Incremental Semi-supervised Learning for Anomalous Trajectory Detection

Rowland R. Sillito

Doctor of Philosophy
Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh
2009

Abstract

The acquisition of a scene-specific normal behaviour model underlies many existing approaches to the problem of automated video surveillance. Since it is unrealistic to acquire a comprehensive set of labelled behaviours for every surveyed scenario, modelling normal behaviour typically corresponds to modelling the distribution of a large collection of unlabelled examples. In general, however, it would be desirable to be able to filter an unlabelled dataset to remove potentially anomalous examples.

This thesis proposes a simple semi-supervised learning framework that could allow a human operator to efficiently filter the examples used to construct a normal behaviour model by providing occasional feedback: Specifically, the classification output of the model under construction is used to filter the incoming sequence of unlabelled examples so that human approval is requested before incorporating any example classified as anomalous, while all other examples are automatically used for training.

A key component of the proposed framework is an incremental one-class learning algorithm which can be trained on a sequence of normal examples while allowing new examples to be classified at any stage during training. The proposed algorithm represents an initial set of training examples with a kernel density estimate, before using merging operations to incrementally construct a Gaussian mixture model while minimising an information-theoretic cost function. This algorithm is shown to outperform an existing state-of-the-art approach without requiring off-line model selection.

Throughout this thesis behaviours are considered in terms of whole motion trajectories: in order to apply the proposed algorithm, trajectories must be encoded with fixed length vectors. To determine an appropriate encoding strategy, an empirical comparison is conducted to determine the relative class-separability afforded by several different trajectory representations for a range of datasets. The results obtained suggest that the choice of representation makes a small but consistent difference to class separability, indicating that cubic B-Spline control points (fitted using least-squares regression) provide a good choice for use in subsequent experiments.

The proposed semi-supervised learning framework is tested on three different real trajectory datasets. In all cases the rate of human intervention requests drops steadily, reaching a usefully low level of $\sim 1\%$ in one case. A further experiment indicates that once a sufficient number of interventions has been provided, a high level of classification performance can be achieved even if subsequent requests are ignored. The automatic incorporation of unlabelled data is shown to improve classification performance in all cases, while a high level of classification performance is maintained even when unlabelled data containing a high proportion of anomalous examples is presented.

Acknowledgements

I would like to thank my supervisor Bob Fisher for providing unfaltering guidance, insight and encouragement throughout the course of this work. I have greatly appreciated his kind-spirited advice and wisdom, without which this thesis would not be the same. I would also like to thank my second supervisors Sethu Vijayakumar and James Bednar, and my examiners David Hogg and Amos Storkey for their insightful comments and suggestions.

I have greatly appreciated the stimulating conversations, inspiration and friendship of my colleagues in the IPAB and ANC institutes and the Neuroinformatics DTC, and would especially like to thank fellow 'Vision Lab' members (during my time: Ernesto Andrade, Scott Blunsden, Toby Breckon, Toby Collins and Tim Lukins) for their advice and moral support. I am also grateful to Irene Madison and Pat Ferguson for always being very helpful with administrative matters.

I gratefully acknowledge the funding for this work provided by the EPSRC and MRC through the Neuroinformatics Doctoral Training Centre.

Finally, I would like to thank my family for all their support and encouragement.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Rowland R. Sillito*)

Contents

Chapter 1

Introduction

This thesis contributes to a growing body of research concerning the development of computer vision algorithms for detecting unusual behaviour in video footage obtained in a surveillance context. Currently, real-world video surveillance places an unreasonable burden on human operators, whose role typically involves attempting to simultaneously monitor the output of large numbers of CCTV cameras. An unfortunate consequence of this situation is that only a small fraction of the footage being captured at any given moment is likely to be subject to human scrutiny [33]: moreover, without prior knowledge of the activities taking place at a given time, the fraction of observed footage is - unavoidably - chosen arbitrarily.

Therefore, while fully automated surveillance remains an intangible prospect, there is a clear potential role for computer vision in quantifying the relative salience of different video streams on the basis of some first-order interpretation of their content. Such strategies could significantly enhance the efficiency of human CCTV operators, by appropriately prioritising their attention at any given moment. To this end, a wide range of different techniques - discussed in detail in Chapter 2 - have been developed for modelling the typical patterns of behaviour that occur in video footage from a given scenario, and for detecting unusual events on this basis.

Although many classification problems are solved in a *supervised* fashion using human-labelled examples, the acquisition of a comprehensive set of labelled behaviours for every surveyed location is an unrealistic prospect. In this light, the majority of techniques for detecting unusual events involve learning - in an *unsupervised* fashion - statistical models that characterise the distribution of behaviours observed in a given set of video data. As a potential compromise between these strategies, this thesis implements and explores a *semi-supervised* framework that allows occasional human input to be incorporated in the construction of behaviour models, such that some control over their content can be exerted with minimal labelling effort.

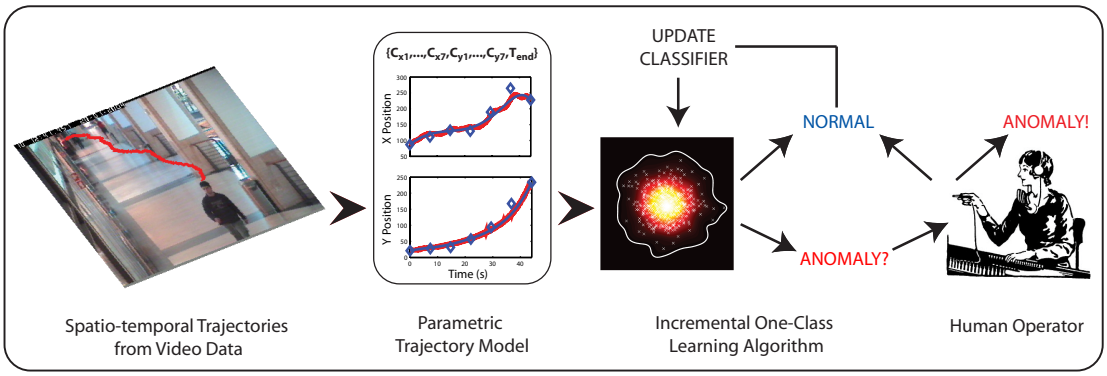


Figure 1.1: **Semi-supervised learning framework for normal behaviour modelling (see Chapter 5):** Trajectories represented in a fixed-dimensionality parametric form (see Chapter 4) are assessed by a classifier. New examples classified as normal are automatically used to train the classifier (see Chapter 3), while anomalous examples are passed to a human operator for approval (initially, all examples are classified as anomalous.)

1.1 Semi-supervised learning of a normal-behaviour model

This thesis proposes a semi-supervised learning framework for incrementally constructing a model of normal behaviour on the basis of a small number of human-labelled normal examples and a large number of unlabelled examples.

The proposed framework - illustrated in Figure 1.1 - constitutes an on-line learning scenario where the incremental construction of a normal behaviour model is allowed to take place in a largely unsupervised fashion, with one key exception: the approval of a human operator is requested before incorporating any new training examples of an unprecedented nature. Initially, *all* examples require human approval before being used for training: however, as the generalisation of the underlying normal behaviour model improves, an increasing proportion of new examples are automatically used for training.

By only requesting labels for the most unusual examples, the proposed framework allows a human operator to filter the training data used to construct a normal behaviour model with minimal effort. The role of human input thus consists in providing a "safety net" to prevent the inclusion of inappropriate training examples in a normal-behaviour model. In a fully unsupervised setting, it is possible that the finite sample of training data used to construct a behaviour model may contain a distribution of examples that is unrepresentative of normal activity: the proposed framework provides a means to sidestep such issues without resorting to the laborious manual labelling of all training instances.

Indeed, given that current approaches to automated surveillance are intended to

assist - rather than replace - human operators, it is desirable to be able to utilise the opportunity for occasional human feedback when training such systems. A series of experiments, presented in Chapter 5, use motion trajectory data from three different surveillance scenarios to demonstrate the potential efficacy of the proposed learning framework, indicating that high classification rates can be achieved while only requiring a fraction of training data to be labelled.

1.2 Key sub-problems

Implementing the learning framework described in the preceding section requires solutions to two main sub-problems: incrementally learning a model of normal behaviour without anomalous counter-examples, and representing behaviours occurring in video data in a consistent parametric form.

Incremental one-class learning. The proposed learning framework is intended to generate a model of normal behaviour on the basis of training data that consists solely of normal examples. This constraint is essential because, although the subsequent identification of anomalous examples is of crucial importance, in many situations the available training data may only contain examples of normal activity.

In this vein, a key component of the proposed framework is an underlying "one-class" learning algorithm capable of incrementally constructing a model of normal behaviour on the basis of a stream of normal training examples, while providing the ability to classify new examples at any point during training. In order for such an algorithm to be useful it must generalise parsimoniously, so that its ability to classify new examples as normal reflects the number of training examples that have been observed so far. Chapter 3 presents a potential solution to this problem based on density estimation: this technique represents an initial set of training examples with a kernel density estimate, before using merging operations to incrementally construct a Gaussian mixture model while minimising an information-theoretic cost function. The resulting algorithm is shown to be effective on a range of benchmark datasets and - crucially - does not require any off-line optimisation step to determine an appropriate level of model complexity.

Parametric behaviour representation. In order to use the proposed learning algorithm for behaviour classification it is necessary to represent behaviours in a consistent parametric form (ie. with fixed length vectors). Moreover, if learning is to take place incrementally, the chosen behaviour representation must - rather than being learned - be defined in a "hard-coded" fashion so that it is available at the start of training.

The review of current approaches to behaviour learning/classification presented in Chapter 2 indicates that motion trajectories - which have formed the basis for a large number of approaches - provide a suitable substrate for a hard-coded behaviour representation. In particular, a number of different techniques can be used to encode motion trajectories in a parametric form, thereby allowing the algorithm proposed in Chapter 3 to be applied. An investigation of the impact of different parametric trajectory representations on the ability to discriminate between different classes of trajectory is presented in Chapter 4. While it is clear that motion trajectories may not always constitute an adequate description of the activities observed in a given video sequence, they provide strong clues about its content and have the advantage of implicitly segmenting activity. Nonetheless, it should be noted that the training framework and learning algorithm explored in this thesis are not specific to trajectory data and could potentially be applied to other forms of behaviour representation.

1.3 Thesis overview

The contributions of this thesis are organised in terms of the following chapters:

Chapter 2 presents a detailed of overview of current approaches to behaviour modelling and classification, together with summaries of the techniques that have been used to detect anomalous behaviour, and the different ways in which current behaviour classifiers can be trained.

Chapter 3 proposes and evaluates a new incremental one-class learning algorithm. This algorithm is shown to outperform a current state-of-the-art approach across a range of benchmark datasets while avoiding the need for a preliminary learning step to determine the underlying model complexity.

Chapter 4 presents a comprehensive empirical comparison of techniques for representing motion trajectories with fixed-length vectors: The impact of different representations on the relative separability of motion classes from several different real trajectory datasets is measured over a range of dimensionalities.

Chapter 5 combines the the findings of the preceding chapters by implementing and exploring the proposed semi-supervised normal-behaviour learning framework. Classification performance, and user intervention rates are measured during simulated training runs on three different trajectory datasets.

Chapter 6 presents an evaluation of the main contributions of this thesis, together with a number of potential directions for future research.

Chapter 2

Behaviour Modeling for Anomalous Event Detection

The two complementary goals of automated surveillance systems are the recognition of specific types of behaviour, and the identification of previously-unseen or atypical behaviour. A wide variety of representations and learning strategies have been used to address these problems: This chapter presents a review of existing work in the field and examines several important dimensions along which existing work can be categorised. Firstly, Section 2.1 provides an overview of the different activity representations that can be extracted from video footage; Section 2.2 then explores the different ways in which behaviours can be modelled once phrased in terms of these representations. The final sections of this chapter review techniques for anomalous behaviour detection (Section 2.3), and the types of training procedure implicit in different behaviour modelling strategies (Section 2.4).

2.1 Representing Video Footage

Providing an interpretation (or indicating an absence therein) of the behaviour occurring in a video sequence depends on the extraction of meaningful intermediate representations from raw video data. This section reviews the different representational strategies that have been used to "distill" video footage for subsequent behaviour modelling and classification. The choice of representation has a large bearing on the types of behaviour that can subsequently be recognised, and may in some cases be closely linked to a specific classification task (eg. recognising fights). This section highlights three main types of activity representation, reviewed as follows: motion trajectories; articulated motion, and scene-wide activity descriptors.

2.1.1 Trajectories

Motion trajectories provide one of the most intuitive cues for distinguishing between different types of behaviour. It is clear that the sequence of an individual's locations over time, although far from a complete encapsulation of their activity, would betray the occurrence of many of the different types of behaviour. In this light, motion trajectories provide a highly efficient means of summarising video data. Indeed they have formed the basis for a large body of work on automated surveillance [59, 33], and have also been applied with success to the problem of video indexing and retrieval [9]. This section reviews the strategies that have been used to represent trajectory data for behaviour modelling purposes.

Raw trajectory data

Perhaps the simplest solution is to construct geometric models directly from trajectory data, in order to model the spatial extent of typical paths in a scene. This technique has been adopted by Fernyhough et al. in [41]; Makris and Ellis in [82]; Junejo et al. in [66, 67], Piciarelli and Foresti in [100, 101]; and most recently by Wang and Grimson in [157]. We return to these approaches and discuss the trajectory modelling strategies they employ in detail in Section 2.2.3.

Discretised representation

Instead of dealing with whole trajectories, an attractive strategy is to construct a discrete "alphabet" of low-level primitives (eg. common local motion transitions), which can then be used to represent trajectories as discrete sequences.

This technique was employed by Johnson and Hogg in [64], in one of the earliest papers on trajectory modelling. To acquire a discretised representation, a large set of trajectories is decomposed into a set of 4-dimensional "flow vectors" describing local motion transitions $f = (x, y, \Delta x, \Delta y)$, from which a finite set of prototype vectors is obtained using a competitive neural network clustering algorithm [119]. The clustering algorithm is constrained to represent an equal number of training examples with each prototype vector, ensuring that the distribution of prototypes reflects that of the underlying data. A very similar motion quantisation approach was adopted by Stauffer and Grimson in [136], augmenting the vector space with a dimension corresponding to the area (of the image-plane silhouette) of each moving object, so that $f = (x, y, \Delta x, \Delta y, size)$.

Sumpter and Bulpitt use the same technique in [138] to acquire a discretised representation of flock of sheep and a "robotic sheepdog"; this is formulated in a 11-

dimensional space corresponding to the local flow vector of the flock, 5 spline parameters describing the shape of the flock, and the location of the "sheepdog" at a given instant. It is also possible to prespecify a quantisation without a learning step: Wang and Grimson create a motion "codebook" in [154] by dividing the image plane into a grid of 10x10 pixel regions and categorising optic flow within a region as one of 4 possible directions.

Another type of discretised trajectory representation can be obtained using the geometric path-modelling approach described in Section 2.2.3. In [83] Makris and Ellis build a network of "sub-path" regions in a scene, allowing trajectories to be represented as discrete sequences corresponding to traversals of the network. A similar trajectory representation can also be obtained using the on-line geometric path clustering approach proposed by Piciarelli and Foresti in [101], which generates a tree of possible routes through a scene.

In a similar vein, Nait-Charif & McKenna describe activity within a scene in [92] (in a care home environment) in terms of transitions between entry zones, inactivity zones, and exit zones. The regions are learned by fitting a Gaussian mixture model, as described in [85], to the start and end points of trajectories. Trajectories can then be described in terms of transitions between, and occupancies of, the learned zones. The idea of representing video footage as sequences of discrete low-level primitives has been widely applied to representations other than trajectories (see Section 2.1.3), and affords a wide range of different techniques for subsequently modelling behaviours (see Section 2.2).

Parametric representation

The preceding discretised approaches to trajectory representation provide one solution to the problem posed by the unbounded nature of trajectory data. Another solution, which makes a further set of machine learning techniques applicable to behaviour modelling, is to project trajectory data into a parameter space of fixed dimensionality. If the dimensionality is sufficiently low, techniques such as density estimation can be applied; otherwise, it is at least possible to measure similarity using the Euclidean distance.

Noting that a trajectory consists of two independent 1-dimensional time series (ie. regularly sampled X and Y coordinates), it is possible to apply 1D signal approximation techniques to parametrise trajectories. This technique is employed by Naftel and Khalid in [90, 89] where the Discrete Fourier Transform (DFT) is used to approximate the X and Y components of trajectories. The DFT projects signal information from the time domain to the frequency domain, allowing signals to be approximated to ar-

bitrary levels of accuracy/dimensionality by disregarding a proportion of the higher frequency coefficients. In [90, 89], DFT coefficients for the X and Y components of each trajectory are concatenated to yield fixed-length vectors, which are then used to cluster the trajectories. A similar strategy is also employed by Sahouria and Zakhour [120] where the X and Y components of trajectories are represented by a set of wavelet coefficients (obtained using the Haar wavelet transform), for the purpose of video indexing. More recently, Sillito and Fisher have adopted a least-squares cubic B-Spline trajectory representation as the basis for an incremental trajectory modelling algorithm in [133] (see Chapter 5).

The only existing comparison of low-dimensional trajectory parametrisations is presented in [90], where Naftel and Khalid compare DFT with Chebyshev polynomial approximation and - using the shopping mall scenario from the CAVIAR dataset [42] - find that DFT allows query trajectories corrupted by the addition of random noise to be most reliably matched to their non-corrupted counterparts. However, this comparison - which only utilises a single dataset - does not address the important question of how well different classes of trajectories are separated when encoded using different representations. Moreover, it does not include previously adopted trajectory representations such as Haar wavelet coefficients [120] and cubic B-spline control points [133]. To address these issues, Sillito and Fisher [134] measured the class separability of several different trajectory datasets when represented using DFT, Chebyshev, Haar wavelet, and cubic B-Spline representations, finding that Chebyshev and cubic B-Spline representations yield a marginal improvement in class separability for most datasets (see Chapter 4 for further details).

Alternatively, if very high dimensional representations are acceptable, a simpler approach to vectorising trajectory data is to specify a maximum vector length (or obtain this directly from a given dataset), and thereafter generate vectors of fixed length by filling the first N elements with the coordinates of a given trajectory, and filling any remaining unused elements by repeating the final coordinates of the trajectory. This "padding" approach has been employed by Hu et al. in [61], to preprocess trajectories before clustering them using a variant of the Self-Organising Map learning algorithm. In a later paper, [60], Hu et al. also propose another trajectory parametrisation, which involves coarsely subsampling trajectories and then interpolating them to achieve the desired dimensionality (note that unlike the padding approach this disregards temporal information). Since the high dimensionality of padded/resampled vectors prohibits modelling using density estimation techniques, their principal attraction lies in enabling the Euclidean distance to be used as trajectory similarity measure.

With the equivalent aim of providing a similarity measure for trajectory clustering,

Porikli et al. propose another - entirely different - style of trajectory parametrisation in [105], where each individual trajectory is represented by its own Hidden Markov Model (see Section 2.2.1). This allows the similarity of pair of trajectories to be measured by assessing the likelihood of one being generated by another's HMM parameters and vice versa, using a symmetrised distance function. Note that, although possible, there is no reason to enforce fixed dimensionality on this particular trajectory parametrisation (different numbers of Hidden states are chosen according to trajectory complexity in [105]) as it provides a similarity metric without requiring the direct comparison (eg. by Euclidean distance) of parameter vectors.

In certain cases (eg. where object tracking is not feasible), trajectory data can be usefully represented in terms of its constituent local motion transitions. In [98], Owens and Hunter parametrised each step of a trajectory with a vector encoding current location along with moving averages of the location and the x and y components of velocity and acceleration:

$$f = \left(x, y, s(x), s(y), s(dx), s(dy), s(d^2x), s(d^2y)\right)$$

Johnson and Hogg adopt a similar representation in [65], seeking to model the probability of a certain transition occurring given a history of three recent locations. The first step in achieving this is to represent a trajectory dataset with a set of 8-dimensional vectors, whose density is then modelled:

$$f = (\Delta x_t, \Delta y_t, x_t, y_t, x_{t-1}, y_{t-1}, x_{t-2}, y_{t-2})$$

Saleemi et al. employ a similar representational strategy in [121] using 5-dimensional vectors corresponding to two pairs of coordinates separated by a certain time interval (which must be greater than 0 and less than 5 seconds), and also model the density of this parametrisation:

$$f = \left(x_{(t+\Delta t)}, y_{(t+\Delta t)}, x_t, y_t, \Delta t\right)$$

Modelling the density of local motion parametrisations affords a wide range of possibilities for behaviour synthesis and classification, and is discussed in further detail in Section 2.2.2.

Landmark-based representation

A small number of approaches parametrise trajectories in terms of a set of hand-coded landmarks within a scene, sidestepping the need for the modelling approaches described in Section 2.2. In [88], Morris and Hogg describe a method for analysing the trajectories generated by pedestrians walking through a car park, on the basis of their

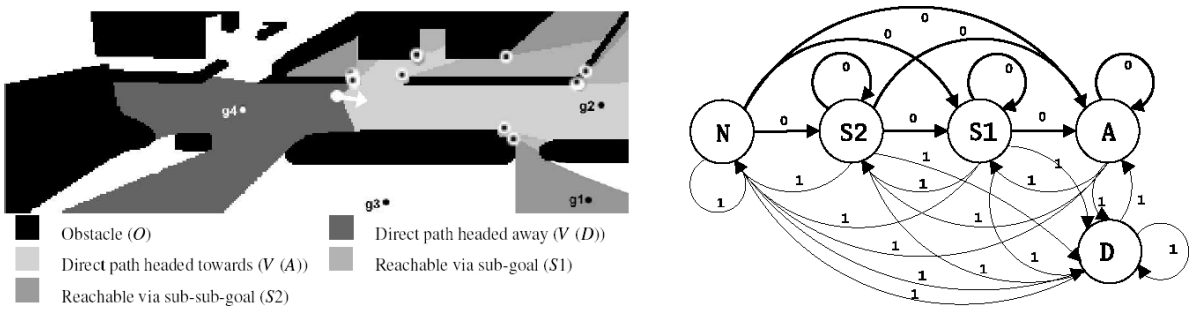


Figure 2.1: Hand-labelled scene and state transition network from [32].

'interactions' with parked cars. Using a handcrafted map of parked cars, each trajectory is decomposed into pairs of speed and 'distance to nearest car' parameters sampled at locations which correspond to global proximity minima for each car in the scene. A cumulative histogram of all the {speed,distance} pairs sampled is constructed, and then used to estimate a probability distribution over those parameters, which is used to represent each trajectory with a vector encoding the 5 least likely {speed,distance} pairs observed.

In a similar vein, Dee and Hogg [32] propose a technique for analysing traversals of scene networks, by quantifying the extent to which a given behaviour can be regarded as "goal directed". The essential premise of their approach is that an anomalous motion trajectory cannot be explained as an attempt to move efficiently from one location to another. For a given scene, Dee and Hogg construct a hand-crafted map encoding the regions corresponding to destinations/goals and entry points, together with the sub-goals which must be traversed when moving between them. This map has a corresponding topological network representation where nodes correspond to map regions, with edges joining nodes whose map regions are visible/accessible from one another (see Figure 2.1). For each possible goal a version of this network is instantiated so that transitions corresponding to motion away from the goal have an associated cost. For a given trajectory, the network traversal cost corresponding to each goal is calculated, and used to identify the goal (with lowest cost) that best explains the trajectory: the cost value associated with this goal serves to quantify the goal-directed nature of the trajectory, and can be used to identify unusual behaviour. Dee and Hogg show that this measure correlates well with human judgments for two different scenarios.

2.1.2 Local Articulated Motion

The utility of the preceding trajectory-based representations depends on the assumption that a human can be well approximated as a moving "blob". In some cases, how-

ever, the changing configuration of a person's limbs may be the key to recognising a range of potentially interesting behaviours: it is possible that certain instances of behaviours such as fighting, or tampering with objects in a scene, could be indistinguishable from ordinary behaviour under the moving blob assumption. The reliable recognition of these behaviours requires a more fine-grained description of an individual's motion. While it is difficult to reliably track the positions of an individual's limbs over time, a variety of techniques exist for characterising the patterns of local motion that arise from different types of behaviour.

In an early example of event detection based on articulated motion, Datta et al. propose a technique for detecting violent behaviour in [30]. Their approach involves detecting human silhouettes by filtering the objects identified by a background subtraction algorithm. The bounding box is divided along the vertical axis into three equal sections: the top section is used to identify the location of a person's head and shoulders. The location of each detected person's head is tracked using colour information, while the silhouette is used to estimate the extent and orientation of upper and lower limbs in each frame. These pieces of information are combined to detect fights in a rule-based fashion: when a person's head moves suddenly (this can be quantified by the first derivative of acceleration, known as "jerk") and a second person's limb is extended towards the first person, the frame is flagged as potentially violent. While this representation is shown to be effective, it is likely to be very difficult to accurately identify limb configurations in real-world sequences.

Another way to address this problem is thus to capture local motion patterns without the specific identification and tracking of limbs. One such approach was proposed by Efros et al. in [38], where a descriptor for local motion is formed by measuring optic flow in a fixed window centred on the location of a moving person. The optic flow is split into two channels corresponding to the x and y components of the measured flow, each then split into two further channels corresponding to the (half-wave rectified) positive and negative components of each motion direction. Splitting the motion into positive and negative components allows a Gaussian blur to be applied to each channel without losing information by averaging positive and negative motion along each axis. The motion descriptor thus finally consists of four "blurry" motion channels which provide an approximate description of the optical flow in a given region. Efros et al. determine the similarity of a pair of action sequences (over some small number of frames) using a spatio-temporal cross-correlation measure (summed over the 4 channels); this enables classification of different actions using the K-nearest neighbours algorithm in conjunction with a labelled database of actions, and is shown to effectively distinguish between various types of action occurring in ballet, tennis

and football videos [38].

The local motion descriptors proposed by Efros et al. have also been utilised by Robertson and Reid in [114] for activity recognition in a surveillance context. As in [38], a database of labelled motion descriptors is used to classify new examples; in [114] the database contains examples of walking, running and stationary people. Each descriptor consists of 5 frames (x 4 channels x 20 x 20 pixels) of optic flow information, resulting in a very high dimensional representation. In order to speed up the evaluation of new actions with respect to the database, a probabilistic matching method (proposed by Sidenbladh et al. in [131]) is employed. In [131], PCA is performed on a database and the projection of a new example is used to define a probability distribution over the possible sign configurations of PCA coefficients; this allows samples to be drawn from the database according to a query example, thereby providing a way to estimate the probability of a new example belonging to a given class. In [114], databases of position and velocity information for different activity classes are kept alongside the database of motion descriptors: given a query the most likely velocity/position/action indicated by each database are combined using a Bayesian network to determine the most likely activity underlying a given instance of motion.

Robertson and Reid adopt a similar strategy in [113] for estimating gaze direction for detected faces/heads. Using a color histogram obtained from example skin regions in a given scenario, it is possible to estimate the probability that a new pixel corresponds to skin. A head descriptor is formed by estimating head locations from the motion silhouettes of detected people, weighting the foreground pixels (of detected head regions) according to their skin probability values, and finally rescaling to a uniform size. As in [114], descriptors are concatenated across 5 frames. A labelled database containing a total of 100 example descriptors for each of 8 possible head directions is used to classify new examples, using the PCA-based matching procedure employed in [114], and the estimated head direction is then improved by incorporating a motion direction cue using Bayes' rule. While the estimated gaze direction is not used for any higher level behaviour modelling in [113], it is clear that it could provide a useful cue for behaviour classification in a surveillance context.

The strategy employed by Robertson and Reid in [114, 113] shows how a high-dimensional local motion descriptors can be used to analyse video footage in combination with a labelled database of actions. The key insight is that if it is possible to formulate an efficient matching procedure, then high-dimensional motion descriptors can be used to classify low-level actions in conjunction with a set of labelled examples. In addition to optic flow representation of Efros et al., there are a range of other strategies for classifying action segments which could also potentially be applied to surveil-

lance - it is worth noting, however, that the motion descriptor proposed by Efros et al. is unique in being specifically intended for classifying motion at a distance. A popular strategy in the action recognition literature is to consider the changing silhouette of a moving person as a spatiotemporal volume: several approaches [73, 128, 49] attempt to find salient feature points in the spatio-temporal volume that characterise a given class of motion, while others [70, 95] parametrise and classify spatiotemporal volumes according to the responses of a set of spatiotemporal filters.

Boiman and Irani propose a novel approach to local action classification in [15] which does not rely on tracking moving objects. Instead of attempting to classify actions, the technique proposed in [15] focuses solely on the identification of unusual local motion patterns. This technique works on the basis of a "query" video clip and a database of video footage containing ordinary behaviour: using the database, the "likelihood" of each pixel in the video clip is estimated by quantifying the extent to which its surrounding (spatiotemporal) region can be explained in terms of similar regions in the database. Boiman and Irani decompose each spatiotemporal region into a set of local descriptors at different scales and formulate a probabilistic cost function which can be used to compare regions in the query with those in the database while allowing for discrepancies between the sets of descriptors and their relative locations. Quantifying the likelihood of a query video region requires a computationally expensive search procedure, which is made more efficient in [15] by initially restricting the search procedure to the coarsest level of motion description. Unusual behaviour can thus be detected by thresholding the likelihood function for a given query. This strategy is shown be effective in various scenarios, although it is unclear whether it could be used to detect unusual behaviour in a real-time context.

2.1.3 Scene-wide Motion Patterns

A major weakness of many of the preceding approaches is that they rely on an accurate estimate of the location of moving objects in a scene. In scenarios where tracking is impractical, for example in crowds, a different type of approach to representing video data is required. This section describes a variety of representational strategies which have the capacity to model global patterns of activity in a scene without requiring tracking. Global motion patterns have the additional advantage of capturing multiple activities within a scene - rather than those of a single individual - thereby providing a potential means to recognise interactions.

An early example of this style of representation was employed by Brand and Kettnaker [17] to describe activity on frame-by-frame basis. Firstly, activity in an office scenario is represented with the parameters of an ellipse fitted to the largest connected

region of foreground pixels (detected by a background subtraction algorithm) augmented with information about how the ellipse has changed since the previous frame (in particular, change in centroid and mass). Secondly, activity at a road junction is represented with the set observed of flow vectors larger than a certain magnitude. Without requiring tracking, these representations are used in [17] as the substrate upon which to build sequence models (single or multiple observation HMMs) and are shown to enable the classification of various activities.

Another, more complex, strategy for scene-wide activity representation proposed by Xiang et al. in [169], has formed the basis for a large body of work on automated surveillance [168, 166, 165, 163]. Their approach focuses on instantaneous motion between scenes, and essentially works by identifying foreground objects (using a Gaussian mixture model background subtraction algorithm) and then calculating a "Pixel Change History" representation for these regions: from this a 7 dimensional representation is formed for each detected region in a given frame, consisting of bounding box parameters for the region combined with the x and y moments of the pixel change history. The collection of 7-dimensional instances in a given video is then modelled with a mixture of K (determined by BIC criteria) Gaussians, so that each component corresponds to an "event class". This means each frame can be represented with a K-dimensional vector containing the posterior probabilities for each event having occurred in a given frame. Video footage can thus be represented as a trajectory in K-dimensional space, which can be firstly be used to divide the video data into segments corresponding to behavioural episodes [165], and secondly as the representational substrate upon which to build sequence models for the different classes of behaviour observed.

In contrast to the preceding approaches, it is also possible to describe activity in a scene in an entirely hard-coded fashion: in [178] Zhong et al. generate binary motion images for a sequence of frames by applying a temporal Gaussian derivative filter centred on each frame combined with a spatial smoothing filter. Each frame is mapped to a vector of fixed dimensionality by dividing the image plane into a regular grid, and computing the sum of detected motion pixels in each cell. The vector representations for each frame are clustered using K-means to provide a dictionary of activity prototypes. Zhong et al. then divide video footage into a series of regular segments (length 4 seconds in [178]) whose content is described with a binary vector corresponding to the presence/absence of each activity prototype. This representation is shown to be potentially effective for activity clustering and unusual event detection in different scenarios including videos of poker tables, nursing homes, and street surveillance footage. In a similar vein, Wang et al. encode scene activity by dividing the scene

into a grid of 10x10 pixel regions, and measuring optical flow within those regions. In each region the flow is quantised in terms of four possible directions, meaning that a 480x720 scene is described in terms of 48x72x4 possible motion primitives - an advantage of this representation is that it does not require a preliminary learning step.

