over the combinations: \((0.9,0.1), (0.7,0.3), (0.5,0.5), (0.3,0.7)\) and \((0.1,0.9)\). And for the \(\beta\) parameter we take the settings: \(1^{1024}, 1^{256}, 1^{64}, 1^{16}, 1^{4}\) and 1.

Using the best performing kernels we create variants of \(K_{\text{all}}^1\) and \(K_{\text{all}}^2\), 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 Chapter 4 and 5. We use a one-class Support Vector Machine. The 747 normal trajectories in the dataset are randomly split into a training set of \(\frac{2}{3}\) and a test set of the rest, to which we add the outlying trajectories. This split is done 10 times per kernel, and for each split the one-class SVM is optimized on the train set using 10-fold cross validation.

#### 6.4.4.1 Results

We present the mean precision@39 results over 10 folds. We statistically compare the results for two kernels using a two-tailed Student t-test with \(p < 0.05\). The presentation of the results follows the same set-up as before.

In Table 18 we see the results for the different basic kernels. The best performance (0.82) is achieved by the \(K_{\text{traj}, \text{lab}}, \text{softmax}, \text{DTW}\) kernel, which does not differ significantly from the best performance (0.80) for the \(K_{\text{traj}, \text{softmax}, \text{DTW}}\) kernel. It does differ significantly from the best score (0.76) for the \(K_{\text{traj}, \text{lab}}, \text{ED}\) kernels and the best \(K_{\text{lab}}\) scores. Moreover, for all the \(K_{\text{lab}}\) and \(K_{\text{traj}, \text{lab}}\) kernels, the soft-max kernels achieve a significantly better score than the non soft-max kernels.
<table>
<thead>
<tr>
<th>Kernel Details</th>
<th>Precision@39 ($\epsilon = 50$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{traj,softmax,DTW}$ best ($\beta = 64$)</td>
<td>0.80</td>
</tr>
<tr>
<td>$K_{start}$</td>
<td>0.08</td>
</tr>
<tr>
<td>$K_{end}$</td>
<td>0.15</td>
</tr>
<tr>
<td>$K_{lab,max,DTW}$</td>
<td>0.29</td>
</tr>
<tr>
<td>$K_{lab,maxnorm,DTW}$</td>
<td>0.22</td>
</tr>
<tr>
<td>$K_{lab,softmax,DTW}$ best ($\beta = 1$) mean$\max_{\min}$</td>
<td>0.67</td>
</tr>
<tr>
<td>$K_{lab,max,ED}$</td>
<td>0.29</td>
</tr>
<tr>
<td>$K_{lab,maxnorm,ED}$</td>
<td>0.26</td>
</tr>
<tr>
<td>$K_{lab,softmax,ED}$ best ($\beta = 1$) mean$\max_{\min}$</td>
<td>0.61</td>
</tr>
<tr>
<td>$K_{traj,lab,max,DTW}$ best ($w_1 = 0.9, w_2 = 0.1$) mean$\max_{\min}$</td>
<td>0.41</td>
</tr>
<tr>
<td>$K_{traj,lab,softmax,DTW}$ best ($w_1 = 0.1, w_2 = 0.9$) mean$\max_{\min}$</td>
<td>0.20</td>
</tr>
<tr>
<td>$K_{traj,lab,max,ED}$ best ($w_1 = 0.7, w_2 = 0.3$) mean$\max_{\min}$</td>
<td>0.43</td>
</tr>
<tr>
<td>$K_{traj,lab,maxnorm,ED}$ best ($w_1 = 0.1, w_2 = 0.9$) mean$\max_{\min}$</td>
<td>0.47</td>
</tr>
<tr>
<td>$K_{traj,lab,softmax,ED}$ best ($w_1 = 0.9, w_2 = 0.1, \beta = 1$) mean$\max_{\min}$</td>
<td>0.76</td>
</tr>
</tbody>
</table>
Table 19: Mean precision@39 for different $K_{\text{all}_1}$ kernels.