Likewise, in [1] Adam et al. adopt a representational strategy that does not require any learning step; their aim is to provide a practical, computationally lightweight solution to unusual event detection. Firstly, optical flow is measured at a grid of fixed "monitor" locations in the image plane: for each location, this is achieved by estimating a likelihood function over different possible local patch displacements between frames, and then choosing the "maximum likelihood" direction and magnitude of motion. The likelihood function - if deemed too flat/ambiguous - can be used as a basis for disregarding measurements altogether. Each frame is thus represented by a set of flow vectors corresponding to the grid of monitor locations. For each monitor a record of the last N measurements is kept, and used to construct a histogram which provides a likelihood measure for new observations. Anomalous events are then detected whenever a certain proportion of recently observed frames contain flow observations whose likelihood falls below a threshold. Although the scene-wide flow patterns calculated in [1] are only used to analyse motion on a local basis, they could potentially be used for modelling global motion patterns.

Optical flow is also employed by Andrade et al. in [3, 2] for the purpose of detecting anomalous events in crowded scenes. In [3], activity within a scene is captured by by measuring optic flow within a regular grid of 8x8 pixel blocks; the resulting set of flow measurements is filtered according to a binary mask produced by a background subtraction algorithm, so that flow measurements outside detected foreground regions are set to zero. This results in a high dimensional parametrisation for each video frame: PCA (Principal Components Analysis) is then applied to the video sequence to find a set of eigenvectors with which to project the flow measurements from each frame into a 10 dimensional space. Video footage, now represented as a 10-dimensional time series, is divided into 4-second segments (as in [178]) which are used to train sequence models and cluster the segments. Flow is extracted in the same way in [2], but modelled in a different way without an intermediate PCA step: an HMM with a mixture-of-Gaussians emission distribution is used to model the changing list of flow vectors measured in the scene, and the same model is trained for each local region of the scene. The flow field representation used in [3, 2] is shown to capture the difference between normal crowd behaviour and events such as falling and sudden overcrowding, albeit for simulated data.

A key advantage of these approaches is that they are able to provide summaries of

video content without solving the difficult problem of reliably detecting and tracking moving objects. However each of these approaches decomposes video footage into short segments, which are typically of an arbitrary predetermined length. It is clear therefore that this type of representation provides a convenient way to sidestep explicit motion representation for the detection/classification of short-term behaviours, but does not provide a means to represent the same longer-term behaviours that can be captured by trajectory data.

2.2 Modelling Behaviour Patterns

Once video data is phrased in terms of the representations described in Section 2.1, it is possible to construct models corresponding to different behaviour patterns, which can then be used for classifying future behaviours and identifying anomalies. This section reviews a variety of techniques for constructing such models. Firstly, Section 2.2.1 reviews techniques for modelling behaviours that are initially represented as continuous/discrete sequences. Then, Section 2.2.2 reviews techniques for discovering clusters of similar behaviour patterns in datasets. Finally, several approaches which represent sets of trajectories with spatial boundaries are reviewed in Section 2.2.3.

2.2.1 Sequence Models

Many of the approaches to video content representation discussed in Section 2.1 yield segments of behaviour represented either as a string of symbols, corresponding to a learned/prespecified quantisation (eg. [83]), or as high dimensional trajectories in some parameter space (eg. [48]). As illustrated in Figure 2.2, the methods described in this section serve two possible purposes: either to model a set of sequences in a probabilistic manner, or to represent individual sequences as a precursor to clustering. Hidden Markov models, discussed first, are unique in addressing both issues.

Hidden Markov Models and their variants

Definition The Hidden Markov Model (HMM) is a generative model which posits a sequence of latent/"hidden" (output-generating) states that gives rise to observed sequences. Each hidden state can give rise to observations according to a discrete (ie. a discrete set of possible outputs with probability table) or continuous (eg. Gaussian, Poisson etc) distribution, with different distribution parameters specified for each state. Output can thus be generated by moving through a sequence of states, generating an observation at each one according to their distribution models, before moving to another state (or staying put). The propensity to change between different

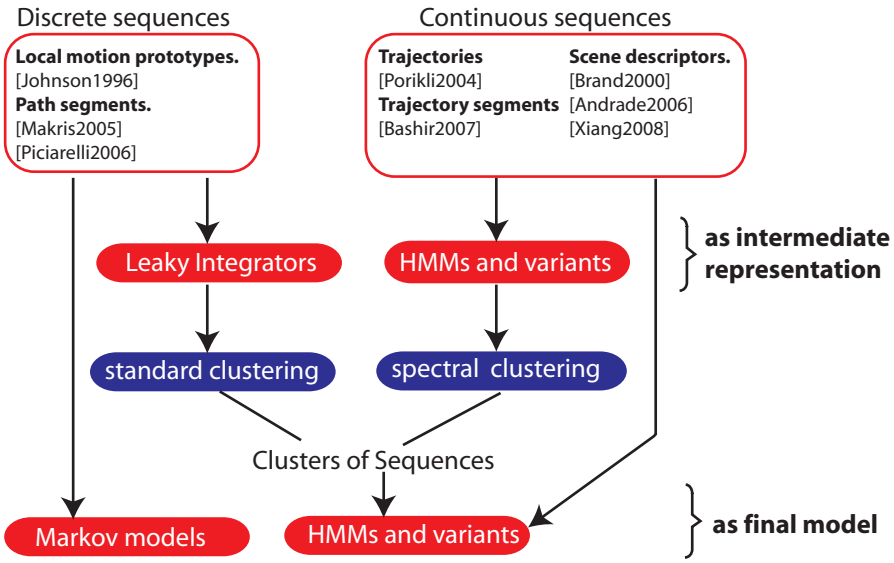


Figure 2.2: The role of sequence models in behaviour modelling.

states is governed by a matrix of transition probabilities T; similarly the propensity for the sequence to start in a given state is specified by a vector of prior probabilities V. Given some assumptions about the number of hidden states and the type of emission distribution, the HMM parameters that best correspond to a given observation sequence (or set of sequences) can be estimated using the Expectation Maximisation algorithm (see Rabiner [107]).

Classification and clustering. HMMs provide a useful tool for sequence classification: Because it is possible to compute the likelihood of a new sequence of observations having been generated by a given HMM, a new sequence can be classified by determining which member of a set of HMMs (corresponding to different behaviours) was most likely to have generated it. In a similar vein, the resulting likelihood values could also be thresholded to identify outliers.

In order to identify clusters in a set of sequences a technique frequently adopted in the behaviour modelling literature is to fit each individual sequence with its own HMM. This provides the ability to measure the similarity of two sequences, by assessing the likelihood of one being generated by another's HMM parameters and vice versa, and combining these quantities in a symmetrised distance function. This principle has been applied for a variety of sequence types including trajectories [104] and scene descriptor sequences [168, 3], typically using the following distance function (where S denotes a sequence and λ a set of HMM parameters):

$$D_{ij} = \frac{1}{2}\{\log P(S_j|\lambda_i) + \log P(S_i|\lambda_j)$$

Having defined a similarity metric, it is then possible to employ similarity-based clustering techniques such as Spectral Clustering (see Section 2.2.2). Once groups of similar sequences have been identified an HMM can be fitted to each group of sequences, to enable classification as described earlier.

Sequence smoothing. Another useful property of HMMs is that they can be used to determine the most likely series of hidden states given a set of observations, which can be used to estimate the "smooth" behaviour sequences underlying noisy observations. The smoothing property of HMMs has been exploited by Robertson and Reid in [114] for providing commentary on video footage of tennis games: an estimate of a person's action at each instance in a short sequence is obtained (using the optical flow based method described in Section 2.1.2), and its likelihood is assessed with respect to a series of hand-defined HMMs corresponding to different behaviours; the HMM that best explains the data is then used to estimate the most likely sequence of actions (which each have an emission distribution defined by a probability table for observable actions) underlying the observed sequence. This effectively imposes a smoothing prior on the observed sequences of low level actions, and is shown to increase the accuracy of action classification in [114]; this technique is also applied to surveillance footage by the same authors in [115].

HMM Trajectory Models Since HMMs can have continuous emission distributions, they are well suited to modelling continuous sequences such as motion trajectory data. Porikli et al. use HMMs for trajectory clustering in [105, 103, 104], where an individual HMM is fitted to each trajectory as a precursor to spectral clustering. The emission distribution for each state is modelled with a Mixture of Gaussians, and it is suggested in [104] that a range of different numbers of states and mixture components are tested in order to choose the configuration that maximises a penalised likelihood function. This effectively means that each HMM state represents a section of a given trajectory, with a model for the distribution of coordinates in that section. A weakness of this representational strategy is that, within a given section (ie. region corresponding to an HMM state), it places no constraints on the ordering of coordinates. For example, the coordinates of a trajectory could be randomly shuffled within sections corresponding to different states of a given HMM with no impact on the resulting model likelihood.

An alternative HMM-based approach, proposed by Bashir et al in [10], addresses this issue by segmenting each trajectory based on significant changes in curvature, and describing each trajectory segment with a parametric representation. (Bashir et al acquire a reduced-dimensionality segment representation by performing Principal

Components Analysis on a set of spatially-normalised/resampled segments - see [9] for details.) Thus, instead of a sequence of 2D coordinates, the trajectory is represented as sequence of coordinates in the dimensions defined by the PCA-based segment representation. Although this approach adopts a normalised segment representation approach - which cannot distinguish between short and long movements of the same shape - it is clear that other representations could be employed. As with the preceding approach, the emission distribution for each state is defined by a Mixture of Gaussians in the segment representation space, but it is clear that the segment-based HMM approach would not assign a trajectory the same likelihood as its shuffled counterpart. Nonetheless, a potential weakness of this approach is the extent to which it relies on a consistent segmentation procedure - if this were to fail misleading results could be obtained. In [10], Bashir et al. construct HMMs for sets of example trajectories corresponding to different classes of motion, and classify new motions by assigning them to the class whose HMM has the highest likelihood.

HMMs for Multiple Independent Observations. In the approaches described so far, each HMM gives rise to a sequence of single (potentially multivariate) observations according to the emission distribution for each state. It may also be desirable to model sequences where multiple observations (which are conditionally independent given a particular hidden state) are made at each time-step: this is addressed by an HMM variant known as the Multiple Observation Hidden Markov Model (MOHMM), which has been applied in a variety of behaviour modelling contexts [168, 3, 17]. Xiang and Gong employ MOHMMs in [168, 166, 165, 163], to model video segments represented by a multi-dimensional trajectory corresponding to typical features in motion history images (as described in Section 2.1.3): in their approach, instead of each HMM state having an emission distribution defined in a high dimensional space, each state has a separate one-dimensional emission distribution for each observed dimension. Andrade et al. employ a MOHMM in the same manner in [3] to model a multidimensional trajectory defined by the PCA components of optic-flow based features (see Section 2.1.3) in order to analyse crowd behaviour. This represents a significant reduction in the number of parameters needed to define the emission distribution and, pertinently, the quantity of training data needed to estimate those parameters.

While both Andrade et al. and Xiang and Gong employ MOHMMs with multiple single-dimensional emission distributions, there is no constraint on the nature (eg. dimensionality) of these distributions for HMMs, save that the observations that they model are conditionally independent given a particular hidden state. Figure 2.3, reproduced from [48], illustrates the difference between HMMs and MOHMMs. It is of

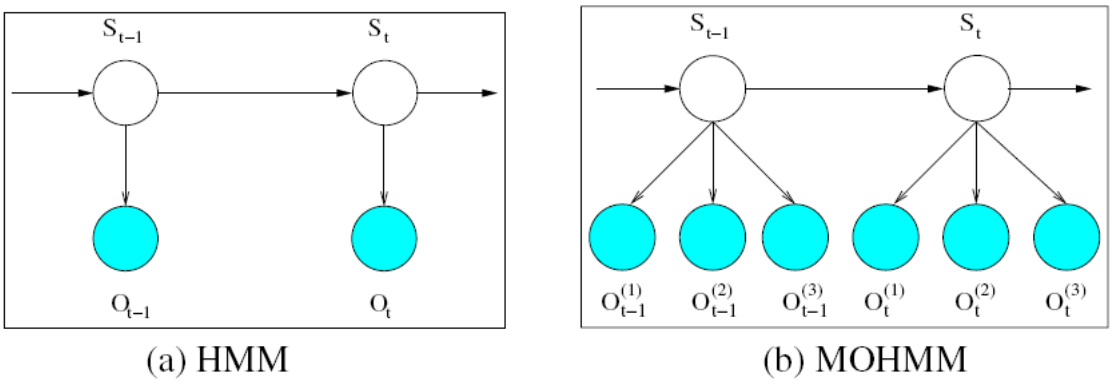


Figure 2.3: HMMs and Multiple Observation HMMs (from [48])

course possible just to use a standard HMM structure if the multiple observations all follow the same emission distribution - this approach is adopted in subsequent work by Andrade et al. [2] where a single (Mixture of Gaussians) emission distribution is used to model a fixed length list of optic flow vectors in each frame. A potential problem is that this doesn't provide a means to distinguish between long and short lists of observations, which may be important in the case of behaviour analysis - Brand et al. encounter this problem in [17] and address it by adding a second emission distribution corresponding to the number of observations (modelled with a log-normal distribution). This model, termed the Multiple Observation Mixture and Counter Hidden Markov Model (MOMC-HMM), is used in [17] to model lists of flow vectors observed in a traffic scenario.

HMMs for Multiple Dependent Observations While the MOHMM structure provides a convenient way to model parallel independent events with a single latent cause, it may conversely be desirable to model the causal links between different classes of event in order to capture the influence that a given event has on the likelihood of observing other events in the next frame. If patterns of scene-wide behaviour are being modelled, it is likely that capturing interactions between local events may provide a powerful cue for distinguishing between different classes of global events. While it is possible to capture patterns of interaction with the MOHMM structure, it is very inefficient (requiring a different state for every distinct configuration of events.) A better alternative is to posit several distinct chains of states (corresponding to each type of event) where each state in a given chain depends on both its own previous state and those from other chains: this structure, proposed by Brand et al in [18] is known as the Coupled Hidden Markov Model CHMM. It has been employed by Oliver et al. in [97] to model different types of interactions between people based on trajectory

features: CHMMs are constructed with two connected chains of hidden states, each corresponding to the trajectory of an individual. Representing each class of interaction with a CHMM is shown to provide better classification accuracy than a standard HMM.

Gong and Xiang explore the use of CHMMs in [48] to model the behaviour occurring in a loading bay scenario, with several interconnected chains of hidden states corresponding to different event classes. In this case the model is being used to derive smooth state sequences (as Robertson and Reid do in [114]) from noisy data - the CHMM structure is shown to give greater accuracy than a MOHMM trained on the same data. Gong and Xiang also propose a variant of the CHMM where, rather than being fully connected, a sparse set of connections between chains is derived. The structure of this model, termed the Dynamically Multi-Linked Hidden Markov Model (DML-HMM), is obtained by training models with different connectivity patterns and choosing the model which maximises a penalised likelihood function, the Bayesian Information Criterion (see [12, 20]), which specifies a trade-off between model likelihood and the number of parameters used. The DML-HMM is shown to perform equivalently to CHMMs in [48], with the added advantage that the resulting structure reveals the salient causal connections between events.

Markov models for prototype sequences

Many of the techniques for representing video data discussed in Section 2.1 involve the construction of a 'prototype' alphabet, typically by identifying clusters in a set of extracted features. Video sequences can then be represented as prototype sequences: in many cases, if prototypes are extracted in a noisy/unreliable fashion, HMMs provide an ideal tool for modelling their sequences. However, if the extraction of prototypes from video data is reliable and consistent, several other techniques can be employed, reviewed as follows.

The geometric path models proposed by Makris and Ellis in [82] provide an example of a reliable, albeit coarse-grained, prototype for representing motion in video data. This approach (described in Section 2.2.3) generates a set of models for the typical routes taken through a given scene, which can then be used to classify new trajectories. The same authors extend this approach in [83] where the set of learned path models (along with entry/exit and 'stop' regions) is represented with a topological network, as illustrated in Figure 2.4. A similar approach is proposed Piciarelli in [101], leading to a network of trajectory segments for a given scene (see Section 2.2.3). In both cases, trajectories can be described as traversals of a topological network. Given a set example traversals of the network, it is very simple to estimate local transition

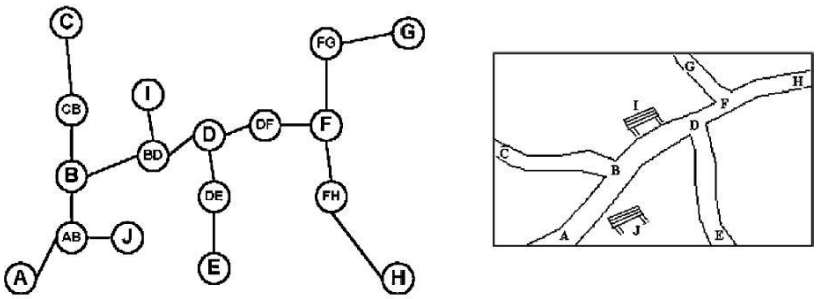


Figure 2.4: Topological representation (left) of scene (right) [83].

probabilities for each node in the network, by simply calculating the proportion of transitions that have been made to each adjacent node as follows: $p_{ij} = \frac{N_{ij}}{\sum_k N_{ik}}$. The combination of the scene network with local transition probabilities corresponds to a Markov model with directly observable states: this provides a means to estimate the likelihood of different network traversals, and thus enables unusual instances to be identified.

One useful property of the preceding approach is that it can be used to predict the next state (ie. path/region to be entered by a pedestrian) based on the set of transition probabilities for the current state. In some cases, however, accurately predicting the next observation may require a number of previous observations rather than just one. This issue has been addressed in [46], where Galata et al. propose a strategy for modelling prototype sequences using Variable Length Markov Models (VLMMs) [116]. Like all Markov models, VLMMs contain a complete set of first order state transition probabilities; however, these are augmented with transition probabilities for a selection of distinct observation sequences which are found to provide a better prediction of the next observation than any shorter sequence (quantified using the KL divergence). Thus, when given a sequence of observations, the longest stored sequence which matches the most recent observations is used to predict the next state. Galata et al. use VLMMs to model articulated motion: this is encoded with a series of prototypes, for which a subset - termed 'key prototypes' - correspond to significant transitions in behaviour. VLMMs are constructed to capture the typical transitions between key prototypes, and are shown to provide better predictions than a simple first order Markov model [46]. When applied to transitions between prototypes corresponding to behaviour in a scene, it is possible that this model could provide a useful means to detect unusual behaviour.

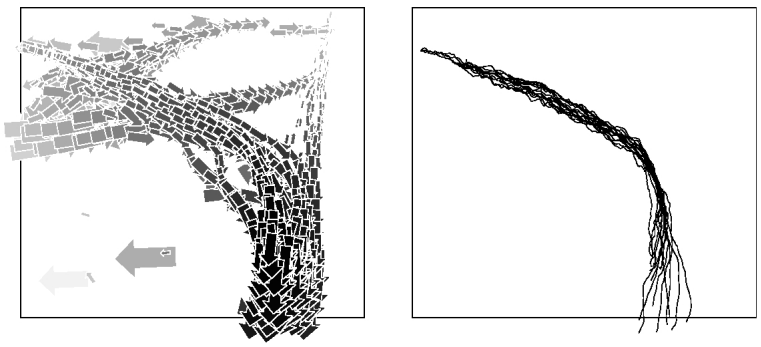


Figure 2.5: Motion Trace Prototype and Corresponding Trajectories [64]

Leaky integrator sequence parametrisation

A novel sequence representation strategy, based on leaky integrators, was proposed by Johnson and Hogg in [64] for modelling trajectories encoded as sequences of local motion prototypes. In order to represent the prototype sequences corresponding to different behaviours with fixed-length vectors, Johnson and Hogg proposed a neural network architecture in which each prototype has a corresponding leaky integrator neuron. As a sequence of flow vectors is observed, the activation level of each neuron in the model is increased whenever its corresponding prototype is observed, otherwise decaying exponentially until new input is received. The set of activation levels in the model therefore provides a "time delayed trace" which captures the recent history of observed prototypes. In [64] a set of 1000 leaky neurons is used to capture prototype activation sequences, which effectively means that the network parametrises each prototype sequence with a 1000 dimensional vector. These 1000 dimensional vectors are then clustered to yield a set of typical trajectory models (ie. cluster centres), which can be used to classify new vectors of prototype activations using a nearest neighbour approach. Figure 2.5 illustrates the set of prototype activations for a given cluster and the corresponding training data. Since it converts a sequence of prototypes to a high dimensional vector, this approach blurs the boundaries between parametric trajectory representations (as described in Section 2.1.1) and sequence models, enabling prototype sequences to be clustered using standard techniques.

2.2.2 Identifying Behaviour Patterns

Section 2.1 provided a variety of ways to represent behaviours or segments of behaviours observed in video data. The sequence models discussed in Section 2.2.1 provide one way to model classes of behaviour patterns. However, a necessary prerequisite for behaviour classification - in the absence of pre-labelled data - is the identifi-

cation of the different classes of behaviour present in a dataset. This section reviews a number of different techniques that have been used to identify natural groupings in collections of parametrised behaviour and, where applicable, the ways in which these can be used to determine whether new examples are anomalous. In several cases, the groups obtained by clustering a dataset can be used to provide a discrete "codebook" with which to represent new items of data.

Clustering Parametrised Behaviour

Many behaviour clustering strategies employ an algorithm known as Competitive Learning (see [119, 54, 71]): this works by initialising a random set of cluster centres, and incrementally adjusting these cluster centres as training data is presented. At each step, a data point is assigned to its closest cluster centre, and the cluster centre is moved towards the data point by some small amount. After multiple presentations of a dataset, each cluster centre accounts for a collection of similar items within the dataset (those which are not closer to any other cluster). Johnson and Hogg use this approach in [64], both to construct a codebook of typical motion vectors, and to identify salient clusters in a high-dimensional representation of trajectories. For improved codebook generation, Johnson and Hogg alter this algorithm by limiting the number of examples each cluster can account for [64].

It is worth noting that Competitive Learning is essentially an incremental variant of the K-means clustering algorithm [80, 12], which iteratively adjusts cluster centres on the basis of multiple presentations of the *whole* dataset, rather than single examples. However, a key advantage of the incremental learning procedure is that it reduces the computational cost of clustering large datasets (as noted by Stauffer and Grimson, who also create a motion vector codebook using this technique in [136]).

A popular variant of Competitive Learning is the Self Organising Map (SOM) algorithm proposed by Kohonen in [71], which adds the constraint that cluster centres are represented in a 2D space or "map". In this algorithm, rather than just modifying the closest cluster centre for each data point, cluster centres with adjacent projections on the 2D map are also modified. The extent of this additional modification depends on a neighbourhood function, which typically decreases exponentially with distance (in the 2D space) from the chosen cluster. This process can be regarded as constructing a non-linear manifold in the input feature space [53, 71]. SOMs have been used to cluster parametrised behaviours in a number of approaches: In [98] Owens and Hunter use a SOM to cluster parametrised trajectory segments (see Section 2.1.1); more recently, Naftel and Khalid used a SOM to cluster whole trajectories represented with DFT coefficients in [90] before modelling each cluster with a Gaussian distribution.

The Expectation Maximisation (EM) algorithm provides a more direct means of modelling the clusters in a dataset with Gaussians. For a given number of Gaussian components, this approach maximises the likelihood of the model parameters (ie. means/cluster centres, covariance matrices, and weights) given the dataset, by iteratively repeating the following steps. Firstly, the "E-step" calculates the relative contributions of each component to the probability density at each data point (the "component responsibilities"). Secondly, the "M-step" re-estimates the parameters of each component, weighting the contribution of each data point by the corresponding responsibility values. An important feature of this algorithm is that is guaranteed to converge to a locally optimal solution [53, 12]. Xiang and Gong use EM to fit a Gaussian mixture model to a large set of parametrised instantaneous motion patterns (see Section 2.1.3). Having chosen the number of components according to the Bayesian Information Criterion (which specifies a trade-off between model likelihood and the number of parameters - see [20, 53, 12]), the resulting Gaussian components accurately reflect clusters of typical low-level actions in the scene.

In a series of trajectory modelling approaches proposed by Hu et al. [61, 60], the "fuzzy K-means" algorithm is used to cluster parametrised trajectories. As with the SOM algorithm, this is an incremental procedure where each new example modifies multiple cluster centres. However, the extent of this modification is determined by a "fuzzy membership function" which assigns values inversely proportional to the distance of each cluster from the new example. The "fuzzy membership" values are conceptually equivalent to the component responsibilities computed when using EM to fit a Gaussian mixture model. In [60] Hu et al. propose a two stage trajectory clustering technique: firstly, resampled versions of trajectories are clustered to reveal spatially similar sets; the original trajectories belonging to these sets are then clustered for a second time to reveal subsets with similar spatiotemporal profiles. Hu et al. assess the quality of clustering obtained for different numbers of clusters using the Tightness and Separation Criterion [170], which is a "fuzzy" equivalent of the standard within-vs-between class scatter ratio (see [45, 12]), but has the advantage (for high dimensional data) of not requiring the calculation of covariance matrices.

Similarity-based Clustering

Some forms of data cannot easily be represented with vectors of fixed length, which renders the preceding clustering algorithms inapplicable because they rely on the calculation of Euclidean distances, means and covariance matrices. However, even if a certain type of data cannot be represented with fixed-length vectors, it is still often possible to define a metric which quantifies the similarity of two data points with a

scalar value. In this light, a technique known as Spectral Clustering [127, 94] provides a way to identify clusters based on the affinity matrix (containing the similarity value for each distinct pair of examples) calculated for a given dataset. Spectral clustering works by calculating the eigenvectors of the affinity matrix, and selecting those with the N largest eigenvalues. Each item of data is then represented with a vector of length N, corresponding to its entries in the chosen eigenvectors; these vectors can then be clustered using K-means. In short, this technique allows the reconstruction of a parametric representation for a given item of data, based on the structure of the affinity matrix.

Spectral clustering has been used to identify groups in several different types of behavioural data. In [105], Porikli and Haga use spectral clustering to identify groups of similar trajectories, after defining an HMM-based trajectory similarity metric (see Section 2.2.1 for details). Andrade et al. adopt a similar approach in [3] where optic flow sequences extracted from a set of 4-second video segments are modelled with Multiple Observation HMMs, which are used - as in [105] - to define a similarity measure for the video segments. Andrade et al employ a spectral clustering algorithm proposed by Zelnik-Manor and Perona in [175], which provides a method for estimating the number of clusters present in a dataset. Similarly, in [168, 167], Xiang and Gong model video segments using MOHMMs and, again, define a likelihood based similarity measure in order to perform spectral clustering. However, Xiang and Gong propose a novel spectral clustering technique in [167] where, rather than representing the dataset with corresponding elements in the largest eigenvectors (ie. those with largest eigenvalues) of the affinity matrix, a set of the most "relevant" eigenvectors (those which best separate the dataset into clusters) is chosen.

In a similar vein, in [66, 67] Junejo et al. represent a trajectory dataset with a fully connected weighted graph, where each edge weight corresponds to the similarity of two trajectories (quantified by the Hausdorff distance): this enables clusters to be found by recursively partitioning the graph according to the edge weights. To achieve this, Junejo et al. employ the Normalised Cut algorithm proposed by Shi and Malik in [130], which aims to partition graphs into maximally dissimilar sets of nodes, while maximising the similarity of nodes within each set. It is interesting to note that this procedure recursively splits the dataset using the second-largest eigenvector of the affinity matrix, and is thus inherently similar to spectral clustering.

An interesting extension to similarity-based clustering is proposed by Wang et al. [157] in the context of trajectory clustering, where each similarity value calculated is accompanied by a measure of confidence which ranges between 0 and 1. In [157], a trajectory similarity measure S measures the distance between two trajectories at the

points where they are closest, while a confidence measure C determines how often the trajectories actually coincide. Wang et al. propose a way to incorporate confidence measures into an aggregate similarity measure, $W_{ij} = \frac{S_{ij}^{C_{ij}}}{(1-S_{ij})^{C_{ij}}+S_{ij}^{C_{ij}}}$, which ranges between $\frac{1}{2}$ in the case of no confidence, and the true similarity value S_{ij} in the case of full confidence. This provides a single affinity value which enables spectral clustering to be employed, while taking into account the confidence values.

Clustering Bags-of-Prototypes

The preceding clustering approaches are only applicable for behaviour representations which can be parametrised and/or used to define a similarity measure. However, many of the video representation strategies described in Section 2.1 represent behaviour with a sequence/set of low level prototypes. The sequence modelling techniques described in Section 2.2.1 provided one way to model/compare different sequences. Another effective strategy is to ignore the ordering of the prototype sequences, and to attempt to extract behaviour patterns based on the co-occurrence of prototypes within a given sequence: an advantage of this approach is that it relaxes the need for perfect tracking, and could potentially facilitate the classification of incomplete sequences.

One of the first examples of this approach was proposed by Stauffer and Grimson in [136] where behaviours are represented as sets of prototypes (obtained by clustering local motion/object size vectors). Given a large set of sequences, Stauffer and Grimson provide a method to identify a set of typical behaviours based on the cooccurence statistics for prototypes within each sequence. Given a large set of sequences, a model is created consisting of a cooccurence matrix (which encodes the relative frequency of each distinct pair of prototypes) and a vector of prior probabilities for each prototype. Using the fact that it is possible to express an estimate for the cooccurence matrix in terms of the vector of prior probabilities, a two component model (consisting of two vectors of prior probabilities) is formulated. The two prior distributions that provide the best approximation to the original cooccurence matrix are found using a gradient descent procedure, and used to split the cooccurence matrix/prior vector to form two new models. This procedure is applied recursively, yielding a tree-like decomposition where the leaf nodes contain models corresponding to distinct behaviours: new sequences can then be easily classified on the basis of the frequencies of their constituent prototypes and the prior probability vectors from each behaviour model.

Another strategy for behaviour clustering on the basis of cooccuring prototypes is proposed by Zhong et al. in [178], where an approach inspired by document clus-

tering is adopted. Each frame of video data is represented with a vector containing a coarse approximation of motion in the scene (see Section 2.1.3), and these vectors are clustered to provide a set of prototypes. Video data is then divided into 4-second segments (analogous to documents), which are represented with a binary vector indicating which prototypes (analogous to words) were observed in each segment. A binary cooccurence matrix is then constructed by containing the binary vectors for each video segment, together with a similarity matrix for the prototypes (calculated using a Chi-squared statistic). These matrices are combined to form a large affinity matrix encoding the similarity both between video segments and prototypes, and between prototypes (an identity matrix occupies the portion of the matrix corresponding to segment-segment similarities). As in the case of spectral clustering, the eigenvectors of this matrix yield a parametric representation (termed the "co-embedding coordinates") for each segment and prototype. The coordinates corresponding to video segments are then clustered using K-means to reveal sets of similar video segments.

In [155] Wang et al. cluster sets of motion prototypes occurring in video segments using a variant of the Hierarchical Dirichlet Process model proposed in [147]. Video data is split into 10 second segments, each represented with a collection of prototypes corresponding to local motion vectors. The Hierarchical Dirichlet Process is a generative graphical model which describes a distribution over the different possible collections of typically cooccuring words - or "topics" - in a set of documents. Given a set of documents, sampling techniques can be used to discover the instantiation of this model which best describes the data, leading to a generative "mixture-of-topics" model for the dataset. An important feature of the HDP model is that it includes a distribution over possible topic distributions, which allows the number of topics to be estimated from the data. Since the original HDP model describes a single class of document, Wang et al. propose an extension allowing multiple document models to be learned, enabling a set of different behaviour patterns - each characterised by different mixtures of low level topics - to be identified. A weakness of the model proposed in [155] is that it requires the number of behaviour clusters to be prespecified. In this light, the same authors extend this model in [154] to identify both the number of topics and clusters, in this case modelling prototypes occurring within trajectories rather than video segments.

Generative Local Motion Models

Another way to model video data described with local motion vectors is to model the underlying probability density function of all the vectors observed. Johnson and Hogg adopt this approach in [65] with the aim of synthesising motion trajectories

on the basis of a collection of real observed trajectories. They estimate a probability distribution for local motion transitions, given a history of the last 3 locations:

$$P(\Delta x_t, \Delta y_t | x_t, y_t, x_{t-1}, y_{t-1}, x_{t-2}, y_{t-2})$$

This can be used to generate synthetic trajectories in two different ways: either by applying gradient ascent at each step to to reveal the most probable trajectory, or randomly sampling from the distribution at each step to generate one of many possible trajectories. The first step in estimating the conditional distribution is to fit a Gaussian mixture model to the set of previous observations, yielding the joint probability density function:

$$p(\Delta x_t, \Delta y_t, x_t, y_t, x_{t-1}, y_{t-1}, x_{t-2}, y_{t-2})$$

This mixture can then be conditioned on the most recent three locations, yielding another mixture model approximating the conditional probability distribution, which can then be sampled as described. Using this technique, Johnson and Hogg were able to synthesise a large set of realistic trajectories, on the basis of local motion observations from 15 real trajectories.

A similar strategy for motion modelling was recently adopted by Saleemi et al. in [121], for the purpose of trajectory analysis in a surveillance context. From a set of trajectory data, Saleemi et al. extract vectors corresponding to motion transitions over time intervals of up to 5 seconds, recording every instance

$$f = \left(x_{(t+\Delta t)}, y_{(t+\Delta t)}, x_t, y_t, \Delta t \right)$$

where $0 < \Delta t \leq 5$. The probability density function for these observations is estimated using Kernel Density Estimation, taking a uniformly weighted sum of the contributions from a set of Gaussian kernels centred on each observation (see Chapter 3 for further details). Saleemi et al. propose a method to predict the most likely motion transition from a given location, by sampling from the estimated distribution using the Metropolis Hastings algorithm (see [4, 79]) with a Gaussian proposal distribution centred on the current location. As with the previous approach, future trajectories can be estimated by iteratively predicting the next location on the basis of the previous estimate. A method for anomaly detection is also proposed, by comparing each location in a trajectory with a prediction based on previous locations. The mean/variance of the set of samples used to generate each predication allows the Mahalanobis distance to be used to quantify the discrepancy between real/predicted observations: thus, for a given trajectory, the mean discrepancy value can then be thresholded to detect anomalies.

2.2.3 Spatial Path Models

Several behaviour modelling approaches use raw trajectory data to define 2D regions in the image/ground plane corresponding to the typical paths taken through a scene. The essential characteristic of these models is that they provide a way to represent groups of tracks by explicitly encoding their spatial extent. These models can be used to classify new trajectories but, unlike the sequence models discussed in Section 2.2.1, do not pose any constraints on the ordering of a coordinate sequence. However, as noted in Section 2.2.1, networks of path models can be used to form the basis for Markov models encoding the possible motion transitions within a scene. Many examples of geometric path modelling [82, 41, 101] use a competitive learning procedure to simultaneously identify clusters of trajectories and their corresponding path models, while others [66, 157] serve to represent clusters of trajectories obtained through other means.

One of the first path modelling approaches was proposed by Fernyhough et al. in [41], where the silhouettes of tracked vehicles are used to define a binary mask of occupied pixels for each trajectory, combined with a sequence of spatial coordinates sampled at regular temporal intervals. Motion masks with significant overlap (at least 80% of pixels) are summed to form spatial regions where each pixel corresponds to the frequency of observed paths that occupied it. For each set of trajectories corresponding to a given motion mask, those whose coordinates - which are sampled at regular intervals - coincide within a certain "tolerance radius" are grouped and averaged: these coordinate sequences are then used to divide each motion mask into a series of sub-regions which are used for symbolic reasoning about object behaviour.

Makris and Ellis propose an alternative strategy in [82] where paths are defined by a series of "nodes" constituting an average trajectory, combined with a bounding envelope (see Figure 2.6). In an incremental learning procedure, each new trajectory is linearly interpolated and resampled before being compared to the set of existing routes. In the absence of any similar existing path models, a new model is instantiated; alternatively, if an existing route is sufficiently similar to the new trajectory, its nodes are updated. The bounding envelope defining the spatial extent of the path is defined by vectors perpendicular to the average trajectory, calculated at each node. The extent to which a trajectory falls outside a given bounding envelope is used to define a distance metric between routes and trajectories, which is used to determine whether new trajectories should update existing routes or initialise new ones.

The path model proposed by Makris and Ellis is used as the basis for a network model of a scene [83], as described in Section 2.2.1. In a similar vein, an online method for simultaneously estimating a set of path models and a corresponding network

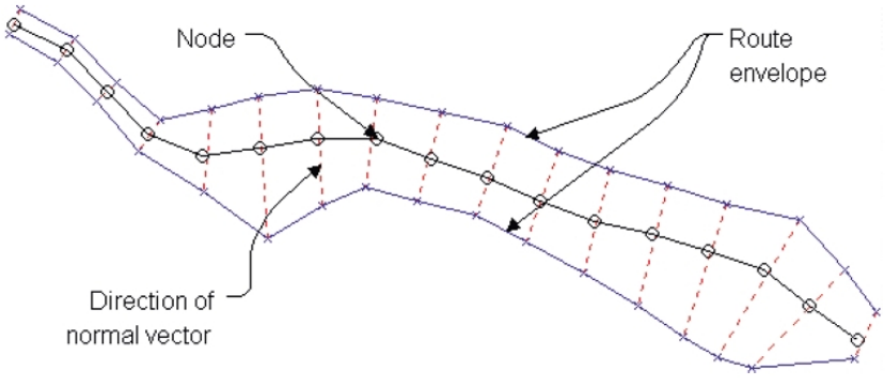


Figure 2.6: Spatial Path Model [82]

structure is proposed by Piciarelli and Foresti in [101]. In their approach, each path model is represented by a mean coordinate sequence, where each coordinate has an associated covariance measure. As in [82], new trajectories that are sufficiently close to an existing path model are used to update it - however, in this case the matching of a new trajectory to its closest cluster occurs in an online fashion. While the distance of a trajectory to the path model (normalised by the covariance measure for each coordinate) lies below a threshold, the corresponding coordinates in the path model are updated. However, if the threshold is exceeded, the path model is split into three sections: a common prefix segment (where the new trajectory agreed with the existing route) and the two possible routes which can be taken after this point. The recursive application of this procedure allows a tree of sub-clusters to be generated, for which transition probabilities can then be calculated to yield a Markov model for the scene.

In several other approaches, path modelling occurs as a final step after trajectory clustering. In [66, 67] Junejo et al. identified groups of similar trajectories using a spectral clustering algorithm, quantifying trajectory similarity using the Hausdorff distance. The groups of trajectories were then used to create path models using a "route + envelope" approach similar to [82]. However, instead of resampling all trajectories to enforce a common coordinate system, the Dynamic Time Warping algorithm (see eg. [122]) is used to identify sets of corresponding points for the calculation of the average trajectory and path envelope. Wang et al. also group trajectories using spectral clustering in [157], but identify path boundaries by thresholding a nonparametric density estimate of the coordinates occupied by each set of trajectories. In both [66] and [157] each path model is augmented with models characterising the distribution of velocities (and trajectory curvature values in [66]) that have been observed within its boundary.

2.3 How is Anomalous Behaviour Detected?

The objective of many of the behaviour learning algorithms discussed is to provide a means to detect behaviour worthy of scrutiny/attention in a surveillance context, frequently termed "anomalous behaviour". There are two types of strategy that can be adopted to detect such behaviour, reviewed as follows.

Deviation from a learned normal behaviour model

Noting that a comprehensive definition of every possible anomalous behaviour is generally an unrealistic prospect, the most widely adopted strategy for classifying new behaviour is to quantify the extent to which it deviates from a model of normal behaviour. This can then be used to rank behaviours according to their anomalous nature, or to provide a hard normal/anomalous classification boundary in conjunction with a user-specified threshold. This principle can be manifested in a variety of different contexts, reviewed as follows.

Distance to nearest cluster. Behaviour classification can be phrased in geometric terms using path boundary models (see Section 2.2.3): in [66] trajectories are marked as anomalous if more than 10% of their points lie outside the boundary of their closest model. This intuitive principle is more commonly manifested in the context of abstract parametric representations. In [64] typical paths are represented with clusters of high dimensional vectors: within a given cluster the Euclidean distances between each member and the cluster centre are modelled with a Gaussian distribution, and new instances are classified as anomalous if their distance lies more than 3 standard deviations outside this distribution. Similarly, in [60], distances within clusters of motion vectors are modelled using an exponential distribution, with an outlier detection threshold determined by the furthest cluster member.

Deviation from Gaussian distribution. A more common approach is to directly model the distribution of parametrised behaviours. In [66] distributions of velocities and trajectory curvatures within a path region are modelled with Gaussian distributions: in each case the Mahalanobis distance is used determine to whether a new instance is anomalous, according to a user specified threshold. Similarly, in [121], a Gaussian distribution is used to model a set of predictions for an object's next location sampled from a nonparametric model, so that deviations can be quantified using the Mahalanobis distance. An alternative approach to quantifying deviation is proposed in [90] where clusters of parametrised trajectories are modelled with multivariate Gaussians:

to detect anomalies, instead of simply thresholding the Mahalanobis distance of new examples, Hotelling's T^2 statistic is used to test whether the cluster belongs to the same distribution as the new instance. However, since a specific significance level is required for rejecting the null hypothesis, the specification of a classification threshold is not sidestepped.

Deviation from nonparametric distribution. Various approaches model distributions with nonparametric techniques. In [157] coordinates occupied by members of a trajectory cluster are modelled using Kernel Density Estimation - a threshold on this density function indicates whether an object's current position is anomalous. A related technique is adopted in [1], where simple histogram models are used to monitor the distribution of optic flow in local image regions, as with KDE the density of new instances can be calculated and thresholded. In a different vein, in [15] a nonparametric probabilistic matching procedure is used to compare a local motion descriptor with a database of examples, providing a likelihood value which can be thresholded to detect unusual instances. Non parametric Bayesian topic models are used to identify cooccuring sets of motion prototypes in [155, 154], and could also be used to assign likelihood values to new examples given a dataset.

Sequence model likelihood. Many of the behaviour models discussed describe sets of learned behaviours in terms of generative sequence models such as HMMs (see Section 2.2.1). Typically a set of these models is used to represent several clusters of sequences (eg. of coordinates, scene descriptors, etc) representing normal behaviour. The likelihood of each model can be assessed with respect to a new sequence, to identify the one that best explains it. Several approaches identify anomalies by placing a threshold on the likelihood of the closest model [3, 17], a weighted mixture of all models [168], or state transitions for a scene-wide Markov model [83, 101]. Alternatively, in [2], where HMMs are used to evaluate a succession of very short (1 second) sequences, anomalies are detected on the basis of a sudden drop in likelihood. For evaluating longer sequences, a useful property of latent sequence models such as HMMs is that they allow partial sequences to be evaluated: in [168] a moving average of changes in likelihood is maintained as each sequence unfolds; a threshold on this value then allows anomalies to be detected before the whole sequence is observed.

Offline analysis. The preceding approaches to anomaly detection enable new instances to be classified on the basis of a model. A small number of approaches aim to identify anomalies in a training dataset as part of the model learning process. In [105] a similarity matrix is calculated so that groups of behaviour can be identified

using spectral clustering; to identify anomalies, a "conformity score" is calculated for each instance by averaging its corresponding column/row of the similarity matrix - a low value indicates that a given instance is unprecedented in the dataset and may be anomalous. In a similar vein, in [178], the clusters of video segments resulting from a spectral clustering process are assessed based on their similarity to other clusters. An "inter-cluster similarity score" (a sum of all pairwise similarities between members of two clusters) is thresholded so that clusters with no resemblance to other clusters - and thus assumed to correspond to unusual events - can be identified.

Classification using anomaly models.

A small number of approaches make use of specific information about the types of anomalous behaviour that are to be detected, either in the form of anomalous training examples, or a rule-based definition.

Anomalous training examples. Given a set of labelled normal and anomalous behaviours, an intuitive approach is to train a discriminative classifier. In [88] vectors (which describe a pedestrian's propensity to approach cars as they walk through a scene) are hand labelled as either normal or unusual, and a simple perceptron is trained to distinguish between the two classes. Since this representation is limited to a very specific feature, it is possible to characterise an adequate classification boundary without a needing an extremely large number of anomalous instances.

An alternative approach, adopted in [166], is to assume that a certain prespecified proportion of clusters in an unlabelled dataset correspond to anomalous activity. In [166], clusters are ranked according to the number of examples they represent, and it is assumed that those containing fewer examples are more likely to be anomalous. The clusters are divided on this basis into two (normal and anomalous) sets, which are each represented with a mixture of probabilistic sequence models. The likelihood of each new behaviour can then be calculated for the two mixture models, and anomalous behaviour can be detected by thresholding the ratio of these likelihood values. Depending on the chosen threshold, this does not necessarily restrict the class of anomalies that can be detected, since new unprecedented anomalous instances may be detectable on the basis of having equal likelihood under the two models.

Rule based approaches. It is also possible to define rules to identify certain types of anomalous behaviour. A very specific rule based definition of anomalous behaviour is adopted in [30], where articulated motion is detected from human silhouettes and new instances are analysed in conjunction with a set of hard-coded conditions indicative

of fighting (see Section 2.1.2), resulting in a binary fighting/not fighting classification.

A more general rule is adopted in [32] where trajectories are assigned a single score quantifying how circuitously a person reaches their destination (see Section 2.1.1). The rule implicit in this approach is that normal trajectories are direct and efficient, or conversely that anomalous trajectories are circuitous and inefficient. The resulting score can be used to rank/classify new trajectories according to this rule, correlating well with human judgments.

2.4 How are the Models Trained?

This section provides an overview of the different training strategies implicit in the behaviour modelling approaches discussed thus far. There are two main stages where training data can play an important part in generating a behaviour classifier: the first is the acquisition of the feature space in which to represent video data, and the second is the building of behaviour models from the video content representations acquired in the first stage.

Stage 1: Feature Extraction/Representation

Main strategies. Video data can be processed in a variety of ways, ranging from object detection and tracking to the extraction of coarse scene descriptors based on optic flow. In order to cluster/model behaviours it is necessary to map these extracted features onto a tractable representational substrate, as described in Section 2.1. Figure 2.7 illustrates the three different approaches that can be adopted: using raw feature data directly (eg. for similarity-based clustering); reducing/fixing the dimensionality of the extracted features, or representing the extracted features with prototypes (ie. instances of a discrete alphabet). Clearly no learning step is required in the case of raw feature data; it is also possible to pre-define a prototype representation for certain types of data (eg. by binning motion vectors according to a predefined scheme) and, similarly, several hard-coded dimensionality reduction strategies (ie. those which do not require additional data) exist for motion trajectory data. However, many approaches rely on an initial training stage where a feature representation is derived from a dataset, discussed as follows.

Prototype representation. Prototype representations for a given feature can be obtained by applying clustering algorithms (see Section 2.2.2) to all available instances of the feature in question, allowing future instances to be represented by their closest cluster centres. Most approaches discussed use an incremental K-means or Self-

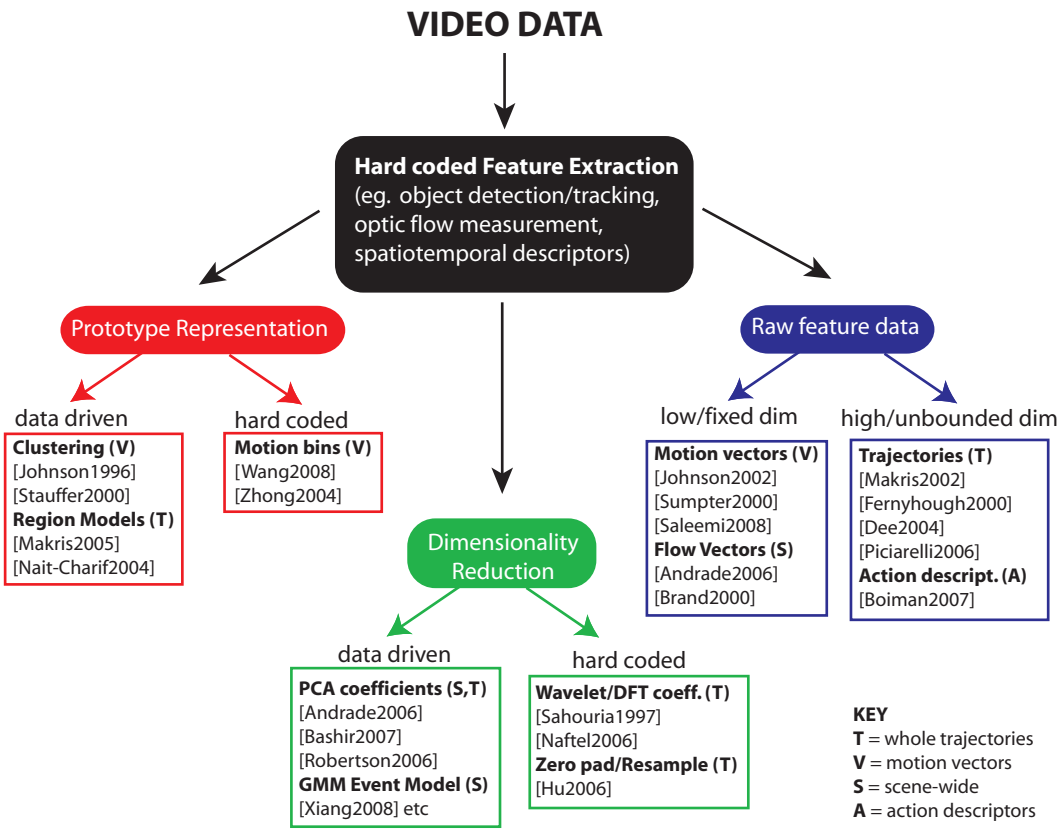


Figure 2.7: Strategies for parametrising video data.

Organising Map algorithm, with an arbitrary - user specified - number of cluster centres. Clearly, the chosen number of clusters/prototypes will have an impact on the fidelity of the resulting representation to the underlying data, and will therefore affect the performance of subsequent behaviour modelling steps.

To apply prototype clustering algorithms effectively requires an initial cross-validation procedure - in some cases this can be achieved without reference to subsequent modelling stages. Various heuristics exist for assessing clustering results (eg. the Tightness and Separability Criterion [170]); alternatively, clustering by fitting a Gaussian mixture model allows the number of clusters to be determined by formal criteria (eg. Bayesian Information Criterion [12]) for assessing the trade-off between model complexity/likelihood. Since several papers do not address the model-selection problem (instead, adopting arbitrary numbers of prototypes), it is possible that the quality of a feature representation may be better judged on the basis of the final high-level model, making cross validation a more complex/costly proposition.

Dimensionality reduction. The most common technique for dimensionality reduction is Principal Components Analysis. Since it is possible to either pre-specify the resulting dimensionality or the proportion of data variation to be accounted for by the representation, the only requirement of this technique is that a representative set of data is available. Prototype representations can also be used for dimensionality reduction: in [168] a Gaussian mixture model is fitted to the features observed in a large set of video frames, so that each video frame can be represented by a vector of probabilities indicating the likelihood that each prototype (ie. feature class represented by a Gaussian component) has been observed.

Learned vs pre-specified. The principal advantage of using a feature representation extracted from a dataset is that it is more likely to accurately reflect the subtle modes of variation in that dataset than a pre-specified strategy. However, it is conversely possible that subsequent novel instances of data could be inappropriately mapped onto the learned set of prototypes/PCA subspace. In this light, one argument in favour of hard-coded representations is that they can provide a more accurate representation of novel data. Finally, if data-driven feature representations are to be used as the basis for online behaviour learning algorithms, it is clear that they must be acquired from a pre-existing dataset in an initial offline step. In this light, a key advantage of hard-coded feature representations is that they remove the need for pre-existing training data in online learning scenarios.

Stage 2: Behaviour Modelling

Unsupervised Learning. Once activity detected in video data is phrased in terms of a suitable representation, behaviour models can be constructed. The principal goal of these models is to provide a means to classify future instances, and to indicate when instances are novel/anomalous. For the majority of approaches reviewed, the clusters present in a dataset are identified in an unsupervised learning procedure (see Section 2.2.2). For fixed-length vector representations, variants of K-means clustering can be applied, with the same model selection issues as described for learning feature prototypes. Where instances - which may not be easily parametrised - can be compared using a similarity measure, spectral clustering (see Section 2.2.2) provides an elegant solution, with several techniques for automatically identifying the number of clusters present in a dataset [175, 167]. Where data occurs as sets of unordered prototypes, document clustering algorithms provide a way to identify sets (or hierarchies therein) of typically occurring prototypes: several such methods (eg. the Hierarchical Dirichlet Process model used in [156]) also enable the number of clusters to automatically

be identified. Finally, it is important to note that, in cases where model-selection is sidestepped by pre-specifying quantities such as the number of clusters present in a dataset, an element of supervision is implicit: it is not clear that such approaches could operate effectively on a new dataset without human input to specify appropriate model parameters.

Supervised Learning. In several cases, the human classification/labelling of training data allows the key problem addressed by unsupervised approaches - the identification of natural groupings within a dataset - to be sidestepped. A database containing labelled pedestrian action/gaze descriptors is used to classify new instances in [114, 113], in conjunction with an efficient PCA-based search procedure [131]. Labelled instances of parametrised motion in a car park scenario are used to train a linear classifier to distinguish normal/suspicious behaviour in [88]. Labelled sequences corresponding to different classes of interactions between moving people/objects are used to build probabilistic sequence models in [48, 97]. A more complex notion of supervision is implicit in the trajectory classification strategy proposed in [32, 34], where human knowledge is encoded in a set of hand-crafted topological networks corresponding to the different routes taken through a scene.

Semi-supervised Learning. The preceding approaches rely on fully labelled datasets to construct behaviour classifiers. It is possible, however, to make use of a set of labelled data in conjunction with a larger set of unlabelled data, a strategy known as Semi-supervised Learning. This approach has received very little attention in the current behaviour modelling literature, with the following exceptions.

In [176], Zhang et al. propose a learning strategy for normal/anomalous behaviour classification, where an HMM trained on normal behaviour is used to update itself and generate anomalous behaviour models from a set of unlabelled data. This involves a recursive procedure where the lowest likelihood (w.r.t. the normal behaviour HMM) segment in the unlabelled dataset is removed and used to train a new HMM representing an instance of anomalous behaviour; the same process is then repeated on the reduced set of unlabelled data for a user-specified number of iterations. The net result of this procedure is a set of HMMs representing anomalous behaviour, with an updated version of the original normal behaviour HMM accounting for the remaining unlabelled data. While this method provides a novel way to combine normal-labelled and unlabelled data, it has a clear weakness in requiring the number of anomaly models to be pre-specified, yielding a set of anomalous behaviour models even if the unlabelled data were all normal.

In contrast, Sillito and Fisher propose a semi-supervised normal behaviour learning strategy in [133] where no assumptions are made about the proportion of anomalous examples present in the training data. Their approach - which represents behaviours in terms of parametrised trajectories - uses an incremental one class learning algorithm [132] (see Chapter 3) seeded with a small number normal-labelled examples to filter a sequence of unlabelled examples: those examples classified as normal are automatically used as further training data, while those classified as anomalous are passed to a human operator for approval (they may then either be labelled as normal, or discarded). This approach allows a normal behaviour model to be constructed from a dataset containing an unspecified proportion of anomalous behaviour while only requiring a small proportion of data to be labelled (see Chapter 5).

Batch vs Incremental Training. A key practical consideration for behaviour learning/classification systems is whether their training data must be presented in a single batch or can arrive in a sequence. A number of approaches discussed (eg. [82, 64, 60]) adopt incremental clustering algorithms, but nonetheless require training data to arrive in a batch so that multiple passes of the clustering algorithm can be made. Defined from a pragmatic viewpoint, incremental training requires that each instance is presented only once and that the resulting model can be used for classification at any point during training. In this light, virtually all approaches discussed in this review require batch training. The semi-supervised normal trajectory modelling algorithm described earlier (Sillito and Fisher [133] - see Chapter 5) is one exception. There are a small number of other exceptions, described as follows; however, a notable absence in the papers discussed is any measurement of the changing classification performance obtained as new training instances are presented.

An incremental scene modelling approach is proposed by Piciarelli and Foresti in [101], where a network of path models is constructed as training data arrives, without requiring multiple presentations of the same training data; this method does, however, require the pre-specification of certain parameters which indirectly control the complexity of the resulting model. Given that the majority of behaviour learning algorithms are unsupervised, a major issue prohibiting incremental training is the model selection process necessary for identifying the clusters present in a dataset. Xiang et al. propose a solution to this problem in [166] where a preliminary batch training stage initialises a set of behaviour models which can then be updated incrementally. This strategy provides a blueprint for adapting a number of existing approaches to enable incremental training, provided that the initial behaviour models can be incrementally adapted.

2.5 Conclusions

This chapter has examined a diverse range of strategies for representing video data and constructing behaviour models, indicating several areas where further contributions can be made.

The examination of training strategies in Section 2.4 indicated that very few behaviour learning algorithms made use of both labelled and unlabelled data, with existing examples relying on a large quantity of user-labelled data for initialisation. Similarly, few behaviour classifiers were designed to be trained incrementally, and no current presentation of these approaches has documented classification performance during training.

It was also noted that many behaviour representations require a batch learning step. If an incremental behaviour modelling algorithm without an initial batch-learning step is to be developed, a reduced set of feature representations is available. This includes raw/binned motion vectors, local action descriptors, raw trajectory data, and parametric trajectory models. As mentioned in Section 2.1.1, although there are many ways to parametrise trajectories, there is a need for a comprehensive comparison of their ability to adequately represent different behaviours.

This thesis attempts to shed further light on these unexplored areas.

Chapter 3

Incremental One-Class Learning

3.1 Introduction

One-class learning - the solution to a problem interchangeably known as "outlier / novelty / anomaly detection" - describes the formation of a generalised description of a single class of data, on the sole basis of examples from that class. In other words, it is the process of acquiring a rule that can identify whether or not a new example belongs to an observed class of data, in the absence of examples from any other class.

This problem presents itself in cases where one wishes to distinguish between members of a class for which examples are abundantly available, and members of another rarely observed class. The identification of anomalous behaviour from video footage, when only normal examples are available, is an important example of this problem. Other examples include the detection of jet engine failure [55], computer network intrusions [173], and disease symptoms [139]. In each of these domains, anomalous examples may be scarce or entirely absent during training, but their subsequent identification is of crucial importance.

This chapter presents a novel incremental one-class learning algorithm, which is shown to yield improved classification performance on a variety of benchmark datasets when compared with an existing state-of-the-art approach. First, however, an overview of existing approaches to one-class learning is presented, followed by a description of current incremental versions of these algorithms. Experiments involving motion trajectory data are presented later, in Chapter 4, where the proposed algorithm forms part of a semi-supervised framework for learning models of normal trajectories.

3.2 One-Class Learning Algorithms

This section provides an overview of existing one-class learning strategies, which fall into three broad categories.

3.2.1 Density estimation

Given a set of examples corresponding to a particular class of data, an intuitive strategy for identifying outliers is to estimate the underlying probability density function (p.d.f.) of the example data. New examples can then be assessed by evaluating their probability density with respect to the estimated p.d.f.: if sufficiently high, the new example is likely to be from the same class as the model; otherwise, it can be classified as an outlier.

Some of the earliest machine learning approaches to outlier detection involve estimating the p.d.f. of a dataset using a technique known as Kernel Density Estimation (KDE): this allows the probability density at the location of a new example to be estimated by summing the (uniformly weighted) contributions from a set of identical kernel functions (often Gaussian) centred on each member of the dataset. This approach was first adopted by Bishop in [11] to filter the data presented to a neural network (trained to estimate the fraction of oil in a pipeline given a 12-dimensional measurement) so that potentially misleading network output was discarded when unprecedented data was presented.

Similarly, Tarrassenko et al. used KDE in [139] to estimate the p.d.f. of a set of 5-dimensional measurements taken from normal mammograms, so that potentially abnormal examples could detected. More recently, Yeung and Chow successfully applied this technique to a 112-dimensional dataset corresponding to normal computer network activity, so that intrusions could be detected. In each of the preceding approaches, some form of cross-validation is used to determine an appropriate outlier detection threshold for the estimated p.d.f.; in the absence of anomalous examples, this requires the deliberate misclassification of a certain (low probability) proportion of a validation set. Another key issue, solved in a heuristic fashion in [11, 139], is the selection of appropriate kernel parameters. This can have a large impact on model quality: since our proposed approach relies on KDE, the issue of kernel parameter choice is discussed later in the chapter (Sections 3.5.2 and 3.6.2) .

Noting that the cost of evaluating a KDE model scales linearly with the size of the dataset, it can be desirable to estimate the density of a dataset in a different way. In some cases a single Gaussian model may be sufficient; for more complex datasets, a weighted Gaussian Mixture Model (GMM) can be fitted using the Expectation Max-

imisation (EM) algorithm (see eg. Bishop [12]). GMMs were used to model handwritten digit data for outlier detection by Tax and Duin in [144]; in their approach an outlier threshold is set for log probability densities at least 3 standard deviations below the mean for the training data. An alternative outlier detection strategy for GMMs is proposed by Roberts in [109]: instead of placing a threshold on the p.d.f., new examples are assigned to their closest component and a threshold derived from Extreme Value Theory (EVT) is used to identify outliers. The EVT threshold depends on the quantity of data that a given component is responsible for; it is successfully used in conjunction with GMMs to identify outliers in biomedical data in [109, 110]. This approach is discussed further in Section 3.5.3 as it is tested in conjunction with the proposed learning algorithm.

Although GMMs are less computationally expensive to evaluate than KDE models, they present two major problems. Firstly an appropriate level of model complexity (ie. number of components in the model) must be selected: to this end, several criteria have been proposed to address this problem by specifying a trade-off between the likelihood of the model given the data and the number of parameters required. Although there is no single "best" criterion, the Bayesian Information Criterion (BIC) is a popular choice (see [111, 20, 164] for experimental comparisons). Nonetheless, the complexity chosen according to such criteria can be misleading when training data is sparse, potentially leading to "underfitted" distributions covering regions that may contain outliers. Another key problem with this approach is that GMM parameters need to be initialised (typically randomly) prior to fitting using the EM algorithm: because EM is only guaranteed to find a local, rather than global maximum (for the data likelihood given the model parameters), the initial parameter values can have a large - and unpredictable - impact on the quality of the resulting model.

3.2.2 Boundary estimation

The preceding density estimation approaches estimated a p.d.f. so that a threshold could be used to define a boundary around the training examples, separating known data from outliers. An alternative strategy, therefore, is to attempt to estimate/define the boundary directly: a number of techniques have been proposed to achieve this, described as follows.

Given a dataset, an intuitive first-order solution to this problem is to find the smallest possible hypersphere enclosing the data. If a new example is not an outlier, it is likely to lie within the hypersphere. This approach was proposed by Tax and Duin as a one-class learning strategy, known as Support Vector Data Description, in [141, 142]. Drawing inspiration from the theoretical underpinnings of Support Vector Machines

(see eg. [19]), Tax and Duin formulate the discovery of a minimum-volume enclosing hypersphere, defined by a weighted subset of the training data (the support vectors), as a convex optimisation problem. The distance of a new example from the centre of the hypersphere can be calculated using a sum of weighted inner products between the example and each support vector.

As with Support Vector Machines, the optimisation includes "slack" variables which allow a trade off between hypersphere volume (inversely analogous to margin size for SVMs) and misclassified training data to be specified. Most crucially, the optimisation process allows the similarity between data points to be expressed using Kernel functions (instead of simple Euclidean distance): this enables arbitrarily complex enclosing hypervolumes to be obtained, according to the chosen kernel function. A popular choice of kernel function is the "Gaussian" kernel - shown to be a successful choice for SVDD in [141, 142] - which expresses the distance between two data points x and $x\prime$ as:

$$k(x, x\prime) = \exp\left(\frac{-\|x - x\prime\|^2}{2\sigma^2}\right)$$

This behaviour of this kernel function is determined by a parameter σ which has a large impact on the resulting model: Figure 3.1 shows the boundaries obtained by the SVDD algorithm for a 2-dimensional dataset, using the Gaussian kernel function with different values of σ. For large values of σ this kernel resembles a linear function of Euclidean distance, yielding a circular boundary similar to that obtained without using kernels. For smaller values of σ the function is peaked, emphasising the affinities of points which are close together while discounting those of more disparate points: this can yield a boundary which accurately reflects that of the underlying dataset. The kernel function can also be regarded as defining a projection of the data into a higher dimensional space [19, 142], where the definition of a hyperspherical boundary maps onto a more complex boundary in the original feature space.