<table>
<thead>
<tr>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
<th>$w_4$</th>
<th>$\epsilon = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.80</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>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0.24$^-$</td>
</tr>
<tr>
<td>0</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>0.33$^-$</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>0</td>
<td>0.69$^-$</td>
</tr>
<tr>
<td>$\frac{3}{10}$</td>
<td>$\frac{3}{10}$</td>
<td>0</td>
<td>0</td>
<td>0.71$^-$</td>
</tr>
<tr>
<td>$\frac{9}{10}$</td>
<td>$\frac{1}{10}$</td>
<td>0</td>
<td>0</td>
<td>0.75$^-$</td>
</tr>
<tr>
<td>$\frac{1}{3}$</td>
<td>0</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>0.37$^-$</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{4}$</td>
<td>0.38$^-$</td>
</tr>
<tr>
<td>$\frac{1}{4}$</td>
<td>0</td>
<td>$\frac{1}{8}$</td>
<td>$\frac{1}{8}$</td>
<td>0.42$^-$</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{6}$</td>
<td>$\frac{1}{6}$</td>
<td>$\frac{1}{6}$</td>
<td>0.39$^-$</td>
</tr>
<tr>
<td>$\frac{3}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0.43$^-$</td>
</tr>
</tbody>
</table>

Table 20: Mean precision@39 for different $K_{\text{all}_2}$ kernels.

<table>
<thead>
<tr>
<th>$w_3$</th>
<th>$w_4$</th>
<th>$w_5$</th>
<th>$\epsilon = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.82</td>
</tr>
<tr>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>0.35$^-$</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{4}$</td>
<td>0.36$^-$</td>
</tr>
<tr>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{8}$</td>
<td>$\frac{1}{8}$</td>
<td>0.38$^-$</td>
</tr>
<tr>
<td>$\frac{3}{4}$</td>
<td>$\frac{1}{8}$</td>
<td>$\frac{1}{8}$</td>
<td>0.40$^-$</td>
</tr>
</tbody>
</table>

For $K_{\text{all}_1}$, we take from Table 18 the $K_{\text{lab}, \text{softmax}, \text{DTW}}$ kernel as basic kernel, with $\beta = 1$. We combine this with the $K_{\text{traj}, \text{softmax}, \text{DTW}}$ kernel for regular trajectories, and the start and end object kernels.

The results for different weight settings for $K_{\text{all}_1}$ are shown in Table 19. For each score a significant difference with the $K_{\text{traj}, \text{softmax}, \text{DTW}}$ 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. $K_{\text{traj}, \text{softmax}, \text{DTW}}$. $K_{\text{traj}, \text{lab}, \text{softmax}, \text{DTW}}$ has the best performance of the $K_{\text{traj}, \text{lab}}$ kernels. Thus we plug this one into Definition 6.3.7. We give the results in Table 20. Again all combinations are significantly outperformed by the $K_{\text{traj}, \text{softmax}, \text{DTW}}$ kernel, as indicated by a $^-$. 
6.4.4.2 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 $K_{\text{traj,lab,softmax,DTW}}$, 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 Chapter 4, 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.

6.5 Conclusions & Future Work

In this chapter we investigated the incorporation of geographical domain knowledge into the alignment similarity kernels of Chapter 4. To achieve this, geographical domain knowledge, i.e. types and labels of places and regions, was added to the regular trajectories in two forms. One form was to create a separate sequence of sets of geo-labels, and the other form was a combined version of the regular trajectory and the sequence of sets of geo-labels. Furthermore, we created special objects to represent the information about the start and the end of the trajectories.

In the clustering experiment the kernels with added domain knowledge discovered examples of interesting vessel behavior. The classification task of predicting vessel types showed that adding domain knowledge can increase classification accuracy significantly. In outlier detection no significantly better performance could be achieved.

For future work in clustering we would like to let domain experts label interesting vessel behavior. However, doing this labeling while incorporating domain knowledge can be quite difficult. It is also interesting to see what outliers are more dependent on geographical domain knowledge and how our method works for these cases.

Furthermore, we are interested in applying this kind of similarity in other domains where comparable domain knowledge exists. The domain knowledge is essentially a parameter of the similarity measure, which can be varied, so the quality and content of this information is of direct influence on the similarity. So, next to applying this similarity in other domains we should also consider other knowledge, i.e. ontologies, in the same domain.

In all three tasks, more complicated weighting schemes are possible for creating kernels. In the classification and outlier detection setting finding the optimal weights can be done automatically. However, for clustering this is more difficult, and playing around with weight settings is something left to a domain expert or end-user.