It is clear that the kernel parameter has a large impact on how suitable a given boundary model is for describing a dataset and detecting outliers: if too "simple" (eg. Gaussian kernel with large σ) the boundary could encompass regions which may contain outliers; conversely, if too complex, the boundary may define an insufficient region to properly characterise the class of data being modelled. In the absence of example outliers, choosing appropriate kernel parameters presents a problem. Various methods have been proposed to address this issue, including: procedures for generating synthetic outliers (Tax and Duin [145]) and, more recently, a consistency based approach which initialises the simplest possible classifier and increases its complexity parameter until the classification performance on a validation set starts to deteriorate

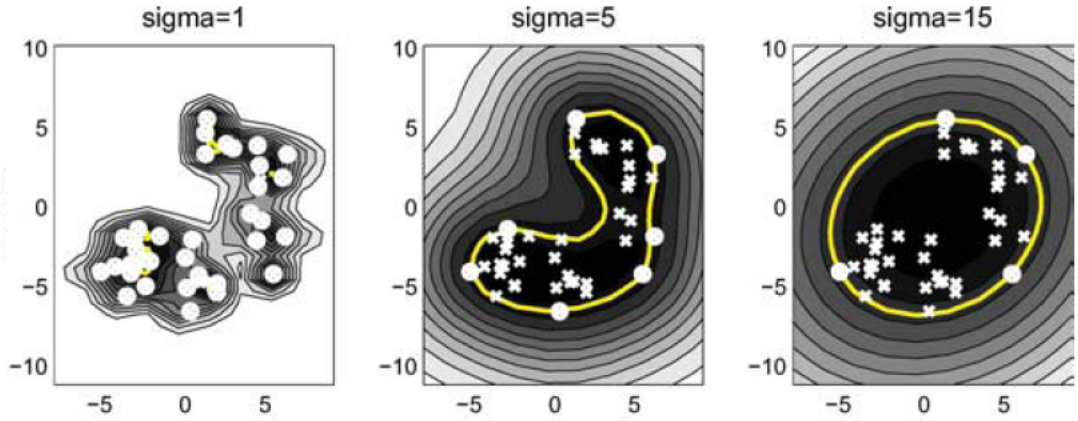


Figure 3.1: The impact of different kernel parameter values on an SVDD boundary (reproduced from [142]; see text for description.)

(Tax and Muller [143]).

Another variant of the SVDD algorithm was proposed by Scholkopf et al. in [125] where, instead of fitting a hypersphere to the data, a hyperplane is obtained which separates the training data from the origin with maximum margin. This approach has been used to detect anomalies in jet engine vibration data by Hayton et al. in [55]. When used with the Gaussian kernel function, the approach proposed by Scholkopf et al. has been shown to yield equivalent results to SVDD [142, 12].

A related approach, which could equally belong in the density estimation category, is proposed by Roth in [118] where a kernel-based version of Linear Discriminant Analysis (LDA) is formulated. This variant of LDA is used to address the one-class learning problem by finding the one dimensional projection which best separates a one-class dataset from a negative replica of itself. Since LDA models each class with a Gaussian distribution, Roth's approach yields a version of the Mahalanobis distance which can be expressed in terms of kernel functions: this model effectively corresponds to Gaussian density estimation in the feature space defined by the kernel function. Like the preceding approaches, this allows flexible boundaries to be defined in the original feature space. Its key advantage, however, is that it allows such boundaries to be defined using statistical approaches for detecting outliers with respect to a Gaussian distribution.

3.2.3 Subspace estimation

The preceding approaches characterise a dataset by estimating its density function or enclosing boundaries. Another way to characterise a dataset is to determine whether it can be well defined by a lower-dimensional feature space: for example if a set of

3-dimensional data is found to lie on a plane, the data can be accurately described using a 2-dimensional representation (together with a mapping between the original and reduced-dimensionality feature spaces.) When the reduced-dimensionality representations are mapped back to the original feature space, it is possible to calculate a reconstruction error for each data point to determine the quality of its representation. This has an important role for one-class learning: if a given class of data is well-characterised by a reduced dimensionality representation, then the reconstruction error for a new data point provides an indication of whether or not it belongs to the same class.

Principal Components Analysis (see eg. [12, 53]) is a useful technique for identifying low dimensional subspaces that can be defined by a linear mapping (ie. from a given feature space to a straight line/plane/hyperplane occurring in that space). This technique defines a subspace with an orthogonal set of basis vectors derived from the largest N eigenvectors of the covariance matrix of a given set of data. For the purpose of one-class learning, this technique is only effective if a given class of data is well characterised by the linear subspace on which it lies. However, real world data may lie on curved subspaces (eg. 2D data may lie on a curve rather than a straight line), in which case a one-class classification scheme based on PCA reconstruction error may perform poorly.

One solution to this problem is to used Kernel PCA, a nonlinear generalisation of PCA for feature spaces defined by kernel functions, proposed by Scholkopf et al in [126]. This approach has recently been adopted as a one-class learning strategy by Hoffman in [58]. It is shown in [58] that Kernel PCA (KPCA) performs similarly to Support Vector Data Description for the "hard margin" case (where all training data must be correctly classified), using a Gaussian kernel in both cases. However, in the "soft margin" case where a proportion of training data can be misclassified, KPCA is shown to outperform SVDD on a synthetic 2D dataset containing outliers. On two real datasets KPCA is able to outperform both SVDD and Kernel Density Estimation (albeit by a small margin). However, as with other kernel-based methods, the success of KPCA relies crucially on the correct choice of kernel parameter and, additionally, that an appropriate number of eigenvectors is chosen.

Auto-associative neural networks provide another solution to the problem of estimating nonlinear subspaces for one-class learning. Such networks have an input layer corresponding to the original feature space, one or more hidden layers with fewer dimensions than the input, and an output layer of the same dimensionality as the input. The network is trained to generate an output equal to its input, thereby achieving a reconstruction from the representational space defined by the hidden layer(s). Auto-

associative networks have been applied to novelty detection in a variety of domains [63, 99, 84] where, as with PCA and KPCA, reconstruction error is used to define classification boundaries. The use of multiple hidden layers enables auto-associative networks to find nonlinear subspaces [12]. In the case of one hidden layer, these networks have been shown by Bourland and Kamp to be, at best, equivalent to PCA [16]; however, Japkowicz et al. show in [62] that nonlinear activation functions in networks with one hidden layer can yield nonlinear reconstruction error surfaces that enable better one-class classification performance than PCA.

Summary

This section has reviewed a variety of different techniques that can be used to construct a one-class classifier from a given dataset. It is clear that many of these techniques are potentially very effective. However it is also clear that the problem of model selection is inherent in every approach discussed: each algorithm has at least one parameter whose value has a crucial effect in determining the resulting classification performance. In all cases the value of this parameter specifies a compromise between the ability to generalise to new instances of the observed class and the ability to detect outliers. To address this, many of the models discussed were accompanied by some form of cross-validation strategy for choosing parameter values. It is nonetheless important to realise that these models are intended to solve a fundamentally ill-defined classification problem, where a second class of hypothetical "novel" items is by definition unavailable: in this light, it is difficult for any model selection strategy that can only utilise information from a single class of data to accurately determine the resulting classification performance. It is difficult to precisely determine the 'best' approach from the literature: since each one is shown to be more effective than other approaches on certain datasets, it is likely that the relative efficacy of each algorithm depends on the specific task at hand.

3.3 Existing Incremental Approaches

While the approaches reviewed in the preceding section provide a range of solutions to the one-class learning problem, each method assumes that training data is available in a single batch. In many cases, this means that if a new example/set of examples is to be added to the training set, the original training procedure must be repeated from scratch on the augmented set of training data. A small number of incremental one-class learning strategies, reviewed as follows, have been proposed to address this issue.

Yamanishi, K.; Takeuchi, J.; Williams, G. & Milne, P. "On-line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms", [171].

In [171] Yamanishi et al. propose a one-class learning strategy based on incremental learning of Gaussian mixture models; in contrast to other GMM based one-class learning algorithms (eg. [112]) this approach is designed so that the GMM parameters can be updated indefinitely by new training data. To achieve this, Yamanishi et al. employ a variant of the incremental EM (Expectation Maximization) algorithm [93] where each new data point makes an exponentially decaying contribution to the sufficient statistics of each component in the mixture model: each data point is only used in a single EM cycle before being discarded. In contrast to the other one-class learning algorithms discussed, this method does not attempt to incrementally build a complete normal class description, but rather to model a finite window of training data preceding each new example. In order to detect outliers, each new data point that is used to update the model is assigned a score which quantifies the resulting change in the mixture model (using an approximation of the Hellinger distance). Once a dataset has been processed, outliers can be identified by ranking data according to their Hellinger distance scores: a high value, indicating a large change in model parameters, provides an indication that a data point is novel. It is important to note this is an unsupervised learning algorithm: the model is updated on all data points - including potential outliers - so that a score can be calculated following each training update in order to rank the data. A potential weakness of this strategy is that a series of consecutive anomalous examples would receive diminishing novelty scores as the model parameters adapted: therefore, in subsequent rankings, these anomalous examples may not be highlighted due to their local temporal context during training. Another weakness of the method proposed in [171] is that it does not include a strategy for choosing an appropriate number of mixture components: this quantity must be pre-specified, and cannot change during training.

Tax, D. M. J. & Laskov, P. "Online SVM learning: from classification to data description and back" [146].

It is also possible to formulate kernel based boundary estimation algorithms for incremental training: in [146] Tax and Laskov develop an incremental version of the Support Vector Data Description algorithm. As with the original SVDD algorithm, the incremental version proposed [146] seeks to enclose a given dataset with a minimum volume hypersphere, defined by a weighted subset of the original data (the support vectors). In the original SVDD algorithm, the optimal set of weights is defined by a

constrained optimisation problem, which can be solved by quadratic programming. To enable incremental training, Tax and Laskov propose a method for augmenting an existing optimal solution with new examples, whereby the weights of existing support vectors are adjusted whenever a new example is added so that an optimal solution is maintained (see Laskov et al. for details [74]). Furthermore, a method is suggested in [146] for initialising the model with a small initial set of data points (all set as support vectors with equal weight), thus removing dependence on a preexisting batch solution. As with the original SVDD algorithm, the incremental updating process is defined in terms of inner products, which can be replaced with kernel functions so that complex boundaries can be defined. Thus, this incremental algorithm also inherits the problem of choosing an appropriate kernel function and parametrisation therein. Since the algorithm assumes a fixed kernel function which cannot change as more examples are observed, it leads (in the absence of prior knowledge) to the paradoxical scenario of needing to retrain the algorithm several times - despite its incremental nature - in order to determine an appropriate kernel parameter through cross-validation (eg. [145, 143]). While it is in principle necessary to retain all data points to achieve the best solution, it is suggested in [146, 74] that the computational/storage costs of this algorithm could be limited by maintaining a finite window consisting of the last N data, discarding the data point with lowest weight at each iteration.

Summary

While both of the preceding approaches enable the incremental updating of a one-class classifier, the algorithm proposed by Yamanishi et al. differs by maintaining an exponentially decaying description of a dataset instead of a complete description, potentially leading to the false detection of outliers as old examples are forgotten. As with their batch counterparts, the performance of these incremental approaches is contingent on the correct choice of model parameters. In both cases it is assumed that parameter values are determined according to prior knowledge, or the results of a cross validation experiment. When training data arrives as a single batch, it is reasonable to perform cross validation to determine the best model parameters. However, for applications where incremental one-class learning is useful, its function is to a large extent undermined by the need to retrain several times for cross validation. While it is clear that incremental one-class learning methods can be justified as a means of augmenting an existing batch classifier with new information, such methods would have far greater utility if model selection could be incorporated in the incremental learning process.

3.4 Overview of Incremental GMM Learning

Noting that the algorithm proposed in this chapter entails the incremental construction of a Gaussian Mixture Model (GMM), this section reviews a range of existing approaches to the problem of incremental GMM learning and establishes how they relate to the proposed algorithm. Sections 3.4.2 and 3.4.3 discuss techniques for incrementally updating a GMM with new data, without the requirement of a complete set of historical data: Section 3.4.2 summarises strategies for updating model parameters, while 3.4.3 summarises techniques for changing model structure in response to new data. Finally Section 3.4.4 briefly discusses several related techniques where model structure is determined in an incremental fashion from a single batch of data.

3.4.1 Goal of GMM Learning

A Gaussian Mixture model defines a probability density function in terms of a set of $K \geq 2$ Gaussian distributions with prior probabilities $\pi = \{\pi_1, \ldots, \pi_K\}$; means $\mu = \{\mu_1, \ldots, \mu_K\}$, and covariance matrices $\Sigma = \{\Sigma_1, \ldots, \Sigma_K\}$ as follows (see eg. [53, 12] etc):

$$p(z|\pi, \mu\Sigma) = \sum_{j=1}^{K} \left(\pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)\right) \tag{3.1}$$

For a given set of data $X = \{x_1, \ldots x_N\}$, and a prespecified number of mixture components K, the goal of GMM learning is to find the parameter values for π, μ, Σ that maximise the likelihood of [generating] the training data. If X is available as a single batch, this optimisation can be accomplished using the Expectation-Maximisation algorithm [35] which, given a set of initial parameter estimates, works by iterative application of the following steps:

1. **(E Step)** Estimate the "responsibility" $r(x_n, k)$ of the kth component for the nth data point x_n:
$$r(x_n, k) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \tag{3.2}$$
2. **(M Step)** Calculate μ, Σ, π for each component k according to the estimated responsibilities:

$$\begin{aligned} \pi_k^{new} &= \frac{1}{N} \sum_{n=1}^{N} r(x_n, k) \\ \mu_k^{new} &= \frac{1}{N.\pi_k^{new}} \sum_{n=1}^{N} r(x_n, k).x_n \\ \Sigma_k^{new} &= \frac{1}{N.\pi_k^{new}} \sum_{n=1}^{N} r(x_n, k).(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T \end{aligned} \tag{3.3}$$

	Parameter Update $\mathcal{S}^{new} = \ldots$	**Refs**
B1. Batch EM	$\sum_{n=1}^{N} r_{new}(x_n,k).C(x_n)$	[35]
B2. Incremental EM	$\mathcal{S}^{old} + (r_{new}(x_n,k) - r_{old}(x_n,k)).C(x_n)$	[93]
O1. Soft assignment	$\mathcal{S}^{old} + r(x_{new},k).C(x_{new})$	[5, 44]
O2. Hard assignment	$\mathcal{S}^{old} + I\left(r(x_{new},k) = \max_j r(x_{new},j)\right).C(x_{new})$	[132, 31, 76]
O3. Exponential decay*	$(1-\alpha)S^{old} + \alpha C(x_{new})$	[136, 69, 171, 76]

Table 3.1: This table summarises several different ways to update the parameters of a mixture model with new data, expressed in terms of sufficient statistics. In each case $C(x_n)$ denotes the contribution of data point $x_n$to the sufficient statistic in question (specifically: $C_\pi(x) = 1$, $C_\mu(x) = x$ and $C_\Sigma(x) = xx^T$). For example, update **B2** for $S^{new}_{\Sigma_k}$ would be $\mathcal{S}^{old}_{\Sigma_k} + (r_{new}(x_n,k) - r_{old}(x,k))x_n x_n^T$. (*Note that for **O3**, π_k is directly estimated using Equation 3.7 and Equation 3.4 is replaced with $S_{\pi_k} = \pi_k$.)

For a given parameter initialisation, this procedure is guaranteed [35] to converge to a local optimum (of the training data log-likelihood): it cannot, however, be directly used for incremental learning as both optimisation steps refer to the whole dataset.

3.4.2 On-line refinement of model parameters.

It is possible to adapt Equations 3.2 and 3.3 so that GMM parameters can be adjusted on the basis of single items of training data. A key insight is that the parameters π, μ, Σ can be represented in terms of the following *sufficient statistics* (specifically, sums of the contributions from each data point) $\mathcal{S}_{\pi_k}, \mathcal{S}_{\mu_k}, \mathcal{S}_{\Sigma_k}$, which can then be updated - in a number of different ways - for single data points:

$$\mathcal{S}_{\pi_k} = \sum_{n=1}^{N} r(x_n,k) \quad \rightarrow \quad \pi_k = \frac{1}{N}\mathcal{S}_{\pi_k} \tag{3.4}$$

$$\mathcal{S}_{\mu_k} = \sum_{n=1}^{N} r(x_n,k).x_n \quad \rightarrow \quad \mu_k = \frac{\mathcal{S}_{\mu_k}}{\mathcal{S}_{\pi_k}} \tag{3.5}$$

$$\mathcal{S}_{\Sigma_k} = \sum_{n=1}^{N} r(x_n,k).x_n x_n^T \quad \rightarrow \quad \Sigma_k = \frac{\mathcal{S}_{\Sigma_k}}{\mathcal{S}_{\mu_k}} - \mu_k \mu_k^T. \tag{3.6}$$

Table 3.1 summarises several different strategies for updating a GMM incrementally. The strategy with the greatest theoretical support is an incremental variant of the EM algorithm (Table 3.1, B2) proposed in [93], where Neal and Hinton showed that performing a partial E-step (so that Equation 3.2 is only recomputed for a single data point) before each M-step is sufficient to guarantee convergence to a local maximum

of the data log likelihood. However, this strategy does not provide a means for online learning, as it requires the whole dataset to be retained together with $N \times k$ evaluations of Equation 3.2. Nonetheless, a range of similar strategies - albeit without the same theoretical guarantees - can be used to incrementally update GMM parameters on a single-presentation basis. One simple strategy, adopted by Friedman and Russell in [44] (Table 3.1, O1), simply adds the contribution of each new example to the sufficient statistics for each component. In a similar vein, several approaches [76, 132, 31] update the closest component to each new data point (Table 3.1, O2). A key problem with the summation of historical contributions from all observed data is that the responsibilities of mixture components for a given data point are bound to change as the parameters are updated, so the contributions of earlier data to the model parameters will be based on poor estimates of the current component responsibilities.

One way to alleviate this problem of misleading contributions from historical data is to maintain an exponentially decaying average (Table 3.1, O3) of the contributions from recently observed data. Given this update the impact of each new data point (and hence the rate at which old ones are forgotten) is determined by a parameter α; furthermore, the component weight is now estimated directly (rather than through Equation 3.4) as follows:

$$\pi^{new} = (1-\alpha)\pi^{old} + \alpha f(r(x_{new},k)) \tag{3.7}$$

This approach forms the basis of a popular method proposed by Stauffer and Grimson in [136] for learning background models in RGB space, and several other methods [69, 171, 76]. It should be noted that component responsibilities $r(x,k)$ do in fact feature in these approaches: in [136, 76] only the component closest to each new data point is updated, while in [69, 171] the contribution of each x_{new} is weighted by $r(x_{new},k)$. An inherent problem with this type of decaying average is that the initial parameter estimates have an undesirably large impact on the resulting model in the early stages of training. Lee proposes a means to address this in [76], by initially updating components through summation of data contributions (Table 3.1, O2) and then switching towards the exponential decay update (Table 3.1, O1) once a certain quantity of data has been observed. This can be implemented by maintaining a separate decay constant for each component such that

$$\alpha_k = (\frac{1-\alpha}{c_k} + \alpha) \tag{3.8}$$

where $c_k = \sum_{n=1}^{N} I\left(r(x_n,k) = \max_j r(x_n,j)\right)$ is a counter for the number of data points represented by the component, so that α_k is close to $\frac{1}{c_k}$ for low c_k but tends towards α as c_k increases. This strategy is shown in [76] to yield faster convergence

for several different input distributions when compared to approaches where $\alpha_k = \alpha$. Nonetheless, this approach, as with [69, 171, 76, 136], requires that α is prespecified.

3.4.3 On-line refinement of model structure.

The methods described in the preceding section provide a means to update an existing GMM with new data. On their own, however, these update equations are insufficient for incremental GMM learning as they assume that both the number of mixture components and their initial parameter values have been appropriately prespecified. This section reviews strategies for dynamically adjusting the component structure of a GMM on the basis of data that arrives sequentially and cannot be retained after being used for training: existing approaches either process this data in chunks [27, 135, 52, 51] or on a single-item basis [136, 123, 5, 182, 132, 72, 31].

Data arrives as single points.

Given the latter constraint, a key problem is initialising the model. In all virtually all approaches, the first data point observed instantiates the first new component with a heuristically determined initial covariance matrix; thereafter a range of different strategies are employed. In [136, 76] new data points that fall outside 2.5 standard deviations of the closest existing component automatically instantiate a new one or - if a pre-specified maximum number of components is exceeded - replace the existing component with lowest weight; otherwise the closest existing component is updated as described in Section 3.4.2. The method proposed in [136] has an additional heuristic constraint - for the purpose of background modelling - where a reduced GMM is generated by ranking components according to $\frac{weight}{variance}$ and selecting the first N components that have a total weight below a prespecified threshold.

A similar approach, with a more principled justification, is adopted by Zivkovic and van der Heijden in [182] where the recursive weight update rule (Equation 3.7) is reformulated to include a Dirichlet prior on the distribution of component weights, with uniform negative coefficients set to $-\frac{P}{2}$ where P is the number of model parameters per component. This results in the following update rule, which imposes a penalty on over-complex models by reducing the weight of components that only account for a small proportion of data points:

$$(1-\alpha)\pi_k + \alpha\frac{r(x_{new},k) - \frac{\alpha P}{2}}{1 - M\frac{\alpha P}{2}} \tag{3.9}$$

Although the weight update is constrained so that $\sum_i \pi_i = 1$, individual weights can fall below zero - this is used in [182] to facilitate the reduction of model complexity:

components with $\pi \leq 0$ are removed, and the remaining positive weights are then renormalised. This method is shown to give good performance, yielding GMMs with the correct number of components for various benchmark datasets. It should be noted that this method must be initialised with a prespecified maximum number of M mixture components (centred on the first M training data, with prespecified covariance), and that the number of components can only be reduced during training. In this light, a key weakness of this method is that it provides no means to adapt model complexity in response to the appearance of new clusters.

In contrast, several other methods have been proposed where new components *can* be generated in response to new data. Arandjelovic and Cipolla propose an incremental GMM learning algorithm in [5] with the specific constraint that incoming training data is "temporally coherent", meaning that consecutive observations are assumed to lie close together in feature space. Their algorithm is typically initialised with a single Gaussian fitted to the first few data points; thereafter two models are maintained: 1) a "historical mixture" (in the first instance this is equivalent to the initial model) which is not updated by subsequent data points and 2) a "current mixture" which is updated by adding the contribution of each new data point to its sufficient statistics (Table 3.1, O1). Every time the current mixture is updated, a more complex model is hypothesised by splitting the current mixture (ie. by subtracting historical components from their corresponding current components). The BIC model selection criterion is then used to determine whether to split any component (this criterion is evaluated in the absence of historical data by calculating the expected likelihood of the number of points represented by a given component, in both merged and split forms). If any components are split, the current model replaces the existing historical mixture. Although this is an elegant strategy for temporally coherent data, if the latter constraint is violated (eg. if current mixture is updated with temporally interspersed points from two distinct locations) new components created by splitting the model may not provide a good representation of the underlying data.

This chapter proposes an alternative approach to this problem [132] whereby each new data point contributes a new component to the mixture in a manner equivalent to a kernel density estimate, and the resulting model organisation is adjusted - to keep the number of components under a prespecified maximum - by merging pairs of components according to a cost function (see Section 3.5 for details). The premise of this method is that - as with kernel density estimation - a large number of mixture components can be used to represent a wide range of different distributions, even though many could be represented with fewer components. This assumption is adequate for the purpose one-class learning, where it is not necessary to have a model structure that

accurately reflects the intrinsic complexity of the dataset, but rather one that yields a useful approximation of the underlying density of that data.

A similar method has subsequently been proposed by Kristan et al. in [72] for the purpose of incremental (1 dimensional) density estimation: as with the proposed approach, Gaussian kernels are continually added to a mixture model, while model complexity is kept below a prespecified maximum by compressing the mixture whenever a new kernel is added. The variance of each new kernel added to the model is estimated using a variant of the "plug-in" method for bandwidth estimation (see eg. [159]) which relies on an estimate of the curvature of the resulting density function. While this method provides the advantage that the kernel width can be re-estimated even when the remainder of the model is no longer a kernel density estimate, its applicability is limited as it cannot easily be extended to the multivariate case (see eg. [177]). When the addition of a new kernel causes the model complexity to exceed the prespecified maximum, the model is compressed in the following way: first a new mixture is created where the variance of each original component is inflated by a prespecified amount; the weights of the inflated mixture that minimise the Integrated Squared Error (ISE) with respect to the original mixture are then determined (having been defined as a quadratic programming problem) using the Sequential Minimal Optimisation algorithm [106]. In many cases, the preliminary inflation of the mixture will cause some of the resulting optimised weights to be zero, allowing these components to be removed. Finally the ISE between the reduced inflated mixture and the original mixture is then optimised with respect to all model parameters using the Levenberg-Marquadt algorithm: again, it is not clear that this part of the algorithm would be suitable for on-line learning at higher dimensionalities, given the quadratic increase in the number of GMM parameters w.r.t. dimensionality.

Another related approach that *is* applicable to multivariate data was recently proposed by Declercq and Piater in [31]. As with the preceding approaches, each new example contributes a new component to the mixture model and pairs of components are merged to reduce model complexity. However, unlike the preceding approaches, there is no upper limit on the number of components in the model: instead, after each new component is added, their method attempts to merge pairs of components until a threshold on a merging cost function is exceeded. Merging cost is evaluated by comparing the empirical distribution function of the data represented by two components with a Gaussian cumulative distribution function - specifically, by numerically integrating the absolute difference between the two curves, and mapping this distance to a pseudo-probability λ using a negative exponential function with a user-specified scale parameter. Two different methods are proposed for updating a mixture, one in-

volving a fixed conservative merging threshold of $\lambda = 0.95$, and another using a lower version of this threshold (rescaled according to the number of examples observed) that encourages merging. These methods are used in [31] to define a two-level updating procedure where a mixture model is learned using the lower threshold, while each component is simultaneously represented by a more complex mixture updated using the higher threshold: this allows a component to be split - when its merging threshold is exceeded - by using EM to fit a pair of components to samples drawn from its the underlying mixture. This method is shown to work well on several toy datasets, although it is not clear how the empirical distribution function for the underlying training data - which is central to the merging cost function - can be evaluated without reference to that data.

Data arrives in batches.

Finally, several other methods have been proposed for updating a GMM with the parameters of a further GMM learned from new data. This can be regarded as a form of "chunked" on-line learning, where many of the issues implicit in the preceding methods, eg. evaluating the cost of component merging, are still of central importance.

An early approach to the problem of merging two mixture models was proposed by Hall et al. in [52] where mixture components are modelled as hyper-ellipsoids, and pairs of components are merged if their merged volume is lower than the sum of their individual volumes. Subsequently, Hall and Hicks proposed a more complex method in [51, 57] where two GMMs are initially trivially combined to produce a $N + M$ component mixture by concatenating their parameters and renormalising priors, so that a matrix of weights $w_{i=1\ldots N+M, j=1\ldots K}$ can then be specified to define a reduced mixture with K components, where each is a weighted combination of components from the concatenated mixture (constrained so that columns corresponding to new components must each sum to 1). Noting that weight matrices which satisfy this constraint lie on specific hyperplane, Downhill Simplex optimisation is then used to search on this hyperplane for weights that optimise the χ^2 distance (which can be calculated in closed form) between the reduced and concatenated mixtures. This procedure is repeated for different values of K yielding a range of reduced mixtures; an approximation to the expected log-likelihood per data point (achieved by calculating the derivative of a sum of exponentials approximating $f(\eta) = \int p(x)q(x)^\eta dx$ at $\eta = 0$) is then used to calculate the BIC criterion so that an optimum number of components can be chosen.

The method proposed in [51] is not intended for an on-line learning scenario, and it may be too computationally demanding for such purposes. In contrast, Song and Wang recently proposed an on-line GMM learning algorithm in [135] which also re-

lies on merging pairs of GMMs. In [135], each new batch of data is used to fit several GMMs of different complexities, and is then represented by the GMM that maximises the BIC criterion. The data is clustered by assigning each item to the mixture component with highest posterior probability: at this point each statistical tests are performed to determine whether the clusters of data represented by each new mixture component are statistically equivalent to any component in the pre-existing mixture. Firstly, the equivalence of covariance matrices is assessed using a test statistic proposed in [75]; if this test is passed (ie. null hypothesis that the matrices are equivalent is accepted), Hotelling's T^2 test is then used to assess the equivalence of the two means vectors. Any pair of components deemed to be statistically equivalent is then merged, with any conflicts (where a new component is deemed to be equivalent to more than one existing component) resolved comparing the likelihood of the data represented by the new component w.r.t. each merging candidate.

In a similar vein, Charron et al. propose a technique for merging GMMs in [27] where a set of weights (as in [51, 57]) defining a set of merged combinations of components from a concatenated mixture is determined using the EM algorithm. This is achieved by reformulating the standard E step (Equation 3.2) so that the responsibility of a merged component for each existing component is calculated in terms of the expected likelihood $(\int p(x)q(x)dx)^n$ of n data points drawn from existing mixture component $p(x)$ (where n is the number πN of data points accounted for) when represented by new merged component $q(x)$. The M step then consists of creating new components by merging existing components according to the responsibilities calculated in the E step. To determine an appropriate number of components, the weight update is reformulated to include a Dirichlet prior (as in [182]) which forces the weights of redundant components to 0 so that they can then be removed. An interesting issue which is not clear from [27] is how the weights defining the initial set of merged components are initialised.

3.4.4 Incremental model construction given batch data.

A number of other incremental approaches have been proposed for learning from a batch dataset: these approaches are "incremental" in the sense that they allow a GMM to be constructed one component at a time, but not in the sense discussed earlier, as each update step requires the whole dataset to be available. Although these approaches do not provide a means to incrementally learn a GMM from a stream of incoming data, they address certain issues - such as selecting an appropriate number of components - which are relevant to problem of online GMM learning.

An early example of this type of approach was proposed by Vlassis and Likas in

[152], where the quality of GMM parameters learned using EM is assessed by measuring the sample kurtosis of the data underlying each mixture component. The excess kurtosis ($k = \frac{\mu_4}{\mu_2^2} - 3$ where the nth moment $\mu_n = E\left((x-\mu)^n\right)$) provides a measure of the "peakedness" of a distribution [161] and has the property that $k = 0$ for a Gaussian distribution. The latter property is exploited in [152] in the form of a weighted total kurtosis measure for a GMM which has a positive value that is close to zero when the data underlying each mixture component is well described by a Gaussian distribution. Mixtures are fitted in [152] by optimising the parameters of an initially small number of components using EM, and then splitting a mixture component if 1) the kurtosis measure starts to increase or 2) the kurtosis measure exceeds a prespecified threshold once a (local) maximum likelihood is reached. If these conditions are met, the component making the largest contribution to the total kurtosis measure is split: noting that this method is intended to deal with 1-dimensional data, splitting is achieved by instantiating two replicas of the original component with means 1 standard deviation to the left/right of the original mean. While this approach is interesting, the use of kurtosis to indicate the goodness-of-fit for individual Gaussian components does not trivially extend to multivariate data. Moreover, the straightforward method of splitting components used in [152] does not translate to the multivariate case.

This problem is addressed in [153], where Vlassis and Likas propose a method for incrementally increasing the number of components in a (potentially multivariate) GMM. To determine the best parameters for a newly inserted component, new components (with a fixed spherical covariance) are hypothesised at the location of all data points, and the component which yields the highest data log likelihood for the mixture is chosen for insertion. A free parameter of some importance is the weight w_{new} of the new component - this is chosen for each candidate component by maximising a 2nd order Taylor approximation (about $w_{new} = 0.5$) of the data log likelihood as a function of w_{new}. Finally, partial EM steps (where the parameters of existing components are fixed) are used to optimise the parameters of the new component before it is added to the mixture. Starting with a one-component mixture, this process continues until some stopping criterion (eg. prespecified maximum number of components, no increase in data likelihood, BIC favours lower complexity etc.) is reached. One drawback of this method is that - given a large dataset - the computational cost of evaluating candidate components is high. Verbeek et al. suggest an alternative method in [151], where the data underlying each existing component is identified (by hard assignment), and used to generate a small set of candidate components. This is achieved by randomly selecting two data points from each set and splitting the data according to their closest point: the mean and covariance of these subsets define two new candidate components; this

process is repeated M times to generate $2M$ candidate components for each existing component. Because a much smaller number of candidates is generated compared to [153], it is possible to speculatively optimise the parameters of each candidate using partial EM steps before augmenting the original mixture with the candidate that yields the highest data log likelihood as in [153], and applying EM to optimise the whole model.

A novel variant of this approach has been proposed by Titsias in [148], where a mixture model is constructed one component at a time, using an outlier component (with a uniform distribution) to represent any data that is not yet explained by an existing mixture component. Initially all data is represented by an outlier component, and the first component is initialised with mixing weight set to 0.5, mean centred on a randomly selected data point, and a spherical covariance matrix scaled by the maximum variance in across all individual dimensions. The EM algorithm then is used to optimise the parameters of the new component together with the mixing weight which determines how much of the data is still accounted for by the outlier component. To add new components the parameters of any existing components are fixed, and the initial task of fitting the data with a single component and an outlier component is repeated with the following important constraint: after the first component has been added, the likelihood function optimised by EM is weighted according to the responsibility of the outlier component for each data point, so that data that is represented by existing components is probabilistically "masked" during the EM optimisation of each new component (making this approach conceptually similar to Boosting [43]). This process yields a natural stopping criterion when only a small proportion of the dataset is represented by the outlier component (eg. in [148] when the number of unaccounted data points is equal to the data dimensionality), allowing the number of clusters to be determined automatically. This method is shown to improve on that of Verbeek et al. on several benchmark datasets, and a variant of this method is applied by Titsias and Williams in [149] to learn models of multiple foreground objects from video sequences.

3.5 Proposed Algorithm

The semi-supervised trajectory learning framework proposed in this thesis (see Chapters 1 and 4) requires an incremental one-class learning algorithm that is capable of finding an appropriate model without undertaking a separate batch cross validation stage. This section attempts to address this problem by proposing a novel incremental one-class learning algorithm which only requires an upper limit on model complexity

to be specified, while Section 3.6 examines the performance of the algorithm, comparing it with that of the Incremental SVDD algorithm discussed in the preceding section.

3.5.1 Incremental Density Estimation

Given a stream of incoming data our goal is to construct a Gaussian mixture model that can be used to detect outliers at any stage during training. The procedure proposed in this section seeks to achieve this by initially representing the dataset with an over-fitted model (generalising poorly to both the dataset being modelled and - crucially - to outliers) and then incrementally improving the model's coverage of the dataset as new examples arrive. This is achieved as follows: in the initial stages of training the mixture model is identical to a kernel density estimate, with each new data point contributing an identical Gaussian kernel; then, after a certain number of data have been added to the model, the number of components is kept constant by merging a pair of components for every new kernel added.

At every stage in training the model consists of a set of N Gaussian components defined by: weights/prior probabilities $\pi = \{\pi_1, \ldots, \pi_N\}$; means $\mu = \{\mu_1, \ldots, \mu_N\}$, and covariance matrices $\Sigma = \{\Sigma_1, \ldots, \Sigma_N\}$. Together, these form the following probability density function, where d refers to the dimensionality of the data (see eg. [53, 12] etc):

$$p(z) = \sum_{i=1}^{N} \left(w_i \cdot \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_i|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(z-\mu_i)^T \Sigma_i^{-1} (z-\mu_i)} \right) \tag{3.10}$$

The training procedure outlined in the remainder of this section provides a way to estimate GMM parameters that enable the strategies outlined in Section 3.5.3 to define useful classification boundaries for detecting outliers.

Initialisation phase In the initial stages of training - before the number of observed data N reaches the user defined parameter N_{max} - we build a mixture model by creating a new Gaussian component for every new training example. Each component is centred on its corresponding data point but all components share an identical covariance matrix K and weight $\frac{1}{N}$. Thus, for a given dataset $X = \{x_1, \ldots, x_N\}$, Equation 3.10 becomes:

$$p(z) = \frac{1}{(2\pi)^{\frac{d}{2}} |K|^{\frac{1}{2}}} \cdot \frac{1}{N} \cdot \sum_{n=1}^{N} e^{-\frac{1}{2}(z-x_n)^T K^{-1} (z-x_n)} \tag{3.11}$$

At this stage in training the model is identical to Kernel Density Estimation, which has been used for outlier detection in a variety of scenarios (see Section 3.2). Training the model is straight-forward: when a new training example x_{new} is received, it is simply added to the set X:

$$X \rightarrow X \cup \{x_{new}\} \tag{3.12}$$

As each new example arrives, the weights of the components are updated (to the new value of $\frac{1}{N}$), and the covariance matrix common to each component is re-estimated (see Section 3.5.2 for details). At the end of this phase, when the maximum number of components have been added $N = N_{max}$, the mixture model is defined as follows (where $\sigma(N_{max})$ is the final estimate of σ - see Section 3.5.2).

$$\begin{aligned} w_{1...N} &= \tfrac{1}{N} \\ \Sigma_{1...N} &= I^d \cdot \sigma(N_{max}) \\ \mu_{1...N} &= x_{1...N} \end{aligned} \tag{3.13}$$

Merging phase Once the maximum number of mixture components has been reached, we employ a merging strategy to keep the number of components constant. As with the previous stage, each new data point x_{new} contributes a Gaussian kernel function to the model; the covariance matrix of each kernel is set to the final estimate obtained in the previous stage. This yields a set of $N_{max} + 1$ components. To restore the model to the desired number of components N_{max}, a pair of components - which may include the new one - is then merged as follows (see Appendix A for derivation):

$$\begin{aligned} w_{merge(i,j)} &= w_i + w_j \\ \mu_{merge(i,j)} &= \tfrac{w_i}{w_i+w_j} \cdot \mu_i + \tfrac{w_j}{w_i+w_j} \cdot \mu_j \\ \Sigma_{merge(i,j)} &= \tfrac{w_i}{w_i+w_j} \left(\Sigma_i + \mu_i \mu_i^T\right) + \tfrac{w_j}{w_i+w_j} \left(\Sigma_j + \mu_j \mu_j^T\right) \\ &\quad - \mu_{merge(i,j)} \mu_{merge(i,j)}^T \end{aligned} \tag{3.14}$$

We wish to choose a pair of components to merge in a way that minimizes the resulting change in the p.d.f. encoded by the model. The Kullback-Leibler divergence provides a means of assessing the "damage" caused by replacing a particular pair of components with a single merged component. Essentially, the KL divergence $\mathcal{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$ quantifies the expected information loss per sample when an approximating distribution Q is substituted for a true distribution P. For a pair of Gaussian distributions $G_p = \{\mu_p, \Sigma_p\}$ and $G_q = \{\mu_q, \Sigma_q\}$ of dimensionality d, it can be calculated as follows [47, 12]:

$$\mathcal{KL}(G_p||G_q) = \frac{1}{2} \left(\log \frac{|\Sigma_q|}{|\Sigma_p|} + \mathrm{Tr}(\Sigma_q^{-1}\Sigma_p) + (\mu_p - \mu_q)\Sigma_q^{-1}(\mu_p - \mu_q)^T - d \right) \tag{3.15}$$

This allows us to quantify the cost of replacing components G_i and G_j (where $i \neq j$) with their merged counterpart $G_{merge(i,j)}$ by calculating a weighted combination (as

proposed by Goldberger and Roweis in [47]) of their respective Kullback-Leibler divergences from $G_{merge(i,j)}$ as follows:

$$cost(G_i, G_j) = w_i \mathcal{KL}(G_i || G_{merge(i,j)}) + w_j \mathcal{KL}(G_j || G_{merge(i,j)}) \tag{3.16}$$

This cost function can be understood by noting that it corresponds to an upper bound on the expected decrease in the log-likelihood of a point drawn from $w_i G_i + w_j G_j$ when the latter is replaced by $G_{merge(i,j)}$ (see Appendix A for discussion/derivation).

Choosing the pair of components that minimises this cost function requires the calculation of a matrix containing the merging costs for all unique pairs of components. This matrix is initialised when the maximum number of components is reached, requiring $\frac{N_{max}(N_{max}-1)}{2}$ calculations of the merging cost function: it is evident that the computational complexity of this calculation is $O(n^2)$, increasing quadratically with the final number of components N_{max}. However, this $O(n^2)$ cost matrix initialisation is a one-off calculation: thereafter, the computational complexity of maintaining this matrix is $O(n)$ as the matrix only needs to be updated for entries corresponding to merged/new components (see next paragraph).

Model updating procedure When a new training example x_{new} arrives, a new component $G_{N_{max}+1} = \{x_{new}, H_{final}\}$ is added to the mixture model, with weight $\frac{1}{N_{ex}+1}$ (where N_{ex} is the total number of training examples received before the current one). To restore the sum of the weights in the model to unity, the weights of the pre-existing components are rescaled by a factor of $\frac{N_{ex}}{N_{ex}+1}$. The merging costs for the new component are then calculated (requiring N_{max} calculations of the cost function), and the cost matrix is augmented with a new row/column corresponding to the new component. Using the newly updated cost matrix, the pair of components with the lowest merging cost is chosen:

$$\{G_i, G_j\} = \arg \min_{G_i, G_j : i \neq j} \big(cost(G_i, G_j)\big)$$

This pair of components is merged (according to Equation 3.14), thus reducing the model to its original number of components (N_{max}). Merging costs are then calculated for the new merged component, requiring a further $N_{max} - 1$ calculations of the cost function; a total of $2N_{max} - 1$ evaluations of this function is therefore required for every new training example added.

The procedure for updating the merging cost matrix is illustrated in Figure 3.2. As shown, two merging scenarios are possible: either a pair of existing components is merged, or the newly added kernel is merged with an existing component. Whenever a pair of components is merged a vacant index is produced in the cost matrix (and

mixture model data structure) corresponding to one of the merged components. If the newly added kernel has been merged with an existing component, its empty index $N_{max}+1$ is simply removed; otherwise, if two existing components have been merged, the new kernel assumes the vacant index created by the merge.

3.5.2 Kernel Parameter Choice

Kernel covariance matrix. An important remaining issue is to determine an appropriate covariance matrix for the kernel functions which are added to the mixture throughout training (and entirely define the model before merging starts.) We determine this matrix from the training data observed before the merging phase commences. As with all previous applications of KDE to the problem of outlier detection [11, 139, 173], a uniform spherical covariance matrix is assumed, defined by a single parameter σ as follows (where I^d is the $d \times d$ identity matrix):

$$K(\sigma) = I^d \cdot \sigma^2 \tag{3.17}$$

While it is in principle possible to estimate full kernel covariance matrices using sampling techniques, as proposed by Zhang et al. in [177], such methods would be prohibitively time-consuming given the proposed online learning scenario.

Leave-one-out criterion for estimating σ. An intuitive approach, given that the set of Gaussian kernel functions constitutes a generative model, is to choose a value for σ that maximises the model likelihood. A problem with this approach is that, since there is a kernel function centred on every data point, the model likelihood is maximised when $\sigma = 0$. However, this problem is addressed by the "leave-one-out" likelihood criterion proposed by Duin in [37], which works by calculating the probability density for each training example x_n given a model constructed from all others $\forall x \neq x_n$, yielding the following log likelihood function for the dataset:

$$L_{sum}(X|\sigma) = \sum_{n=1}^{N} \log \left(\frac{1}{(2\pi\sigma)^{\frac{d}{2}}} \cdot \frac{1}{N-1} \cdot \sum_{\forall x \neq x_n} e^{-\frac{||x_n - x||^2}{2\sigma^2}} \right) \tag{3.18}$$

This has recently been shown to be a good model selection criterion for multivariate kernel density estimation by Zhang et al. in [177]. A potential problem with this function is that it could be unduly influenced by outlying samples: in particular, by assigning low likelihoods to values of σ which result in near-zero probability density values for a very small number of outlying samples. Since the training data may be sparsely distributed in the early stages of training, this is an important consideration. A potential solution to this problem is to take the sum of those log likelihood values

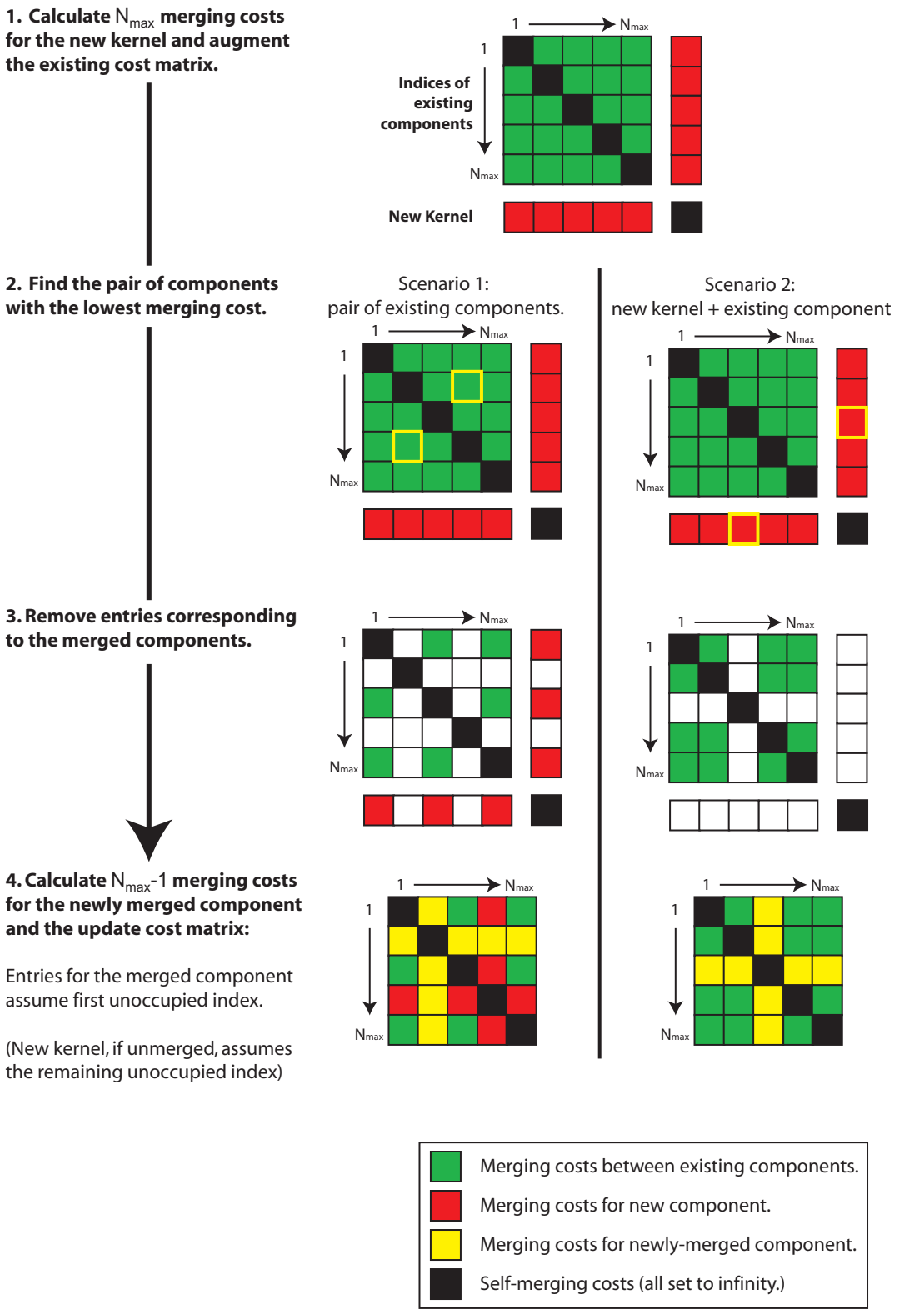


Figure 3.2: Illustration of the procedure for updating the merging-cost matrix. It should be noted that this matrix is symmetrical: thus for each merging operation a total of $2N_{max} - 1$ entries (first corresponding to the new component and then the merged component) must be calculated. (See text for further details.)

lying within the *interquartile range*, thereby omitting the highest and lowest 25% of log likelihood values from the sum:

$$L_{IQR}(X|\sigma) = \sum_{\forall x_n \in IQR\{x_1,\ldots x_N\}} \left(\log \left(\frac{1}{(2\pi\sigma)^{\frac{d}{2}}} \cdot \frac{1}{N-1} \cdot \sum_{\forall x \neq x_n} e^{-\frac{||x_n - x||^2}{2\sigma^2}} \right) \right) \tag{3.19}$$

How is σ determined? The maximum-likelihood value of σ is estimated by evaluating the chosen likelihood function for a range of 200 linearly spaced values between the smallest and largest nearest neighbour distances observed so far (where N is the current number of training instances), and choosing the maximum. We re-estimate the value of sigma every time a new example is added to the dataset X:

$$\sigma_{est}(N) = \arg\max_{\sigma}(L(\{x_1,\ldots,x_N\}|\sigma))$$

Noting that this estimation procedure is likely to give misleading results when only a small number of training instances have been observed, we initially set σ to an arbitrarily small value of $\sigma_{min} = 0.001$. The data-driven estimate σ_{est} is then phased in gradually as training progresses, by weighting σ_{min} and σ_{est} in the following manner (where N_{max} is the maximum number of kernels that will be added to the model, and $N \leq N_{max}$ is the number of kernels currently added):

$$\sigma(N) = \sigma_{min}(1 - \frac{N(N-1)}{N_{max}(N_{max}-1)}) + \sigma_{est}.(\frac{N(N-1)}{N_{max}(N_{max}-1)})$$

The weighting factor $\frac{N(N-1)}{N_{max}(N_{max}-1)}$ expresses the number of unique pairings of the data points that have been observed so far, $\frac{1}{2}N(N-1)$, as a proportion of the total when the merging phase starts, $\frac{1}{2}N_{max}(N_{max}-1)$. When the merging phase starts, the kernel covariance matrix is then fixed to:

$$K_{final} = I^d.\sigma(N_{max})$$

Section 3.6.2 examines the behaviour of the different likelihood functions, while Section 3.6.5 compares their impact on classification performance.

How does the kernel covariance affect the resulting model? In the case of straightforward kernel density estimation (as opposed to density estimation achieved through the merging rule described in Section 3.5.1) the kernel covariance K effectively acts as a model complexity parameter. If defined as $K = I^d.\sigma$, then large values of σ result in smoother density estimates (which are more invariant to the distribution of the

underlying data sample) while smaller values result in more complex density functions (which more closely reflect the nuances of the underlying data sample) [12]. For Gaussian mixture models resulting from the proposed merging procedure, the kernel function has a more complex role, affecting the resulting density estimate in two different ways.

It is simple to show (see Appendix A for derivation) that for any mixture component in the resulting model representing multiple examples $\{z_1, \ldots z_n\}$ (ie. which has been involved in merging operations) the resulting covariance matrix can be expressed as

$$Cov(\{z_1, \ldots z_n\}) + K \tag{3.20}$$

where $Cov(\{z_1, \ldots z_n\})$ is the maximum likelihood estimate for the covariance of $\{z_1, \ldots z_n\}$, and K is the kernel covariance matrix. In this light, it is clear that K (and hence σ) acts as a bias term contributing to the spread of each component in the mixture model. So, as with kernel density estimation, larger values of σ will result in smoother density estimates which are less specific to the underlying data.

Given a model where all mixture components represent multiple examples, it is therefore possible to remove the bias associated with the kernel covariance K by simply subtracting it from each component in the model. This, however, does not fully remove the impact of K on the model. Given the merging procedure described in Section 3.5.1, the relative cost (as measured by KL divergence) of merging a newly added kernel with an existing mixture component, compared to that of merging a pair of existing components, is clearly affected by the kernel covariance matrix K. In particular, smaller K would reduce the cost of merging a new kernel with an existing component, while larger K would increase this cost, encouraging pairs of existing components to be merged.

The two-fold impact of K on the resulting mixture model is examined in Section 3.6.6, where classification performance is measured for two choices of σ, before and after the subtraction of K from each mixture component.

3.5.3 Outlier Detection Strategies

The preceding sections outlined a way to incrementally generate a Gaussian mixture model from a stream of data. In order to use this mixture model for outlier detection, a classification boundary must be defined. We explore two possible strategies for defining such a boundary, summarised as follows (the impact of these strategies on classification performance is examined in Section 3.6.5).

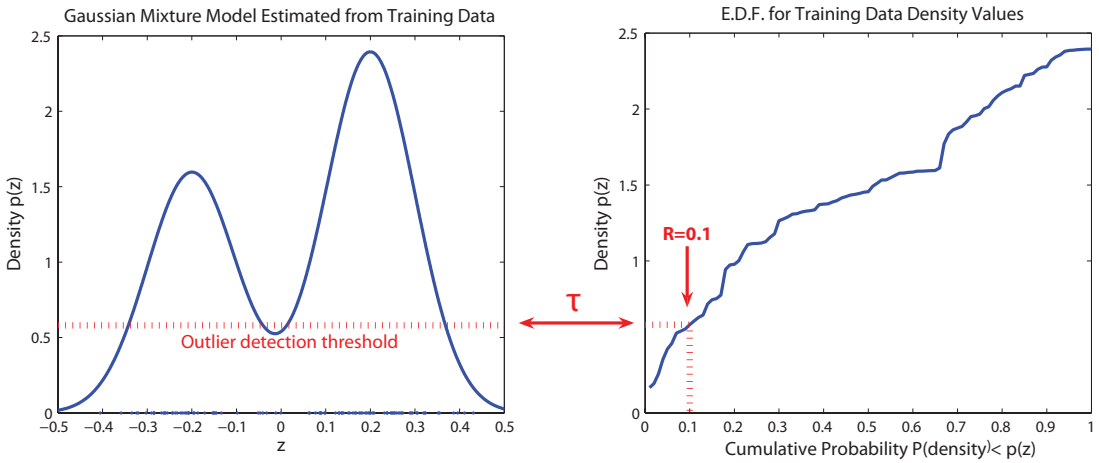


Figure 3.3: Setting a density threshold for outlier detection (see text for description).

Density Threshold The most intuitive way to define such a boundary is to place a threshold on the value of the estimated probability function defined by Equation 3.10; as noted in [11], this is equivalent to modelling outliers with a uniform distribution. Thus, if a test example has a probability density value which falls below a threshold τ, it is classified as an outlier:

$$p(z) \begin{cases} < \tau & \rightarrow \quad \text{outlier} \\ \geq \tau & \rightarrow \quad \text{normal} \end{cases} \tag{3.21}$$

The remaining problem is to choose an appropriate value for τ. The strategy adopted here is to set a threshold which classifies the portion of the training data with the lowest $(R \times 100)\%$ of density values as outliers. This is achieved by estimating the Empirical Distribution Function (EDF) [159] for the density values assigned to training data $X = \{x_1, \ldots, x_N\}$ by the current model (where $I(\cdot)$ is the indicator function):

$$\hat{P}_X\left(\text{density} \leq p(z)\right) = \frac{1}{N} \sum_{i=1}^{N} I\left(p(x_i) \leq p(z)\right)$$

The set of observed density values $p(x_i)$ and their corresponding cumulative probability values $\hat{P}_X\left(\text{density} \leq p(x_i)\right)$ can then be linearly interpolated to find the value τ corresponding to the desired threshold $\hat{P}_X\left(\text{density} \leq \tau\right) = R$. This process is illustrated in Figure 3.3 where a threshold corresponding to $R = 0.1$ is determined for a one-dimensional Gaussian mixture model.

Extreme Value Threshold An alternative strategy for detecting outliers with respect to a Gaussian mixture model has been proposed by Roberts in [109, 110], where a model from Extreme Value Theory [29] is used to assign a probability value to the deviation of a new example from each component in the model.

The key insight is as follows: for a series of separate sets of M samples drawn from a Gaussian distribution, the deviation of the most extreme example in each set has been shown to follow the Gumbel distribution. The parameters of this distribution depend on the number of samples M, so that for larger values of M the expected deviation of the most extreme sample increases. Thus, for a single multivariate Gaussian estimated from M samples, the probability of observing a new (extreme) sample whose Mahalanobis distance exceeds d is given by the Gumbel distribution function as follows:

$$P(D > d|M) = 1 - \exp\left(-\exp\left(-\frac{d - \alpha(M)}{\beta(M)}\right)\right) \tag{3.22}$$

The location $\alpha(M)$ and scale $\beta(M)$ parameters of this distribution depend on M as follows [109, 110]:

$$\alpha(M) = (2\ln M)^{\frac{1}{2}} - \frac{\ln\ln M + \ln 2\pi}{2(2\ln M)^{\frac{1}{2}}} \qquad \beta(M) = (2\ln M)^{\frac{1}{2}}$$

To use Equation 3.22 to detect outliers with respect to a Gaussian mixture model, a strategy shown to give good results in [109, 110] is, given a test example z, to evaluate its Mahalanobis distance $d_k(z) = \sqrt{(z-\mu_k)^T\Sigma_k(z-\mu_k)}$ from each component $k \in \{1 \ldots N_{max}\}$ in the model. The Mahalanobis distance $d_{k*}(z)$ from the closest mixture component $k* = \arg\min_k(d_k(z))$ is then used to determine whether z is an outlier, according to a threshold ρ placed on Equation 3.22 as follows:

$$P(D > d_{k*}(z)|M_{k*}) \begin{cases} < \rho & \rightarrow \quad \text{outlier} \\ \geq \rho & \rightarrow \quad \text{normal} \end{cases} \tag{3.23}$$

The value M_{k*} denotes the number of training samples for which the $k*$th mixture component is responsible: this can be estimated as $M_{k*} = N \cdot w_{k*}$ where w_{k*} denotes the prior probability associated with $k*$ and N denotes the total number of training examples observed so far [109, 110].

An advantage of this approach is that - despite operating on a component-wise basis - a single threshold value ρ will effect more conservative classification boundaries for components representing fewer observations; furthermore it alleviates the need for continually estimating an empirical distribution function over the observed training density values.

3.6 Experiments

This section examines the performance of the proposed algorithm on a variety of datasets. The datasets and their associated classification problems are described in Section 3.6.1, while the remainder of the section presents a series of experiments based on these datasets.

3.6.1 Datasets

This section describes the datasets used to test the proposed algorithm. The chosen datasets all contain instances of continuous-valued vectors corresponding to two or more distinct classes. For the purpose of testing the proposed one-class learning algorithm, a single class of data is arbitrarily chosen from each dataset to represent "normal" data, while the remaining classes are used to represent outliers.

In all the experiments described, the data used to train the algorithm are rescaled linearly so that the elements in each vector all lie between -1 and 1. The rescaling factors calculated for the training data are applied to the test data, thus allowing us to assess the performance of the proposed algorithm on unseen data. It should be noted that rescaling occurs as an offline step using the whole set of training data. While this appears to violate the premise of incremental learning, it is based on the reasonable assumption that feasible limits for the measurements represented by each vector element are likely be known *a priori* in most circumstances. (Indeed, it would also be realistic to estimate this scaling from a preliminary sample of data.)

Synthetic Spiral Dataset

In order to visualise the behaviour of the proposed algorithm, a synthetic two dimensional dataset was created by drawing random samples within a 3×3 square region. Samples lying with a spiral shaped region, defined by the following inequality, are used to define a "normal" class (where $a = 0.7$, $w = 0.1$, and r and θ are polar coordinates corresponding to radius and angle):

$$a\theta < r \leq a\theta + w$$

Similarly, samples lying outside the spiral-shaped region are used to define an outlier class. Figure 3.4 shows the resulting dataset, from which 2500 samples inside the spiral region are used as training data, and a further set of 5000 samples (2500 inside and 2500 outside) are used as test data.

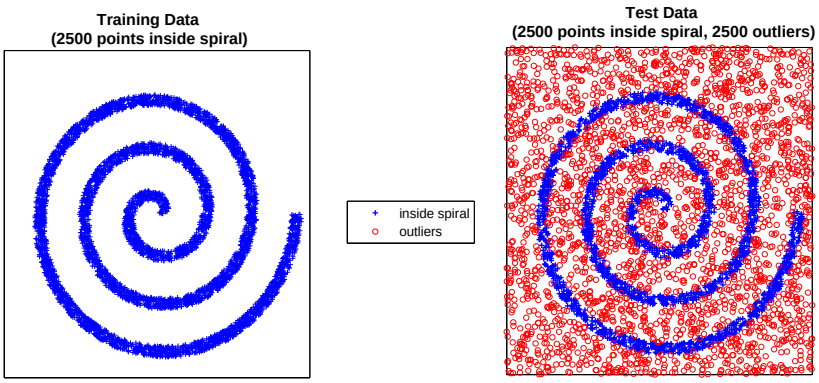


Figure 3.4: Synthetic spiral dataset (see text for description).

Real Datasets

In order to test the proposed algorithm, a variety of datasets were selected from the UCI Machine Learning Repository [6], listed as follows. The chosen datasets were arbitrarily selected on the basis of: containing real-world data, having fewer than 40 continuous valued attributes, and multiple classes where at least one contains more than 250 instances.

1. The *Wisconsin Breast Cancer Database*, which contains 699 (9-dimensional) vectors corresponding to cytological measurements. We use the 458 examples of benign cells as the normal class and the 241 examples of cancer as outliers.

2. The *Letter Recognition Database*, which contains 20,000 (16-dimensional) parametrisations of printed letters, with 26 classes corresponding to the alphabet. We use the 789 examples of the letter 'A' as the hypothetical normal class, and all other classes as outliers.

3. The *STATLOG Landsat Satellite Database*, which contains 6435 (36-dimensional) vectors corresponding multispectral images of 6 different types of ground coverage. We use the 1533 examples of 'red soil' as the normal class and all others as outliers.

4. The *STATLOG Shuttle Database*, which contains 58,000 (9-dimensional) vectors corresponding to readings from radiators within a NASA space shuttle. The dataset contains 7 classes corresponding to different activity patterns. We use the 45586 examples of the most common class 'Rad Flow' as the normal class, and all others as outliers.

5. The *Pima Indians Diabetes Database*, which contains 768 (8 dimensional) vectors corresponding to medical diagnostics used to test for diabetes. We use 500 ex-

amples corresponding to non-diabetics as the normal class, while the remaining 268 examples corresponding to cases of diabetes are used as outliers.

6. The *Pen-Based Recognition of Handwritten Digits Data Set*, which contains 10992 (16 dimensional) vectors corresponding to resampled pen trajectories from 10 classes of handwritten digit. We use the 1143 instances of the digit '0' as the normal class, and all others as outliers.

In each case no pre-processing takes place other than the rescaling step mentioned earlier; however, any instances with missing values are discarded. Furthermore, many of the datasets are split into suggested training/test sets intended for testing multi-class supervised learning algorithms. In order to test our one-class learning algorithm, we combine these sets and use 90% of the examples from the chosen normal class for training, while retaining the remaining 10% for testing, together with the outlier class/classes. For the shuttle dataset, where the selected 90% portion of available training data contains a very large number of examples (41027), only the first 2500 examples are used, enabling experiments to be conducted in reasonable time.

To give a preliminary impression of class-separability, Figures 3.5 and 3.6 show two different 2-dimensional visualisations for each of the preceding datasets, corresponding to linear projections determined using Principal Components Analysis (which gives the 2 dimensions which capture maximum variation) and Linear Discriminant Analysis (which gives the 2 dimensions yielding the best within-vs-between class scatter). The Shuttle and Pima Indians datasets - in contrast to the other datasets considered - do not appear to be well separated for either of the 2D projections, indicating that these datasets may present the most challenging test for the proposed one-class learning algorithm.

3.6.2 How do the kernel parameter selection methods behave?

This section presents a set of preliminary experiments documenting the behaviour of the two strategies described in Section 3.5.2 for estimating the kernel covariance parameter σ.

The values of σ yielded by attempting to maximise the log-likelihood functions described by Equations 3.19 and 3.18 were recorded for different quantities of training data, ranging from 2 examples to 250 examples, for each dataset described in Section 3.6.1. The purpose of this experiment was to compare the behaviour of the two methods described in Section 3.5.2, and to determine how many items of training data need to have been observed before a reasonable value of σ can be obtained.

For each dataset, Figure 3.7 shows how the estimated value of σ changes as more

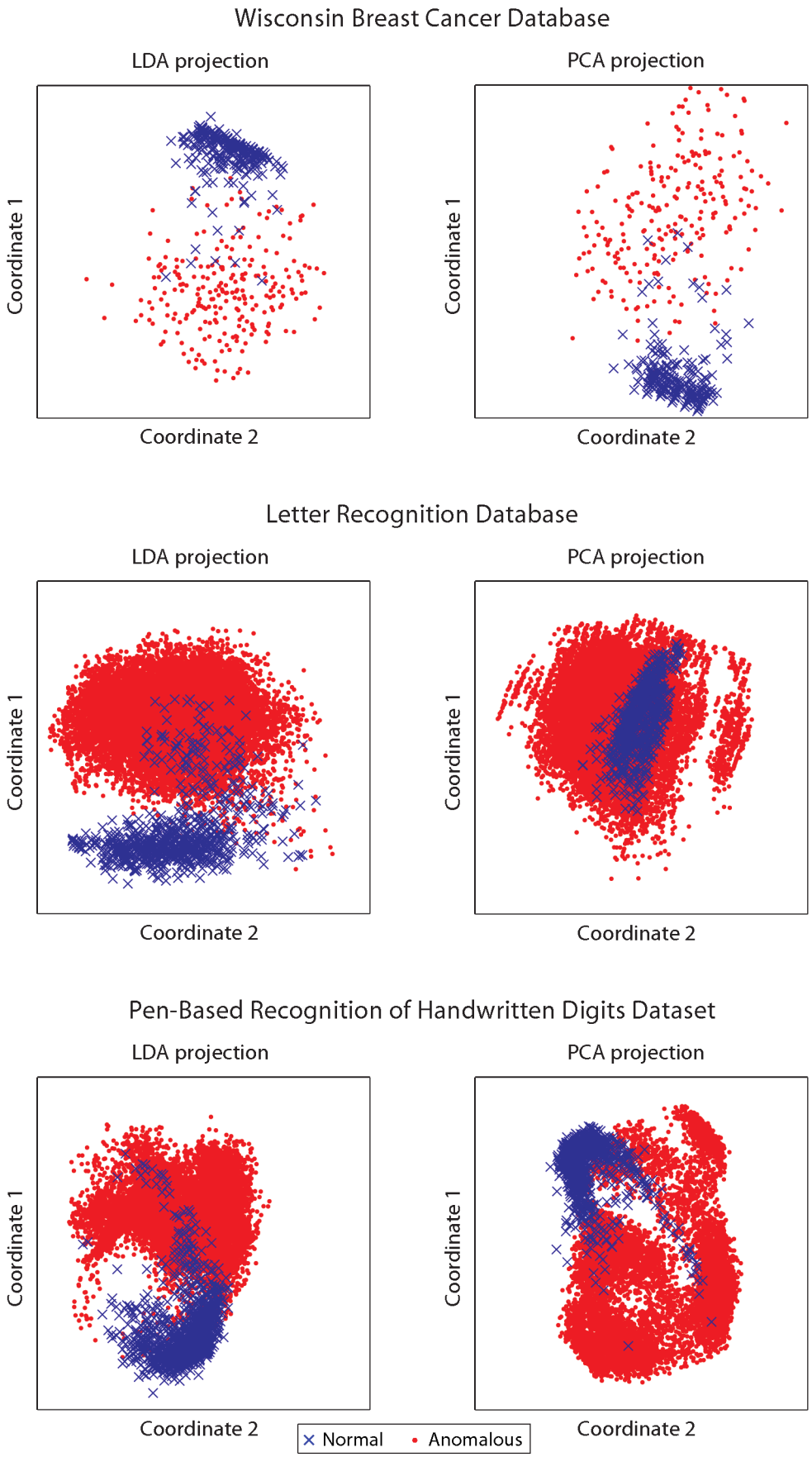


Figure 3.5: 2D Visualisation of real test datasets (Cancer, Letter and Pen-Digits datasets) using PCA and LDA projections. (See text for discussion.)

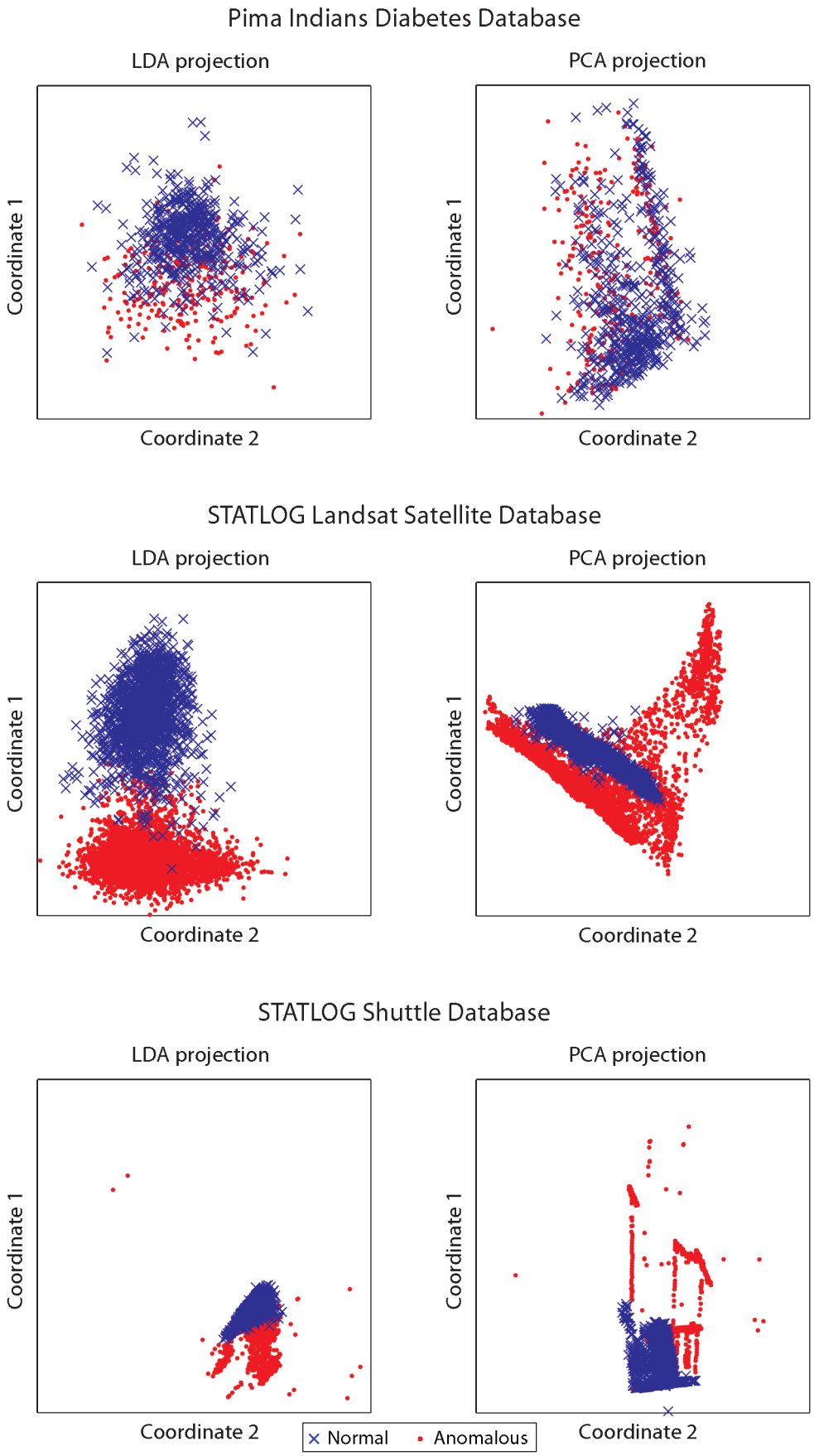


Figure 3.6: 2D Visualisation of real test datasets (Pima Indians, Satellite and Shuttle datasets) using PCA and LDA projections. (See text for discussion.)

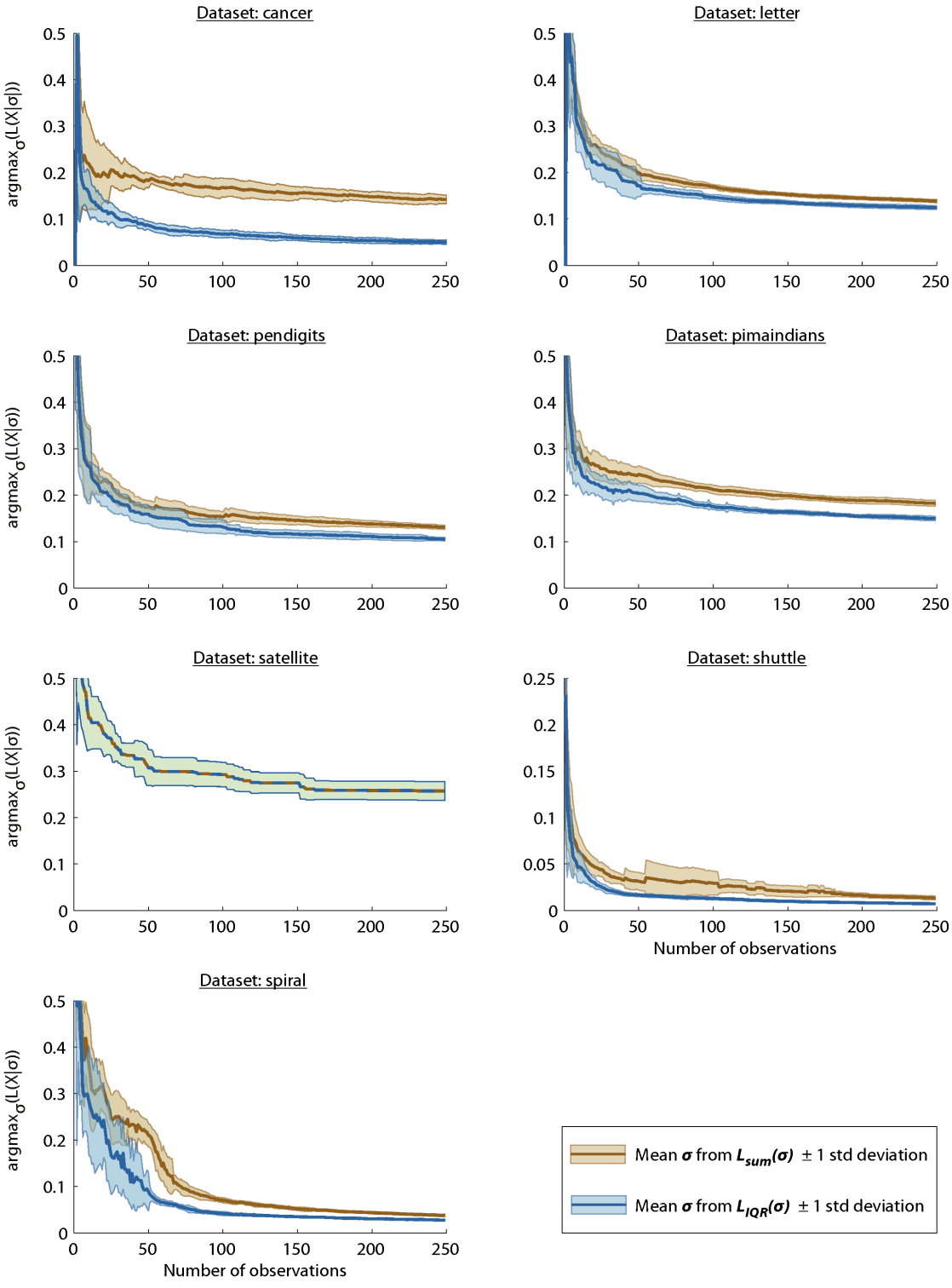


Figure 3.7: Behaviour of model parameter estimation techniques for kernel covariance matrix for increasing quantities of training data (see text for discussion).

data points are observed. In each plot, the mean estimated σ vs number of observations curves (generated for 10 different random selections of 250 training data) are shown, accompanied by shaded regions indicating ± 1 standard deviation at each point. The orange curves indicate the values of σ estimated using $L_{sum}(X|\sigma)$, the sum of leave-one-out log-likelihood values (Equation 3.18), while the blue curves indicate the values estimated using $L_{IQR}(X|\sigma)$, the sum of those values lying within the interquartile range (Equation 3.19).

General behaviour. In both cases it is clear that the estimated value of σ rapidly decreases as more training data is observed. In most cases the variance of the estimated values (taken for different random selections of training data) also decreases. After an initial rapid drop in the estimated value of σ, both measures appear to stabilise: at this stage they continue to decrease, but at a much slower rate. Noting that the cost of evaluating both functions increases quadratically in the number of training examples, it rapidly becomes impractical to continue estimating σ. In this light, we use 100 training examples to estimate σ in the experiments described in the following sections, noting that the phase of gradual decrease appears to have been reached at this point in each of the plots shown in Figure 3.7.

How do the methods differ? Although the pattern of rapid decrease followed by very gradual decrease is shared by both methods for estimating σ, it is clear from the plots shown in Figure 3.7 that the two methods behave differently in all cases (except for Satellite dataset, for which the two methods give identical results).

As discussed in Section 3.5.2, the underlying rationale for using $L_{IQR}(X|\sigma)$ (the sum of log-likelihood values lying within the interquartile range) rather than $L_{sum}(X|\sigma)$ (the sum of all log-likelihood values) is to prevent a small number of isolated data points from disproportionately increasing the estimated value of σ. Indeed, this reasoning is supported by the results shown in Figure 3.7, which indicate that the σ values estimated using $L_{sum}(X|\sigma)$ are almost always higher than those estimated using $L_{IQR}(X|\sigma)$; furthermore, in some cases (eg. for the spiral dataset), the values estimated using $L_{IQR}(X|\sigma)$ appear to stabilise more quickly.

Noting the observed difference in the behaviour of the two methods, it may be sensible to adopt $L_{IQR}(X|\sigma)$, which favours smaller values of σ, in situations where misclassified normal examples are more tolerable than misclassified outliers. Conversely $L_{sum}(X|\sigma)$ may be favourable in situations where "false alarms" due to misclassified normal examples are more costly than the misclassification of a small number of outliers. The classification performance resulting from these methods, and the various

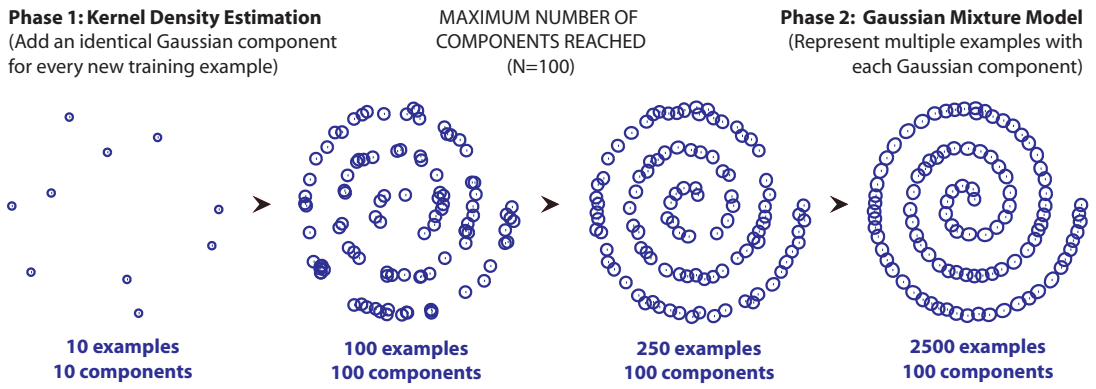


Figure 3.8: Mixture model organisation during training on the synthetic spiral dataset (see text for description).

outlier detection techniques discussed in Section 3.5.3, is examined in the following sections.

3.6.3 How does the proposed algorithm behave?

This section presents experiments illustrating the behaviour of the proposed algorithm in its simplest form. The sum of all leave-one-out log-likelihood values (Equation 3.18) is used to estimate σ, while a classification threshold is determined by placing a threshold on the resulting probability density function (Equation 3.21).

In this section, and for all subsequent experiments, the proposed algorithm is tested with the maximum number of components N_{max} set to 100. This is based on the reasonable assumption that a set of 100 Gaussian components is more than adequate to model most datasets, and the observation that the estimated values of σ appear to stabilise by this point (see Section 3.5.2).

Visualisation of mixture model organisation. To ascertain that the proposed algorithm behaves as expected, the organisation of the resulting mixture model was recorded during training on the synthetic two dimensional spiral dataset described in Section 3.6.1. Figure 3.8 shows both the position, and 1 standard deviation boundary, of each component at several stages during training: at 10 and 100 observations, during the kernel density estimation phase, and then at 250 and 2500 observations during the merging phase. It is clear from the model organisations recorded for 100, 250 and 2500 observations that the proposed merging procedure yields a progressive improvement in the representation of the underlying training data distribution as more examples are observed.

How does classification performance change during training? To further characterise the behaviour of the proposed algorithm, classification performance was measured at regular intervals during training on each of the datasets described in Section 3.6.1. In order to define a classification boundary for this experiment, we use the density threshold approach described in Section 3.5.3, setting a threshold τ, where $\hat{P}_X(\text{density} \leq \tau) = 0.9$, so that approximately 10% of the normal training data is deliberately misclassified. It should be noted that this classification threshold may not necessarily provide the best classification performance (in Section 3.6.4, ROC curves [40] are used to fully characterise the range of performance levels that can be obtained); nonetheless, it provides a useful characterisation of the behaviour of the proposed algorithm.

Figure 3.6.3 illustrates the changing performance of the proposed algorithm for each dataset. In each case the proposed algorithm was trained on 10 different random permutations (90% for training, 10% for testing) of the designated class of normal data. As each training example is added, the resulting classification performance is measured, averaged over 10 trials. In each plot, the blue curve shows the proportion of the normal test data correctly recognised as normal (True Positive rate), while the red curve shows the proportion of the designated outlier data misclassified as normal (False Positive rate): both curves are accompanied by shaded regions indicating ± 1 standard deviation at each point. Furthermore, in each plot a vertical dashed line indicates the start of the merging phase, which occurs when 100 components have been added to the model.

In general, classification performance can be seen to improve as more training examples are observed, with a high True Positive Rate (TPR) and low False Positive Rate (FPR) achieved at the end of training (see Table 3.2 for relevant figures). The classification performance obtained on the Pima Indians dataset is one principal exception to this trend: in this case FPR increases steadily as TPR increases, meaning that generalisation to the normal class is occurring at the expense of the misclassification of a large proportion of the outlier class; after training on this dataset, a classification rate (corrected for differences in class proportions) of $57.7 \pm 1.13\%$ is obtained for the test data, which is only marginally better than random guessing (ie. 50%). In a similar vein, FPR appears to increase during training on the Shuttle dataset; however, in this case, a reasonable level of classification performance (a class-corrected classification rate of $88.1 \pm 1.08\%$) is obtained at the end of training. The only other dataset for which a high (ie. $\geq 1\%$) False Positive Rate is observed is the Spiral dataset; in this case, however, performance improves with training, with FPR falling steadily as more training examples are observed.

Where the algorithm appears to fail, it is possible to draw one of two conclusions: one possibility is that the normal and outlier classes overlap significantly in feature space, thereby preventing the probability density of the normal class from providing a useful classification boundary; alternatively poor classification performance may reveal an underlying failing of the proposed algorithm. The 2-dimensional PCA/LDA projections shown in Figures 3.5 and 3.6 lend some support to the former hypothesis, as the Pima Indians and Shuttle datasets appear to be the least well-separated. Section 3.6.4 attempts to shed further light on this question, by comparing the performance of the proposed algorithm with an existing state-of-the-art approach.

Dataset	Classification (%)	False Pos. Rate (%)	True Pos. Rate (%)
cancer	94 ± 2.35	0 ± 0	88 ± 4.71
letter	91.7 ± 2.23	0.201 ± 0.0615	83.7 ± 4.44
pendigits	94.2 ± 1.73	0 ± 0	88.3 ± 3.46
pimaindians	57.7 ± 1.13	72.6 ± 1.14	88 ± 2.31
satellite	94.3 ± 1.08	0.867 ± 0.0474	89.5 ± 2.16
shuttle	88.1 ± 1.09	13.0 ± 2.43	89.3 ± 0.83
spiral	89.0 ± 0.871	8.55 ± 1.95	86.5 ± 0.977

Table 3.2: Classification performance at the end of training using proposed algorithm (using $L_{sum}(X|\sigma)$ and density threshold). The left hand column shows an aggregate measure of classification performance corrected for differences in class proportions (ie. $C = \frac{1}{2}[TPR + (1 - FPR)]$.)

3.6.4 How well does the proposed algorithm perform?

This section attempts to place the behaviour of the proposed algorithm in context, by comparing its performance with that of the Incremental Support Vector Data Description algorithm [146, 74] described in Section 3.3 (referred to hereafter as IncSVDD).

We use a freely available implementation of IncSVDD contained in the *DDtools* MATLAB toolbox [140]. In all tests we use a Gaussian kernel function and optimize the kernel parameter (for the whole training dataset) using the *consistent_occ* function from [140], which implements the consistency-based model selection criterion proposed in [143]. We use this criterion to find the best kernel parameter value from a set of 20 values uniformly distributed between the shortest and longest Euclidean distances observed within the dataset. To improve this initial estimate, we then repeat the parameter optimization process on a finer scale, for a further 20 values surrounding the optimal parameter from the first set.

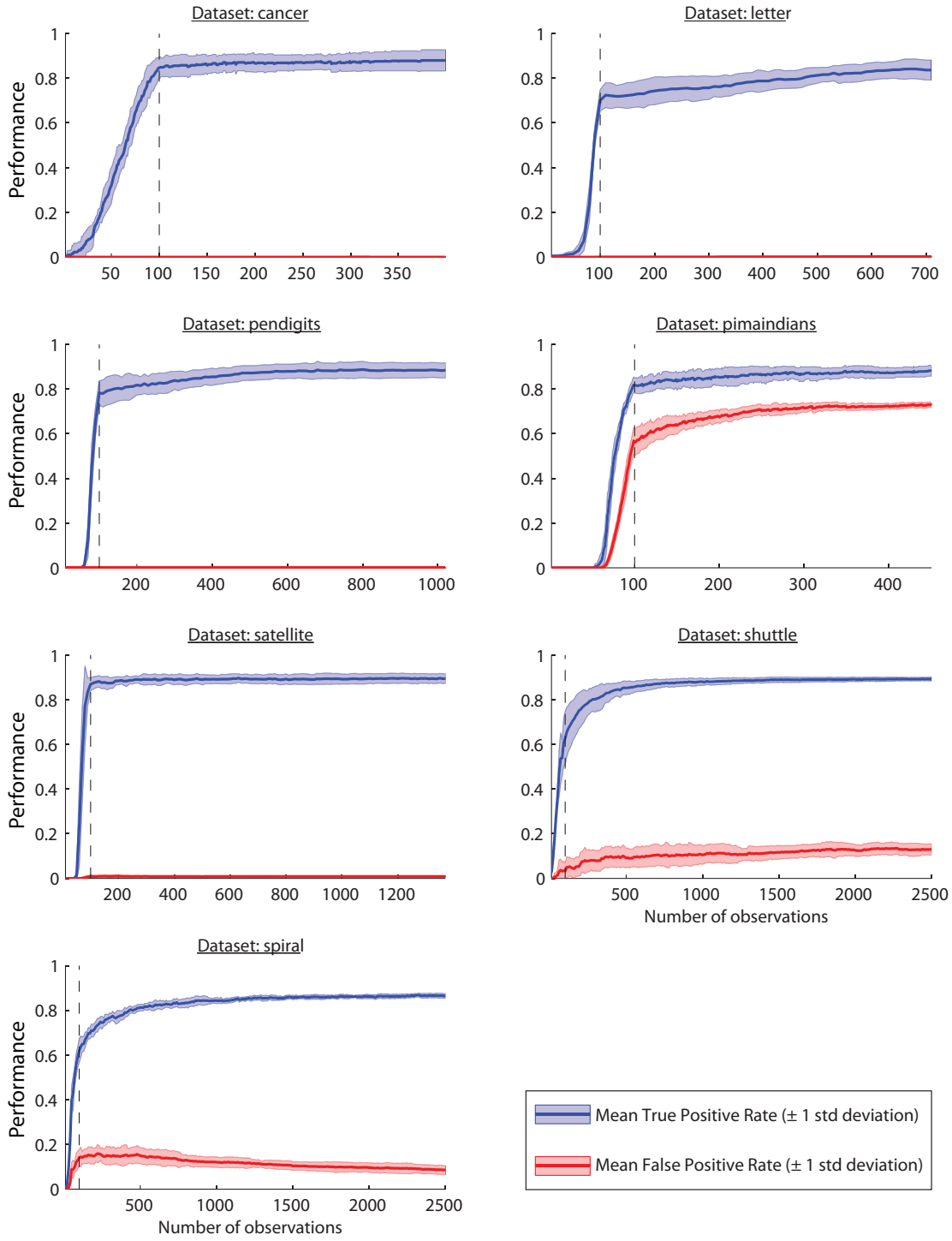


Figure 3.9: Classification performance vs number of observations, for a mixture model generated using the proposed algorithm. In this case, the maximum number of components is set to 100, σ is estimated using $L_{sum}(X|\sigma)$, and a classification boundary is defined by placing a density threshold at τ where $\hat{P}_X(\text{density} \leq \tau) = 0.9$. (See text for discussion).

Dataset	cancer	letter	pendigits	pimaindians	satellite	shuttle	spiral
Proposed Alg.	0.9208	0.9571	0.9918	0.1628	0.8193	0.6906	0.3474
IncSVDD	0.8709	0.2566	0.9903	0.0144	0.7469	0.6523	0.0004

Table 3.3: Comparison of IncSVDD with the proposed algorithm (using $L_{sum}(X|\sigma)$ and density threshold). This table shows mean values of $50 * AUC_{\text{FPR} \leq \frac{1}{50}}$ taken over 10 trials for each dataset: this value measures the area under the ROC curve for FPR values below $\frac{1}{50}$, and is normalised to range between 0 and 1.

As with the previous section, the most basic variant of the proposed algorithm is used: the sum of all leave-one-out log-likelihood values (Equation 3.18) is used to estimate σ, while a classification threshold is determined by placing a threshold on the resulting probability density function (Equation 3.21). We compare the performance of the two algorithms at the end of training on each dataset by generating ROC (Receiver Operating Characteristic) curves which document the range of possible trade offs between TPR and FPR that can be obtained by varying the classification threshold for each classifier. (For IncSVDD, this threshold corresponds to Euclidean distance from the centre of a high-dimensional hypersphere defined by a weighted subset of training data in conjunction with a kernel function [142, 146].)

Figure 3.10 shows the ROC curves obtained for the two algorithms at the end of training on each dataset. Each ROC curve displayed is the mean ROC curve obtained for 10 trials (corresponding to different random selections of training data), generated by vertically averaging TPR values for each corresponding FPR value as described in [40]. For almost all datasets, the mean ROC curve obtained for the proposed algorithm lies above that obtained for IncSVDD, for FPR values below 0.02. In other words, it correctly classifies a greater proportion of normal data than IncSVDD for those thresholds which allow no more than 1 in 50 outliers to be misclassified. This observation can be quantified numerically by measuring the area under the ROC curves in the region where FPR ≤ 0.02 (see Table 3.3 for resulting values), yielding a difference which is statistically significant ($p = 0.0156$ Wilcoxon Signed-Rank Test [86]).

In general the two algorithms yield comparable performance, with the exception of the Spiral and Letter datasets, where the proposed algorithm outperforms IncSVDD by a large margin, and the Shuttle dataset, where IncSVDD yields better classification performance. It is also interesting to note that IncSVDD did not outperform the proposed algorithm for the Pima Indians dataset, lending weight to the hypothesis - posed in the preceding section - that the normal and outlier classes from this dataset may overlap in feature space.

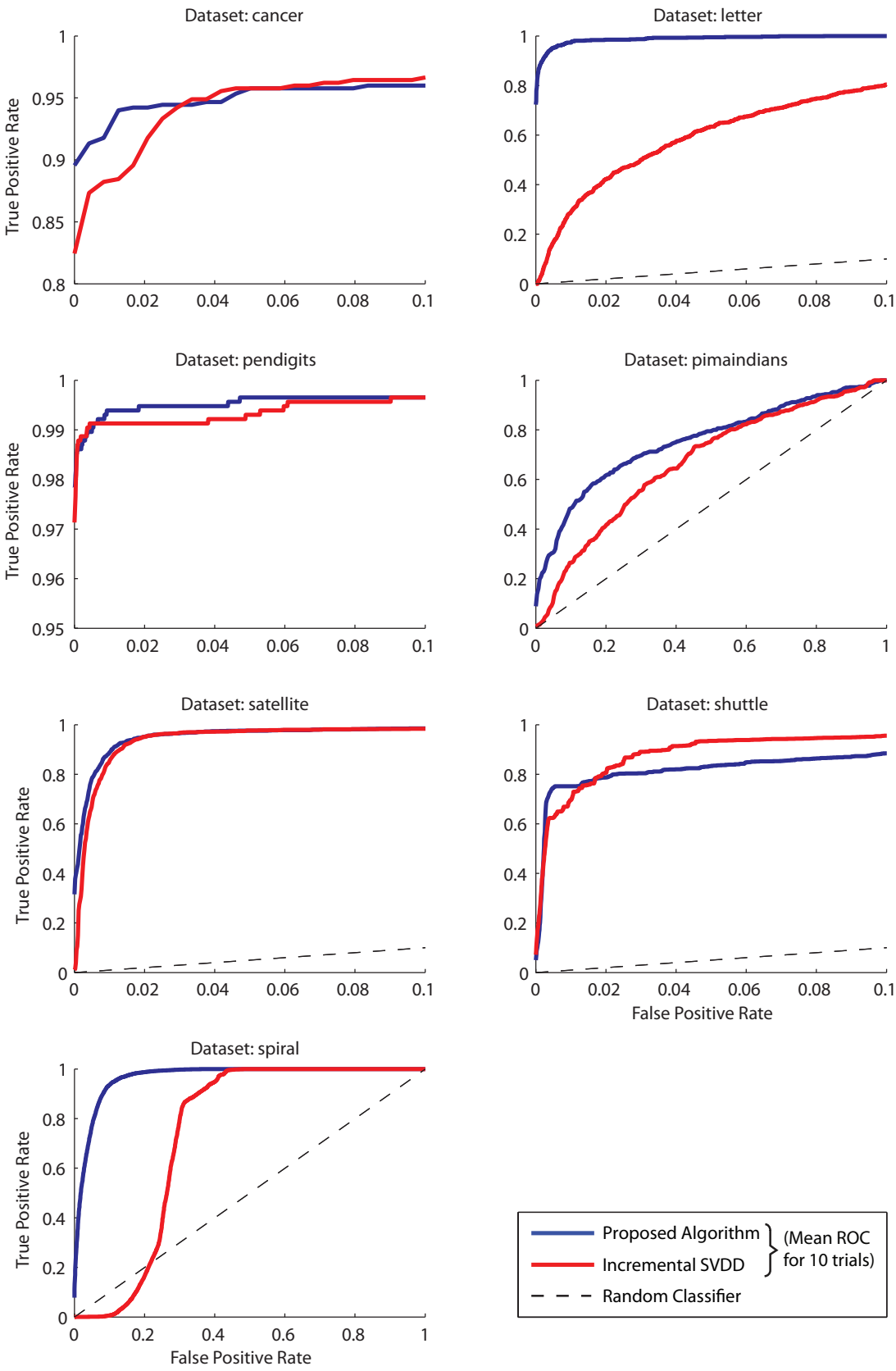


Figure 3.10: Comparison of ROC curves for Incremental SVDD and the proposed algorithm (using $L_{sum}(X|\sigma)$ and density threshold). Note that the range of the axes differs between plots. (See text for discussion.)

3.6.5 Which configuration works best?

As potential improvements to the basic configuration of the proposed algorithm (used to generate the preceding results), Sections 3.5.2 and 3.5.3 suggested alternative methods for selecting the σ parameter (using the interquartile range of leave-one-out log-likelihood values), and defining a classification boundary (using the component-wise extreme value statistic proposed by Roberts [109]). This section examines the impact of these alternative methods on the resulting classification performance.

As for the previous section, we generate ROC curves (vertically averaged over 10 trials) to evaluate classification performance at the end of training on each dataset. These curves, shown in Figure 3.11, are generated for four different configurations of the proposed algorithm: models with density-based classification boundaries (thick and thin blue curves for σ estimated using $L_{sum}(X|\sigma)$ and $L_{IQR}(X|\sigma)$ respectively), and models with EVT-based classification boundaries (thick and think purple curves for σ estimated using $L_{sum}(X|\sigma)$ and $L_{IQR}(X|\sigma)$). For reference purposes the ROC curves for IncSVDD are also shown.

Overall, the results achieved for the four combinations of σ selection method and classification boundary appear to be very similar, with no clear trend linking relative performance levels (ie. relative to the configuration used in the preceding sections - denoted by the thick blue curve) with any given alternative configuration. However, performance did appear to differ significantly on two of the datasets. For the Spiral dataset, models where σ was estimated using $L_{IQR}(X|\sigma)$ (shown in Section 3.6.2 to yield smaller values than $L_{sum}(X|\sigma)$) produced better performance than those using $L_{sum}(X|\sigma)$, regardless of the type of classification boundary. Noting that the Spiral dataset requires the model boundary to occupy a very precisely defined region surrounded by outliers, it is clear that the resulting classification performance is sensitive to the slightest over-generalisation, thus favouring a smaller value of σ.

Conversely, for the Shuttle dataset, the models where σ is estimated using $L_{sum}(X|\sigma)$, provide better performance (in conjunction with a density-based boundary) than models using $L_{IQR}(X|\sigma)$ for either boundary type; in this case the bias introduced by a larger value of σ may be beneficial. However this classification performance is reduced when an EVT-based classification boundary is used; one potential reason for this is that - unlike density values, which may reflect overlapping low-density contributions from several components - the EVT probability values are calculated with respect to a single component, providing reduced capacity for generalisation (and thus yielding lower values of TPR for a given FPR value).

However, it is also important to note that generalisation due to the overlapping "tails" of components (ie. when a given location is covered by the low probability re-

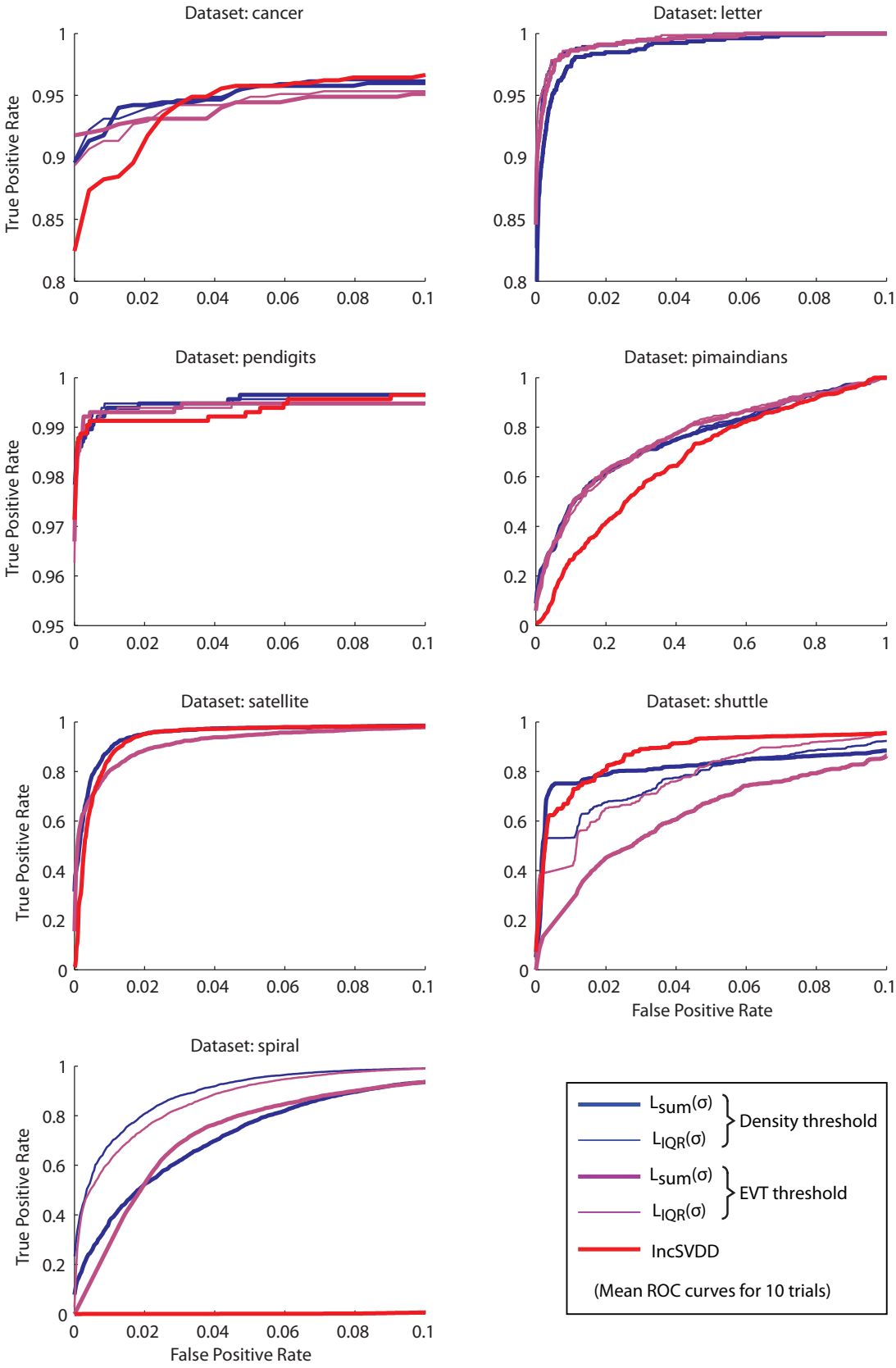


Figure 3.11: ROC curves comparing the impact of different method for estimating σ (see Sections 3.5.2 and 3.6.2) and different methods of defining classification boundary (see Section 3.5.3) on resulting classification performance. (See text for description).

gions of several Gaussian components whose densities add together to yield a density above the normal classification threshold) may not always be desirable: in such cases, the component-wise EVT threshold may be the best choice. This point is illustrated by the performance on the Letter dataset, where the EVT-based boundaries appear to yield a marginal improvement over the density-based boundaries, regardless of the method used to select σ. For this dataset, which contains parametrised images of printed letters, the summation of weak resemblances between a new instance (eg. the letter 'R') and a number of model components (all representing the letter 'A') appears to be undesirable.

3.6.6 How does the kernel function affect classification performance?

The preceding section (see Figure 3.11) showed that the value of σ (as determined by the two model selection criteria $L_{sum}(X|\sigma)$ / $L_{IQR}(X|\sigma)$ which favour slightly higher / lower σ respectively) had a noticeable effect on classification performance for certain datasets . As discussed in Section 3.5.2, the value of the kernel parameter σ affects the resulting mixture model in two different ways: 1) By acting as a bias term which smooths the resulting density estimate and 2) by altering relative cost of different merging operations as each new kernel is added to the model. To determine how these factors account for the observed difference in classification performance for different σ, a further experiment was conducted where the kernel bias term $I^d.\sigma$ was subtracted (at the end of training) from each covariance matrix in the model[1].

Figure 3.12 shows ROC curves documenting classification performance (obtained using a density based threshold) for models generated with two different values of σ (determined by using either $L_{sum}(X|\sigma)$ or $L_{IQR}(X|\sigma)$), both before and after the subtraction of the kernel bias term . In most cases, the impact of the kernel covariance on the underlying model structure (through altering the relative cost of merging pairs of existing components when each new kernel is added) appears to be minimal. For the Spiral and Shuttle datasets, where there is a relatively large difference between models with different σ values before the subtraction of the kernel bias term, there is almost no difference afterwards. For the Letter dataset, however, there is a large difference in classification performance for different σ values *after* the kernel bias term has been subtracted, implying that in this particular case the underlying model structure is significantly affected by different values of σ.

Nonetheless, removing the kernel bias term has a large effect on classification performance in almost all cases. In several cases (ie. the Cancer, Letter, Pendigits and

[1]To avoid the possibility of zero elements in the diagonal of the resulting covariance matrices another, much smaller, bias term $I^d.10^{-6}$ was then added to each matrix.

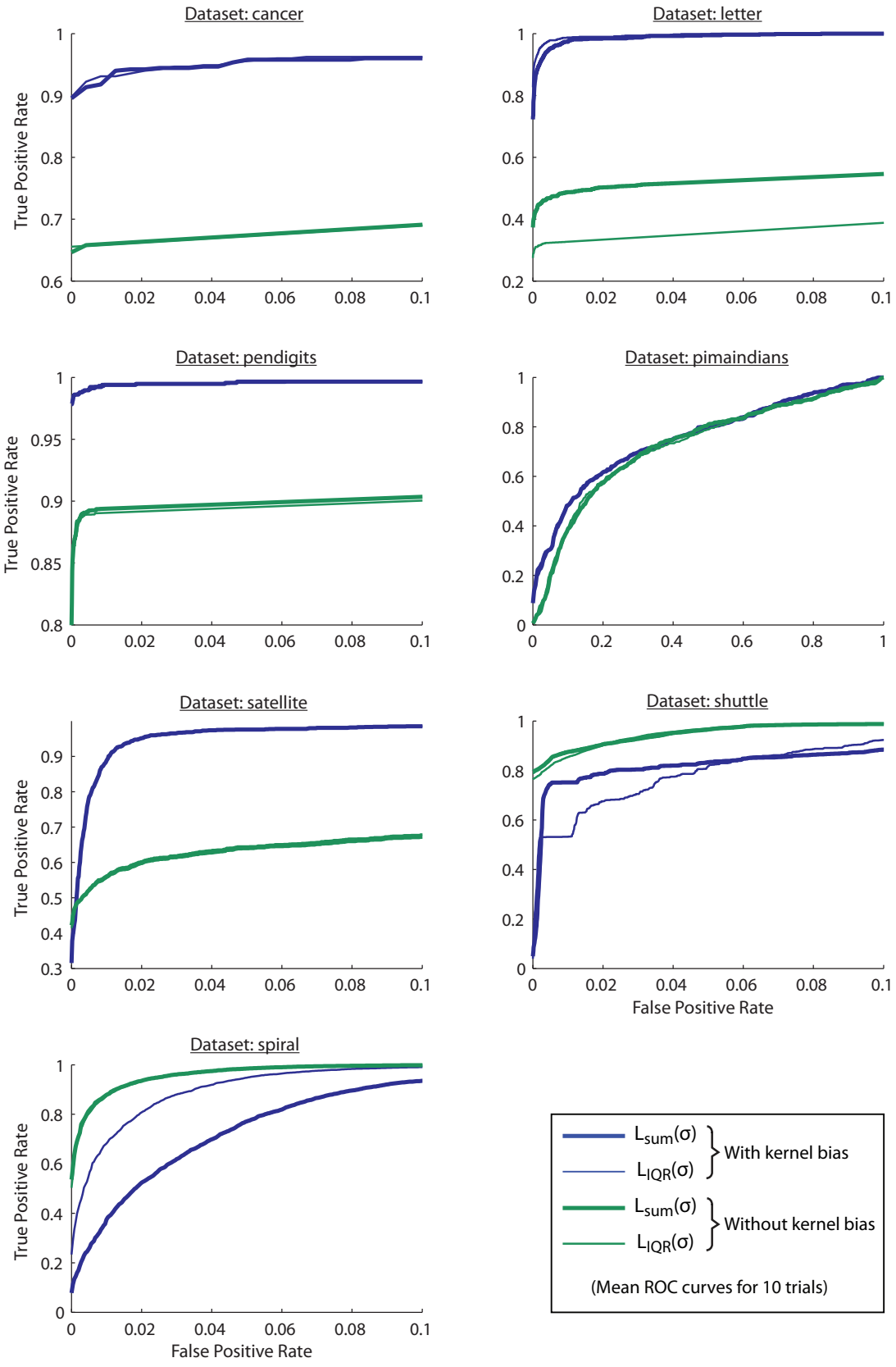


Figure 3.12: ROC curves illustrating the impact of subtracting the kernel bias term (see Section 3.5.2) from the mixture models obtained at the end of training given two different values of σ (see Section 3.6.2). A density-based threshold is adopted in all cases. (See text for description).

Satellite datasets) subtracting the kernel bias term substantially reduces classification performance. Conversely, for the Spiral and Shuttle datasets, removing the kernel bias yields a substantial improvement in classification performance. The observed improvements/reductions in classification performance indicate that the impact of the kernel covariance matrix on classification performance can largely be accounted for in terms of its role as a bias/smoothing term. Noting that the only datasets where removing the kernel bias did not reduce performance (the Spiral and Shuttle datasets) were those with the largest number of training examples (2500), it would be reasonable to conclude that the kernel bias term plays an important role in compensating for the lower ratio of training examples to mixture components encountered with the other datasets. In this light, it would be interesting to conduct further experiments with larger datasets to determine whether removing the kernel bias term would provide improved performance in all cases, once a sufficient number of examples have been observed.

3.7 Discussion

The algorithm proposed in this chapter provides a way to incrementally build a Gaussian mixture model to represent a single class of data: experimental results on both synthetic and real-world datasets indicate that the proposed algorithm is capable of delivering a useful level of classification performance, outperforming the Incremental SVDD algorithm [146] in the majority of cases when the latter is optimised using the consistency-based criterion proposed in [143]. It is worth noting that the parameter selection procedure used for the IncSVDD algorithm has the benefit of operating on the basis of the whole set of training data in an off-line optimisation step, while the proposed algorithm performs parameter selection "on-the-fly" on the basis of the first 100 (as configured here) points observed, making it significantly more useful in a practical situation.

However, it is important to note that the parameter selection procedures implicit in both algorithms operate on the sole basis of heuristics derived from the (one class) training data, rather than classification performance on a (two class) validation set (which is by definition unavailable in scenarios where one-class learning is required). The preceding results are therefore as much a comparison of the ability of the (training data based) parameter selection procedures to deliver reasonable parameters for the two learning algorithms as they are of the learning algorithms themselves. In this light, the results presented in this chapter do not rule out the possibility that IncSVDD may provide better classification performance than the proposed algorithm if given

the correct parameters; however, when considering each algorithm in conjunction with its best available model selection procedure, the proposed algorithm compares favourably with IncSVDD.

The intended purpose of the proposed algorithm is to provide a mechanism for incrementally learning models of typical motion trajectories. The next chapter of this thesis addresses the issue of representing motion trajectories in a consistent parametric form, so that the proposed algorithm can be applied. Finally, Chapter 4 explores the application of the algorithm proposed in this chapter to suitably parametrised trajectory data, as part of a semi-supervised learning framework.

Chapter 4

Parametric Trajectory Representation for Behaviour Classification

4.1 Introduction

In order to apply the learning algorithm described in Chapter 3 to trajectory data, it is necessary to encode each trajectory with a vector of fixed length: this poses a problem as there is no restriction on the length of a given trajectory. A solution to this problem is to represent each trajectory with a low-dimensional parametric approximation. However, there are many different possible parametric representations that could be applied to trajectories, and no clear indication of the most appropriate choice for the modelling/classification approach proposed in this thesis. This chapter attempts to address this problem through experimental comparison.

Firstly, several different ways to encode a trajectory with a pre-defined number of parameters are described in Section 4.2. A number of techniques for quantifying class separability are then reviewed in Section 4.3. Finally Section 4.4 present quantifications of the trade-off between dimensionality and class-separability for each representation, using several different real trajectory datasets.

4.2 Parametric Trajectory Representations

This section reviews several different strategies for approximating a trajectory using a fixed set of parameters. As illustrated in Figure 4.1, a motion trajectory can be considered in terms of two independent signals: X and Y positions vs time. In this light, a trajectory representation can be obtained by fitting a separate parametric curve to

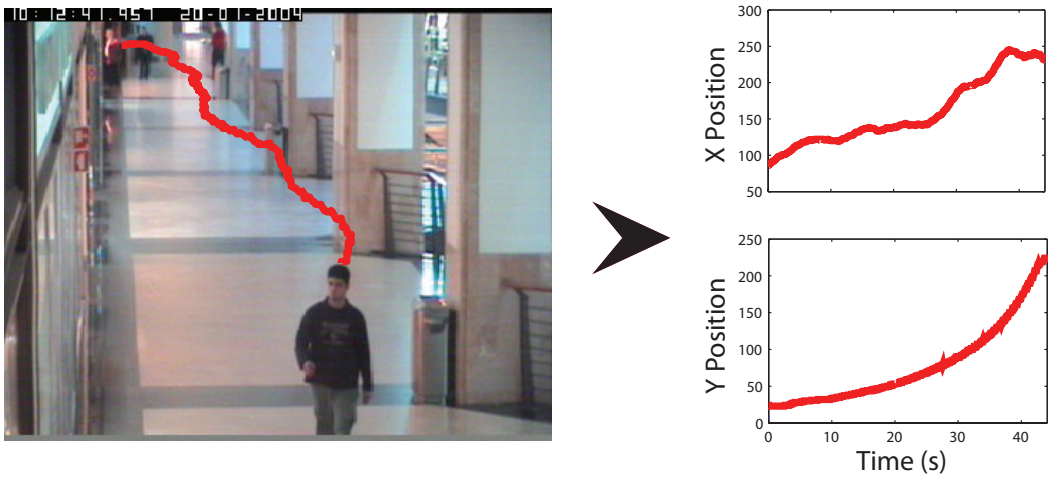


Figure 4.1: Representation of trajectory data as separate (X and Y vs time) signals.

both signals and concatenating the resulting parameters into a single vector [120, 90], which means that many solutions to the problem of 1D curve/function approximation can be directly applied to the task of trajectory representation.

Each of the representational strategies discussed in this chapter allows a trajectory to be described with an arbitrary number of parameters, which means that varying levels of accuracy can be obtained for a given strategy. For example, Figure 4.2 illustrates the how the fidelity of a cubic spline trajectory representation (see Section 4.2.1) changes as a function of the number of control points. Clearly, with a large enough number of parameters it is possible to model a given trajectory almost exactly. In the context of behaviour of classification, the quality of a representation corresponds to its ability to separate a set of distinct classes, which may not always correspond directly to approximation error. In this light, an interesting question is how the quality of the representations differs when only a small subset of the possible parameters is used to approximate each trajectory.

Each representational approach discussed provides a means to describe arbitrary-length coordinate vectors $\vec{X}$ and $\vec{Y}$, sampled at times $\vec{T} = \{t_1, \ldots t_N\}$, with fixed-length parameter vectors $\vec{C^X}$ and $\vec{C^Y}$, which can then be concatenated as a single vector describing the trajectory, together with the total time t_N:

$$\left\{\vec{X}, \vec{Y}, \vec{T}\right\} \rightarrow \left[C_1^X, \ldots, C_M^X, C_1^Y, \ldots, C_M^Y, t_N\right] \tag{4.1}$$

In each case the parameters $\vec{C^X}, \vec{C^Y}$ extracted for a given coordinate sequence define a curve with respect to an underlying curve parameter, over a fixed interval. As suggested by Figure 4.1 a natural choice for this parameter is *time*, another possibility - relevant to trajectory representation - is *arc-length* (ie. the total distance traversed at a

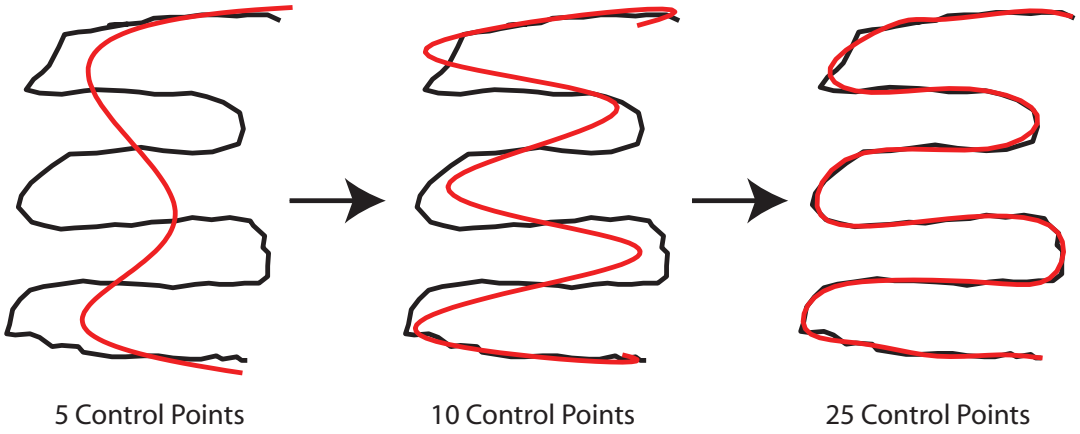


Figure 4.2: Cubic spline trajectory approximation with different numbers of control points.

given point) [50]. We explore both possibilities - thus, in the remainder of this section, the parameter s_n accompanying the nth coordinate pair x_n, y_n could refer interchangeably to one of the following:

1. Proportion of total time: $s_n = \frac{t_n}{t_N}$
2. Proportion of total arc-length: $s_n = \frac{\sum_{i=2}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}{\sum_{i=2}^{N} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}$

The use of a normalised curve parameter $s \in [0,1]$ is important as it means that the spatial locations occupied during a trajectory can be inferred directly from $\{\vec{C^X}, \vec{C^Y}\}$ without knowledge of the total distance traversed/time taken; moreover, it is a necessary consequence of certain representations (eg. those using Chebyshev or Wavelet basis functions) which are only defined in terms of a specific interval.

Ultimately, all of the following trajectory representation strategies define a curve approximation as sum of M basis functions $h_1(s), \ldots, h_M(s)$ weighted by a corresponding set of coefficients $c_1, \ldots c_M$, so that $X(s) = \sum_{m=1}^{M} h_m(s) \cdot c_m$. What distinguishes each method is the nature of the basis functions, and the manner in which the coefficients may be determined from a discrete coordinate sequence. In each case we describe how to determine a set of coefficients $\vec{C^X} = [c_{1,\ldots} c_M]$ approximating a coordinate sequence $\vec{X} = [x_{1,\ldots} x_N]$, noting that exactly the same procedure can be applied to $\vec{Y}$.

4.2.1 Least-Squares B-Spline Approximation

It is possible to describe a wide range of non-linear curves using a linearly weighted sum of a set of non-linear basis functions. B-Spline functions [8, 53] are an important example of such basis functions and have been widely adopted, both for the creation of

nonlinear curves in computer graphics contexts [102, 124], and for the representation of such curves in computer vision algorithms [50, 13, 24].

A B-spline curve $X(s)$ can be defined by a set of p B-spline functions $B_{i,d}(s)$, each weighted by a corresponding control point C_i^X:

$$X(s) = \sum_{i=0}^{p-1} C_i^X B_{i,d}(s)$$

The parameter d indicates the "order" of the B-spline functions, which determines the smoothness of the resulting curve, giving it $d-2$ continuous derivatives at each knot point (see below). Cubic B-spline functions, where $d = 4$, are a popular choice (see eg. [50, 13, 24]) and are adopted here, ensuring smooth joins between curve segments.

Each B-spline function $B_{i,4}(s)$ is defined by a knot vector $\vec{\tau}$ and the following recursive formulae:

$$B_{i,1}(s) = \begin{cases} 1 & \text{if } \tau_i \leq s < \tau_{i+1} \\ 0 & \text{otherwise} \end{cases} \qquad B_{i,m}(s) = \frac{s-\tau_i}{\tau_{i+m-1}-\tau_i} B_{i,m-1} + \frac{\tau_{i+m}-s}{\tau_{i+m}-\tau_{i+1}} B_{i+1,m-1} \tag{4.2}$$

The knot vector $\vec{\tau}$ encodes a set of points in the underlying parameter space (eg. time) which determine the interval over which the basis functions are defined, and their distribution within that interval. For a set of p basis functions of order d this vector has length $p+d$. For the purpose of representing multiple curves with the same basis set, the first/last d knots are set to be equal to the start/end of the desired interval, and the intermediate $p-d$ knots are uniformly spaced.

An approximation for a set of coordinates $\vec{X} = [x_1, \ldots x_N]^T$ with parameter values $\vec{S} = [s_1, \ldots s_N]^T$ can be expressed in terms of a vector of p unknown control points $\vec{C^X}$, and an $N \times p$ matrix Φ where $\Phi_{n,i} = B_{i,d+1}(s_n)$, so that:

$$\vec{X} \approx \Phi \vec{C^X} \tag{4.3}$$

The p control points which minimise the sum of squared errors between the original coordinates and their approximation can then be found using the Moore-Penrose pseudoinverse operator $\Phi^\dagger = (\Phi^T\Phi)^{-1}\Phi^T$ as follows [12]:

$$\vec{C^X} = \Phi^\dagger \vec{X} \tag{4.4}$$

Finally, it should be noted that this least-squares regression procedure is not specific to B-spline functions and can be used in conjunction with any other type of basis function, including the Fourier, Wavelet and Chebyshev basis functions discussed in the remainder of this section. In each case, however, there are other more efficient ways to fit curve models using these functions.

4.2.2 Discrete Fourier Transform

The Fourier Transform is a mathematical operation which maps a signal between the time and frequency domains. This allows a series of measurements sampled at regular intervals to be described as a weighted set of sinusoidal functions corresponding to different frequencies, where the weights for the constituent frequencies encode the information contained in the original signal.

For a discrete signal $\vec{X}$ of length N, there are $\frac{N}{2}$ distinct frequencies; the magnitude of the k-th frequency can be calculated as follows (where $\mathbf{i}$ is the imaginary unit $\sqrt{-1}$) [106]:

$$f_k(\vec{X}) = \frac{1}{N}\sum_{n=1}^{N} x_n \exp\left(-\frac{2\pi\mathbf{i}(k-1)(n-1)}{N}\right)$$

The resulting value of f_k is a complex number where the real part corresponds to a convolution of the original signal with a Cosine function (of wavelength $\frac{N}{k}$), and the imaginary component corresponds to a convolution with a Sine function. In a similar vein the signal can be reconstructed from the extracted frequencies $\vec{F} = \{f_1 \dots f_N\}$ as follows:

$$x_n(\vec{F}) = \sum_{k=1}^{N} f_k \exp\left(\frac{2\pi\mathbf{i}(k-1)(n-1)}{N}\right)$$

For the purpose of trajectory representation, an important property of this representation is that the first M values of f_k form a useful coarse-grained approximation of the original signal, by disregarding the $N - M$ highest spatial frequencies (Naftel et al. represent trajectories in this fashion in [91, 90].) Noting that the imaginary component of f_1 is always zero, a set of coordinates $\vec{X} = [x_1, \dots x_N]$ can thus be represented with a $2M - 1$ dimensional vector as follows (where $\Re(z)$ and $\Im(z)$ denote the magnitude of the real and imaginary components of z):

$$\vec{C}^X = \left[\Re(f_1(\vec{X})), \Re(f_2(\vec{X})), \Im(f_2(\vec{X})), \dots, \Re(f_M(\vec{X})), \Im(f_M(\vec{X}))\right]$$

Since trajectories are assumed to be regularly sampled in time, the Discrete Fourier Transform yields a spatiotemporal parametrisation by default. To achieve spatial parametrisation with this method, we linearly interpolate the original trajectory to create a new coordinate sequence regularly sampled in terms of arc-length.

4.2.3 Chebyshev Polynomials

Chebyshev polynomials, which are often used for function approximation [106], provide another useful set of basis functions that can be used to represent trajectories

[22, 90]. The nth degree Chebyshev polynomial function is defined as follows:

$$T_n(s) = \cos(n \arccos(s))$$

When parametrised by an appropriate set of coefficients $\{c_1, \ldots c_N\}$, a function $f(s)$ defined on the interval $[-1,1]$ can be modelled with a weighted sum of Chebyshev polynomials as follows [106]:

$$f(s) \approx \sum_{k=0}^{N-1} c_k T_k(s)$$

As with DFT coefficients, the first M coefficients can be used to provide a useful approximation of the original signal [106]. In order to calculate coefficients to approximate a function $X(s)$ with the first M Chebyshev polynomials, it is necessary to evaluate $X(s)$ and $T_0(s), \ldots, T_{M-1}(s)$ for values of s where $T_M(s) = 0$. There are M such values for $T_M(s)$, with the kth one given by:

$$s_{0,k} = \cos\left(\frac{\pi(k+\frac{1}{2})}{M}\right)$$

Clearly, the parameter values $\vec{S}$ associated with a discretely sampled coordinate sequence $\vec{X}$ do not necessarily correspond to $s_{0,0}, \ldots s_{0,M-1}$. Moreover, Chebyshev approximation operates on the interval $[-1,1]$ while the normalised curve parameters defined at the start of this section lie between 0 and 1. We address this issue by rescaling $\vec{S}$ to the interval $[-1,1]$ and using linear interpolation to find an an X value, $x_{0,k}$, for each $s_{0,k}$. Finally, each coefficient c_j can be calculated as follows [106]:

$$c_0 = \frac{1}{M} \sum_{k=0}^{M-1} x_{0,k} \cdot T_0(s_{0,k}) \quad c_{j>0} = \frac{2}{M} \sum_{k=0}^{M-1} x_{0,k} \cdot T_j(s_{0,k}) \tag{4.5}$$

4.2.4 Haar Wavelet Coefficients

Wavelets provide another useful class of basis functions that can be used to parametrise signals in a multiscale fashion. Unlike Fourier and Chebyshev functions, which contribute globally to the resulting signal, Wavelet basis functions are localised; in other words, they assign zero values to all but a specific subsection of the interval over which the signal is defined.

There are many different possible wavelet basis functions (see eg. [21]); however, for the purposes of this chapter we restrict our attention to Haar wavelets, as these have previously been applied to trajectory parametrisation by Sahouria et al. in [120], and are popular for time series indexing (see eg. [162, 25]). The Haar 'mother wavelet' function ψ and scaling function ϕ are defined on the interval $[0,1]$ as follows [21]:

$$\phi(s) = \begin{cases} 1 & \text{if } 0 \leq s < 1 \\ 0 & \text{otherwise} \end{cases} \qquad \psi(s) = \begin{cases} 1 & \text{if } 0 \leq s \leq \frac{1}{2} \\ -1 & \text{if } \frac{1}{2} < s \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Given a mother wavelet function ψ, a range of rescaled and shifted versions of the function can be derived, to define a set of local basis functions at multiple scales/resolutions. Given the above definition for ψ, the Haar wavelet function corresponding to the kth location at the jth resolution can then be defined as follows [158]:

$$\psi_{j,k}(s) = 2^{\frac{j}{2}} \psi(2^j s - k)$$

Figure 4.3d illustrates the resulting Haar wavelet functions at the first (ie. coarsest) three resolutions $j = 0 \ldots 2$. For each successive scale, the number of basis functions corresponding to different locations increases by a factor of 2, which means the number of basis functions used to approximate a given signal must always be an integer power of 2; similarly, the signal length must also be an integer power of 2. As with the preceding approaches, a signal can be expressed as a weighted sum of wavelet basis functions - the Jth resolution approximation to a signal $X(s)$ can be written in terms of a set of 2^J coefficients $\{\alpha, \beta_{0,0}, \ldots \beta_{J-1,2^J-1}\}$ as follows [158]:

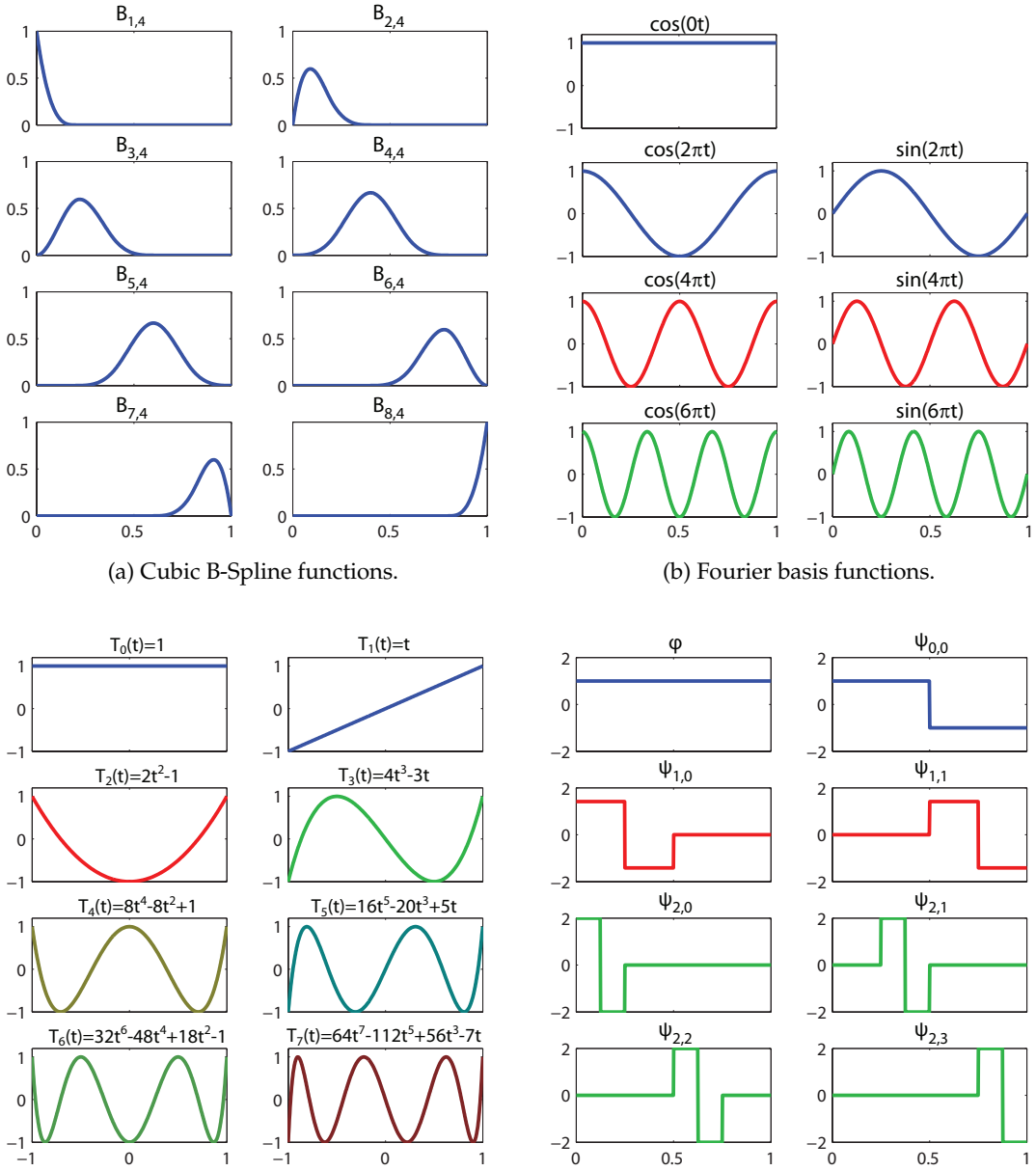
$$X(s) \approx \alpha \phi(s) + \sum_{j=0}^{J-1} \sum_{k=0}^{2^j-1} \beta_{j,k} \psi_{j,k}(s)$$

Given a signal $\vec{X} = [x_1, \ldots x_N]$, where N is an integer power of 2, the Haar wavelet coefficients can be calculated as follows [158] (it should be noted that this calculation can be performed in a more efficient manner using repeated applications of a transformation matrix - see [137, 21] for details):

$$\alpha = \frac{1}{N} \sum_{i=1}^{n} \phi(s_i) \cdot x_i \qquad \beta_{j,k} = \frac{1}{N} \sum_{i=1}^{n} \psi_{j,k}(s_i) \cdot x_i$$

The requirement that signal length should be an integer power of 2 poses a problem for trajectory parametrisation, which involves sequences of arbitrary length. This issue is addressed in [120] by resampling trajectories to an appropriate length; we adopt this strategy here, resampling (using linear interpolation) each coordinate sequence to a uniform length of 512 (ie. 2^9) elements. A vector of 2^J coefficients (where $J \geq 1$) can then be constructed to represent the resampled signal:

$$\vec{C}^X = [\alpha, \beta_{0,0}, \ldots \beta_{J-1,2^J-1}]$$



(a) Cubic B-Spline functions.

(b) Fourier basis functions.

(c) Chebyshev polynomials.

(d) Haar wavelet functions.

Figure 4.3: This figure shows sets of basis functions corresponding to the four representational strategies discussed in Section 4.2. In each case - except for the Fourier basis functions which must be used in odd numbered sets - a set of 8 basis functions is shown, corresponding to a 16+1 dimensional trajectory representation (8 coefficients each for X and Y coordinate sequences, and 1 for time). For the Fourier, Chebyshev and Haar wavelet functions, the colouration of the curves indicates the different resolutions represented by subsets of the 8 (or 7) basis functions. See text for further details.

4.3 Measures of Separability

Given a set of trajectories corresponding to several different classes of motion and a potential parametric representation, a reasonable question to ask is: to what extent do the motion patterns from each class form distinct clusters in the space defined by the chosen representation? Answering this question requires the ability to quantify the extent to which the regions of feature-space occupied by each class overlap, a quality known as *separability*. This section highlights three techniques that may be used to quantify class separability.

A key feature of the following methods is that they can be applied deterministically to a given labeled dataset, without requiring the pre-specification of any parameters/assumptions about the nature of the data.

4.3.1 Within-vs-between class scatter

If the different classes constituting a dataset are well separated in a given feature space, it is likely that the dispersion of the class means will be large, while the dispersion of the points within each class will be small. It is possible to quantify this intuitive quality in the following manner [45, 160].

The dispersion of the class means is captured by the *between-class covariance matrix* S_B, which measures the covariance of the class means μ_k with respect to μ_{global}, the mean of the whole dataset, with the contribution of each class mean weighted by the proportion of the dataset it accounts for $\frac{N_k}{N_{global}}$ as follows:

$$S_B = \sum_{\forall k} \frac{N_k}{N_{global}} (\mu_k - \mu_{global})(\mu_k - \mu_{global})^T$$

The overall measure of within-class dispersion is captured by the the *within-class-covariance matrix* S_W, which is calculated by adding the covariance matrices Σ_k corresponding to each individual class, again, weighted by the portion of the dataset accounted for:

$$S_W = \sum_{\forall k} \frac{N_k}{N_{global}} \Sigma_k$$

The matrices S_B and S_W can be combined in a variety of different ways (see [45, 160]) to yield a single value which increases proportionally to the separability of the classes. Of these measures, one of the most widely adopted [45, 39, 160, 12] is the following:

$$J_S = Tr\{S_W^{-1} S_B\} \tag{4.6}$$

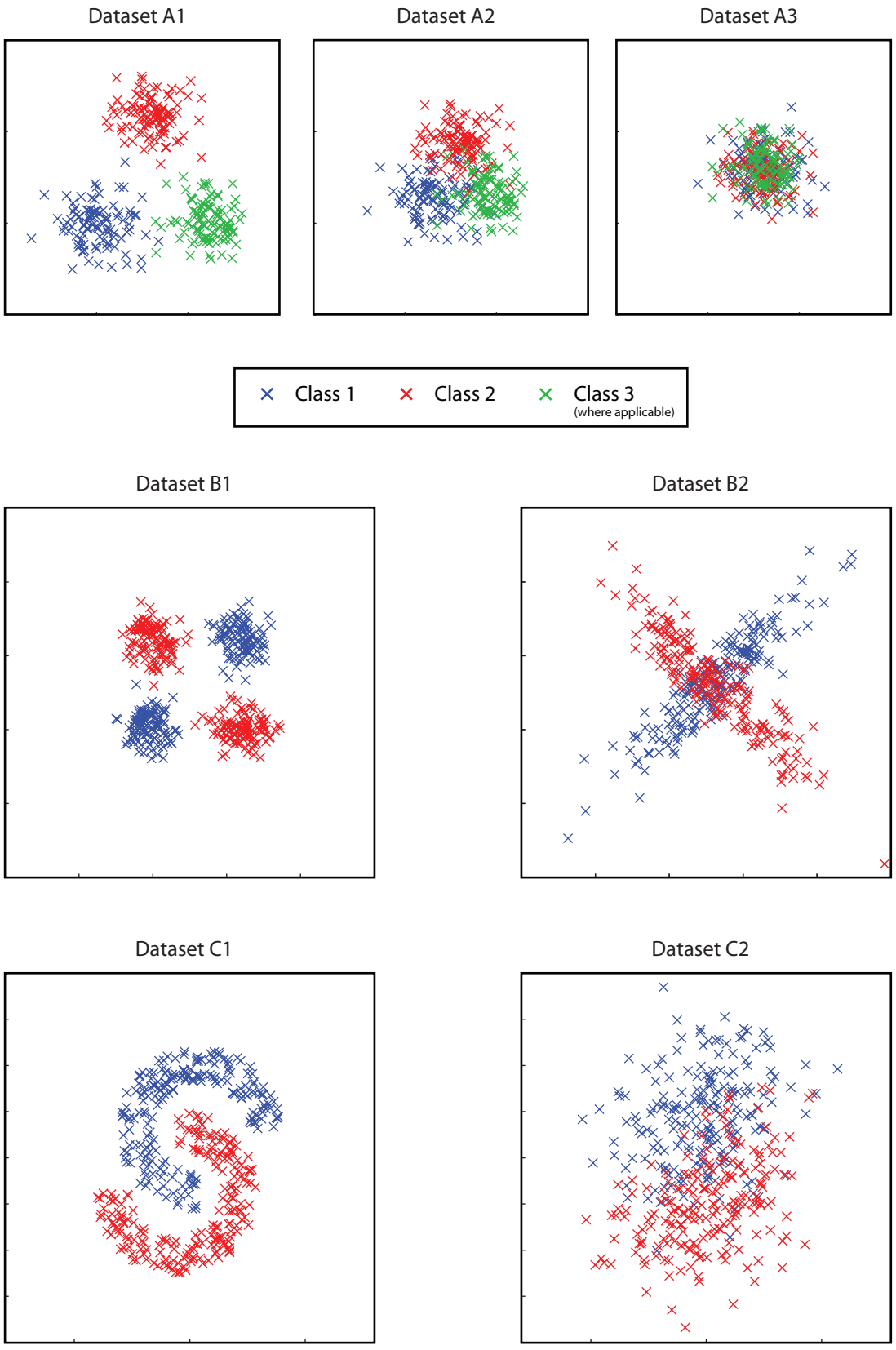
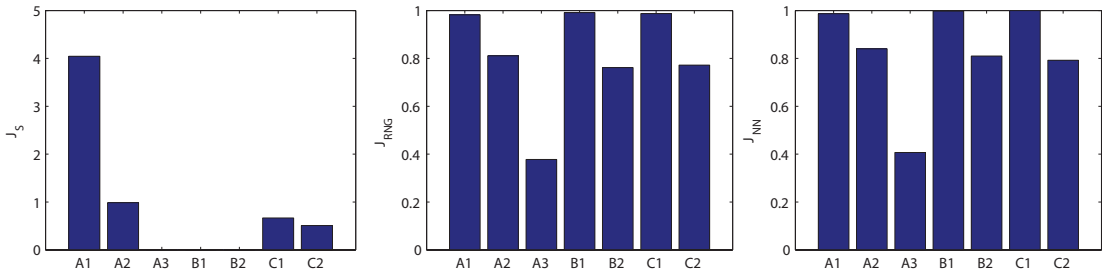


Figure 4.4: Synthetic 2D datasets used to examine the behaviour of the different separability measures described in Section 4.3 (see text for discussion).



(a) (b) (c)

Figure 4.5: Separability measures applied to synthetic datasets shown in Figure 4.4. (a) Within-vs-between class scatter J_S (Equation 4.6); (b) Graph-based edge-weight statistic J_{RNG} (Equation 4.9); (c) Nearest-neighbour classification accuracy J_{NN} (Equation 4.10).

To illustrate the behaviour of this separability measure, it is applied to a series synthetic 2D datasets as shown in Figure 4.4. Datasets A1,2,3 contain three normally distributed classes with increasing degrees of overlap, ranging from perfect separation (A1) to total overlap (A3). The corresponding values of Equation 4.6 are plotted as a bar chart in Figure 4.5a: it can be seen that this separability measure correctly assigns a large value to dataset A1, a lower value to A2, and a very small value to A3.

A further set of datasets B1,2 and C1,2 provide a more difficult scenario. In B1, the classes in the dataset are well-separated, but each class is bimodally distributed; as a comparison, B2 contains two overlapping unimodal class distributions (whose means and covariances are virtually identical to those in B1). The results in Figure 4.5a, show that B1 and B2 are both assigned a very low separability score. Similarly in C1, classes are uniformly distributed in two non-overlapping "horse shoe" shaped regions, while in C2 the two classes come from overlapping unimodal distributions: again the two datasets are assigned very similar separability values.

It is clear that the principal failing of this separability measure is its underlying assumption that each class is well described by a unimodal distribution. This problem is acknowledged in [45], where it is suggested that the dataset could be clustered to yield a larger set of unimodal sub-classes, to which Equation 4.6 could then be applied. However, by introducing the problem of clustering, this solution ceases to provide a measure that can be deterministically calculated from the dataset.

4.3.2 Graph-based edge-weight statistic

Noting the cases in which the preceding separability estimator fails, it would be desirable to be able to quantify separability without making assumptions about the parametric form of the class distributions. In this light, an attractive non-parametric tech-

nique has been proposed by Zighed et al. in [181], described as follows.

To measure separability, a Relative-Neighbourhood Graph (see [150]) is constructed from the dataset, where each node represents an individual data point, and edges join only those pairs of nodes that have no mutually closer neighbour. Formally, nodes x_i and x_j are joined if the following condition is met (where $d(x_i, x_j)$ denotes the Euclidean distance between the two points):

$$d(x_i, x_j) \leq \max\left[d(x_i, x_k), d(x_j, x_k)\right]_{\forall k \notin \{i,j\}}$$

Furthermore, each edge is weighted according to the similarity of the data points it joins, where similarity is defined as the following inverse function of Euclidean distance (which can range between 0 and 1) [181]:

$$S_{ij} = \frac{1}{1 + d(x_i, x_j)}$$

Thus a matrix W describing the resulting graph can be constructed from the dataset, where each element w_{ij} is defined as follows:

$$w_{ij} = \begin{cases} \frac{1}{1+d(x_i,x_j)} & \text{if} \quad d(x_i, x_j)_{i \neq j} \leq \max\left[d(x_i, x_k), d(x_j, x_k)\right]_{\forall k \notin \{i,j\}} \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

It is then possible to define quantities analogous to within and between class scatter in terms of this matrix, through summation of the weights of edges joining members of the same class/those joining members of different classes as follows (where $I(\blacksquare)$ denotes the indicator function, and L_n denotes the label of the nth data point):

$$E_{within} = \sum_{i=1}^{N-1} \left(\sum_{j=i+1}^{N} I(L_i = L_j) \cdot w_{ij} \right) \qquad E_{between} = \sum_{i=1}^{N-1} \left(\sum_{j=i+1}^{N} I(L_i \neq L_j) \cdot w_{ij} \right) \tag{4.8}$$

Noting that edges are only present for pairs of data points that are (relatively [150]) proximal, it is possible to measure separability in terms of E_{within} and $E_{between}$. In particular, for a dataset whose classes are well-separated, the weights of edges joining members of the same class E_{within} will constitute the majority of the total sum of edge weights, $E_{within} + E_{between}$. This leads to the following measure, ranging between 0 and 1, which increases proportionally to class separability:

$$J_{RNG} = \frac{E_{within}}{E_{within} + E_{between}} \tag{4.9}$$

Again, we examine the behaviour of this measure on the synthetic datasets shown in Figure 4.4. As for the preceding method, the results shown in Figure 4.5b correctly

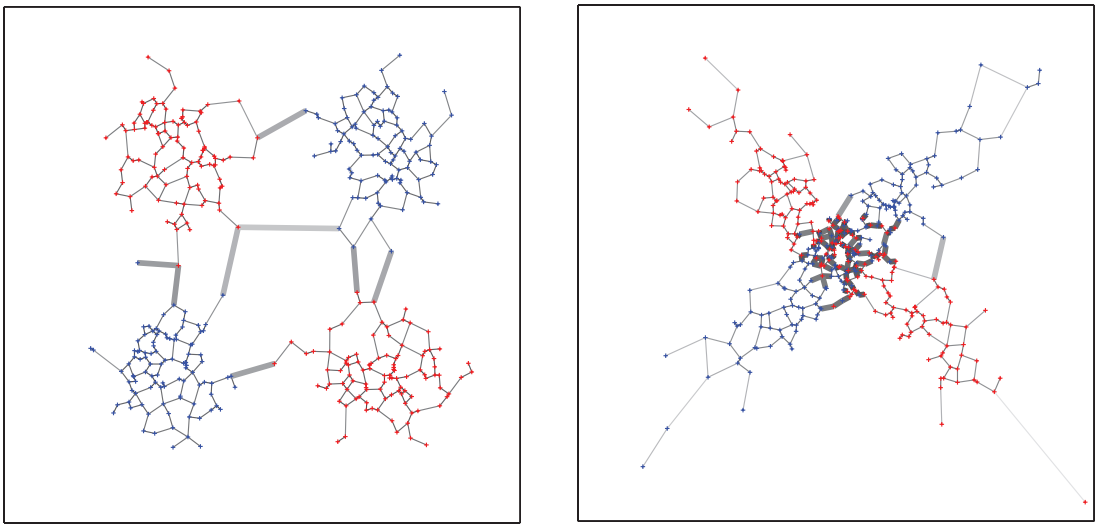


Figure 4.6: Relative Neighbourhood Graphs visualised for datasets B1 (left) and B2 (right). Thick lines indicate between-class edges, while thin lines indicate within-class edges. The darkness of each line indicates the edge weight (lighter shades indicate lower weights).

indicate the relative separability of datasets A1-A3. However, unlike the preceding method, datasets B1 and C1 are assigned a high separability value, similar to that of A1, while lower values are correctly assigned to datasets B2 and C2. This method therefore represents a useful substitute for the preceding method, when assessing the relative separability of datasets whose classes are not guaranteed to be unimodally distributed.

4.3.3 Nearest-neighbour classification accuracy

The final measure of class separability considered here is the average leave-one-out nearest-neighbour classification accuracy for the dataset. This classification method assigns to each data point the label of its nearest neighbour (as determined by Euclidean distance [53, 160, 12]); the better the separation of the different classes in dataset, the more likely it is that 1-nearest neighbour classification will be accurate.

Given a set of class labels, and a function $L(x_i)$ providing the label of the each data point x_i, a leave-one-out function predicting the class of each data point given the remainder of the dataset can be defined as follows (where $d(x_i, x_j)$ denotes Euclidean distance):

$$P(x_i) = L(\arg \min_{\{x_j | i \neq j\}} (d(x_i, x_j)))$$

The mean classification accuracy could thus be calculated as follows (where $I(\blacksquare)$ denotes the indicator function, and L_n and P_n denote the true and predicted labels for

the nth data point):

$$A = \frac{1}{N}\sum_{i=1}^{N} I\left(L(x_i) = P(x_i)\right)$$

Given datasets where class sizes are unequal, it is useful to modify this criterion so that it does not depend on the relative class sizes. For example in a two class dataset, where 80% of the examples come from a single class, an 80% classification accuracy could be misleading. To address this issue, the above measure is calculated separately for each class C_k so that:

$$A_k = \frac{1}{N_k}\sum_{\forall x_i \in C_k} I\left(L(x_i) = P(x_i)\right)$$

Finally this measure is averaged over the K classes present in the dataset to provide the following estimate of classification performance:

$$J_{NN} = \frac{1}{K}\sum_{k=1}^{K} A_k \tag{4.10}$$

Figure 4.5c shows the leave-one-out classification performance for the datasets shown in Figure 4.4. The resulting classification performance closely resembles the separability values assigned by the preceding graph-based measure J_{RNG}, indicating the efficacy of this measure. In this vein, J_{NN} provides an intuitive way to confirm the veracity of the values assigned by the preceding separability measures.

4.3.4 Summary

In this section we have reviewed three different methods for assessing the separability of a set of classes in a given representational space. It is clear that the graph-based measure J_{RNG} [181] and the nearest-neighbour classification performance J_{NN} provide a better reflection of class separability than the within-vs-between class scatter ratio J_S, if the distribution of data within individual classes is not unimodal, while behaving similarly if the classes are unimodal. Given the unknown nature of the class distributions in the trajectory datasets analysed in the remainder of this chapter, we will use J_{RNG} and J_{NN} as the principal means of measuring separability, while referring to J_S as a supplementary measure of the relative dispersion of class means.

4.4 Experiments

4.4.1 Trajectory datasets

This section describes a number of different trajectory datasets which are used in the following section to examine the behaviour of the representations described in Section 4.2, using the separability measures described in Section 4.3. Each dataset is chosen on the basis of containing multiple, distinct, classes of motion pattern.

CAVIAR "Shopping Mall" Dataset The CAVIAR dataset [42] provides a series of different video sequences, together with corresponding set hand-labeled ground truth data. We use a set of trajectories from this dataset corresponding to a side-on view of a corridor in a Portuguese shopping centre, as illustrated in Figure 4.7. This trajectory data has been used to assess representation/clustering methods in [90], where it was hand-labeled according to a set of six different motion classes - corresponding to motion along the corridor (two possible directions), and motion to/from the shop (four possible directions). We have replicated this labeling, as shown in Figure 4.7, yielding a set of 111 trajectories, with at least 10 examples for each of the 6 classes.

NGSIM "Peachtree Street" Traffic Dataset The NGSIM archive [23], part of a United States Department of Transportation initiative, provides a variety of real vehicle trajectory datasets which are intended to enable accurate parametrisation of traffic simulations. We use the "Peachtree Street" dataset, which contains over 2000 examples of vehicle trajectories over a road segment with a wide range of entry/exit points. For the purposes of this chapter we disregard all trajectories lasting fewer than 2 seconds. We label the trajectories according to their entry/exit point combination, initially resulting in 124 different classes; however, we discard any class containing fewer than 10 examples, yielding a final set of 33 different classes, corresponding to a total of 1758 trajectories. This dataset is illustrated in Figure 4.8, together with an aerial photograph of the relevant area, overlaid with regions corresponding to a set of 8 cameras which were used to collect the data (see [23] for details). It should be noted that the trajectory data has been mapped to a local coordinate system, relative to the centre of the street.

Australian Sign Language Dataset The Australian Sign Language dataset, created by Kadous [68] and available from the UCI Machine Learning Repository [6], provides a large set of trajectory data corresponding to hand movements for different words expressed in Australian Sign Language. The dataset contains hand trajectories for 95 different words, with 27 examples of each word, giving a total of 2565 trajectories.

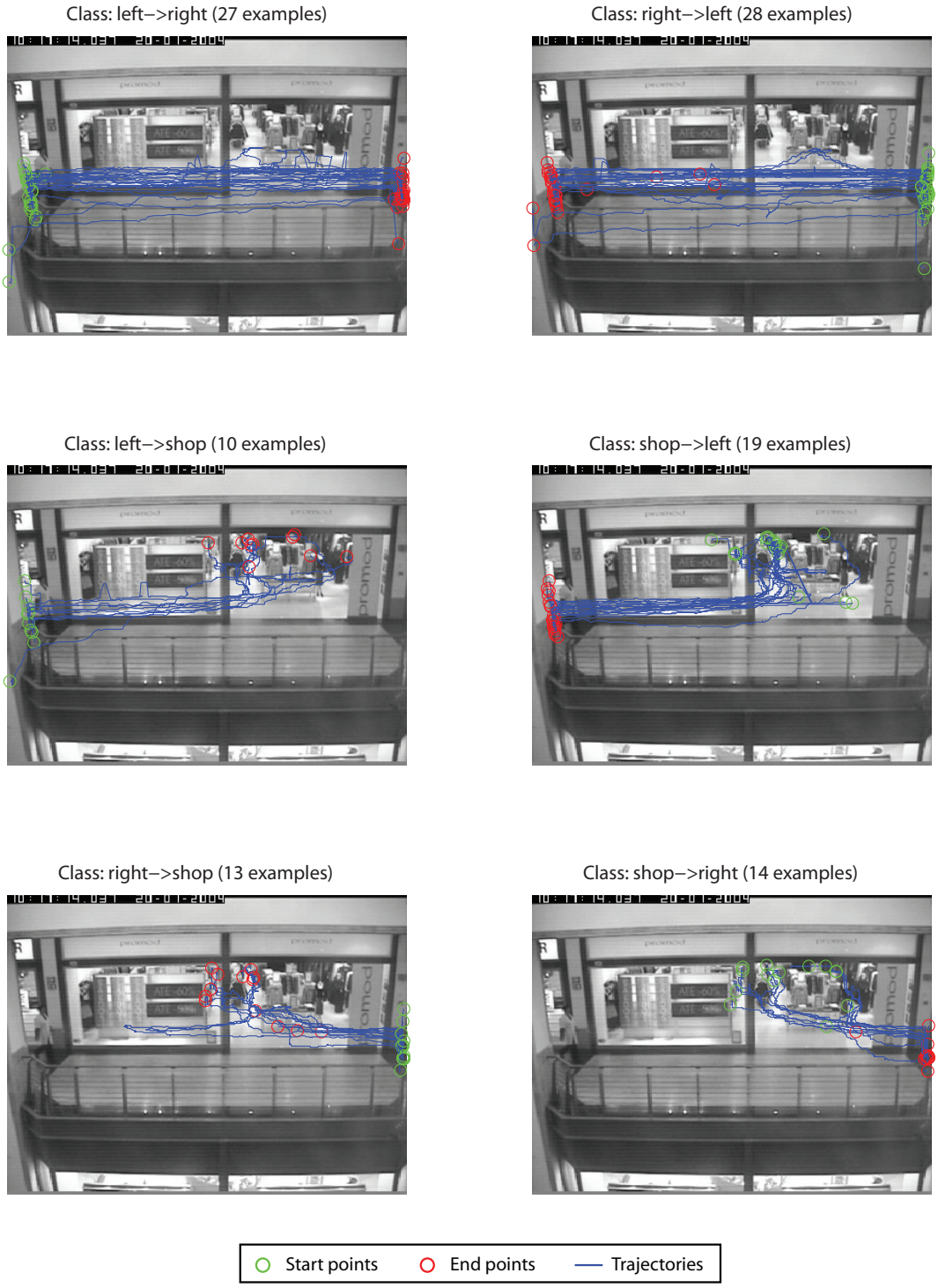


Figure 4.7: CAVIAR "Shopping Mall" scenario. Each section of this figure shows a set of trajectories corresponding to a manually assigned class. See text for further details.

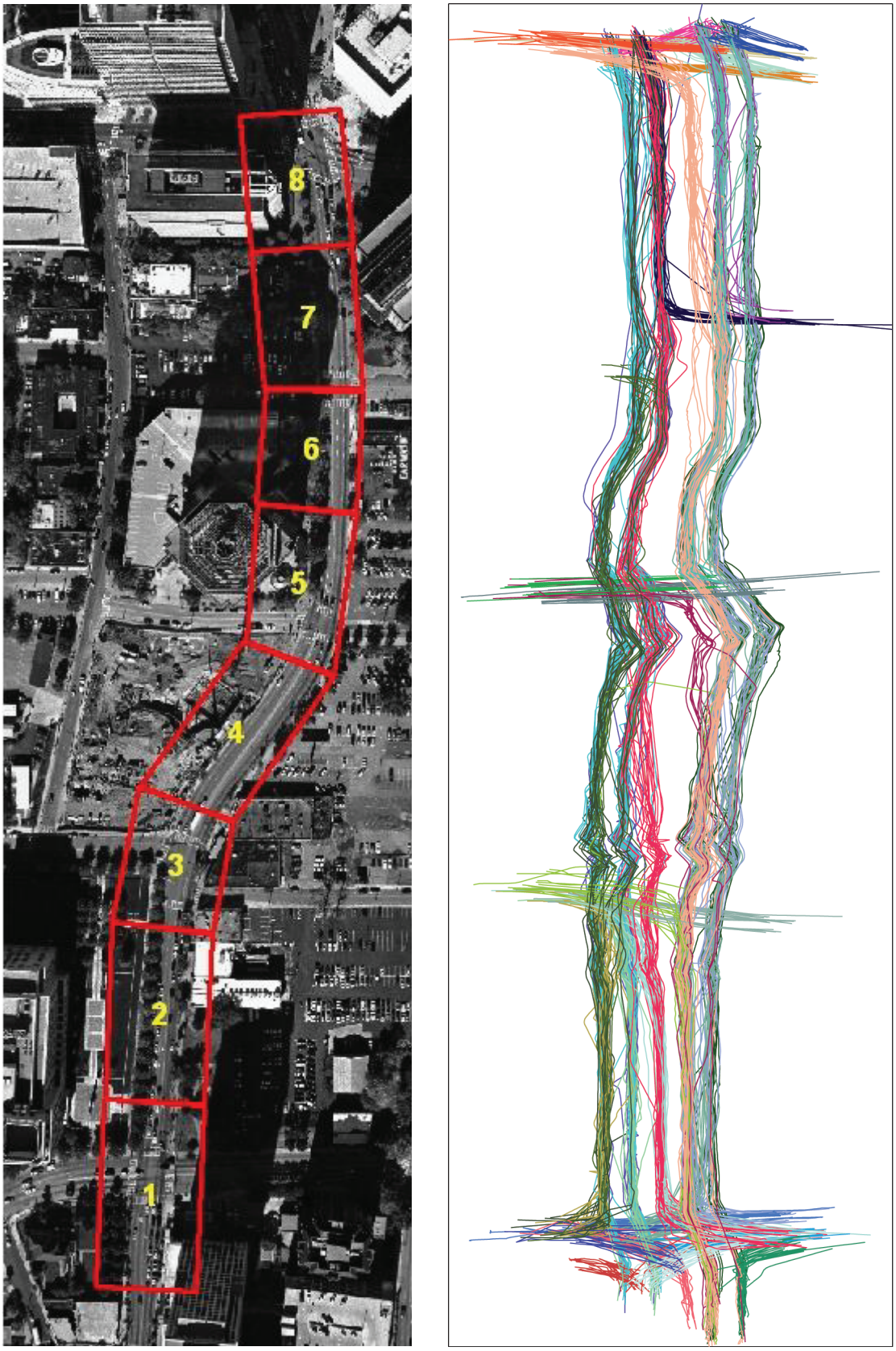


Figure 4.8: NGSIM "Peachtree Street" scenario with camera locations overlaid (left) and vehicle trajectories coloured according to route (right). See text for further details.

While the dataset contains the positions of both the left and right hands, together with information about their articulated configurations, a significant portion of information is captured by the position of the right hand. This assumption has been adopted in several approaches [90, 10], where word classification on the basis of right-hand trajectories has provided a challenging benchmark test. Figure 4.9 shows example trajectories for a random selection of 25 classes: it is clear that the variation between the trajectories corresponding to different words is sometimes extremely subtle.

Gun Point Dataset Another hand-trajectory dataset, created by Ratanamahatana and Keogh [108], contains two classes of trajectory corresponding to a surveillance scenario where the task is to detect the presence of a gun. One class of trajectories corresponds to people lifting their hand and pointing forwards, while a second class of trajectories corresponds to a subtly different motion pattern where a gun is removed from a holster before being pointed forwards; each class contains multiple examples of the same motion produced by two different people. An example from this scenario is illustrated in Figure 4.10a, and a corresponding set of trajectories is shown in Figure 4.10b. The dataset contains a total of 233 trajectories, with 121 examples of the gun pointing action, and 112 examples of the hand pointing action.

Pen Digits Dataset The *Pen-Based Recognition of Handwritten Digits* dataset [6], contains 10,992 pen trajectories corresponding to instances of the digits '0' to '9' written on a graphics tablet. We use 200 randomly chosen examples of each digit, leading to a total of 2000 trajectories with 10 distinct classes. Each trajectory is translated/uniformly rescaled to a common coordinate frame using the same procedure described for the processed/resampled version of this dataset, which is also available from [6]. Figure 4.11 shows an illustration of several example trajectories from each class.

Summary The datasets described in this section organise trajectories into classes in two different ways. As defined here, the classes in the CAVIAR and NGSIM datasets correspond entirely to route information. The challenge that these datasets pose for a potential trajectory representation is to separate instances corresponding to the different routes, while accommodating the large spatiotemporal variations possible with a given route. For example, within a given trajectory class from the NGSIM dataset, some trajectories may correspond to vehicles which stopped at certain traffic lights, while others may involve lane changes etc. Conversely, for the ASL, Gun-point and Pen-digits datasets, trajectories from different classes may start and end in similar locations: thus, in order to separate the different classes in these datasets, an accurate representation of the shape of each trajectory is required.

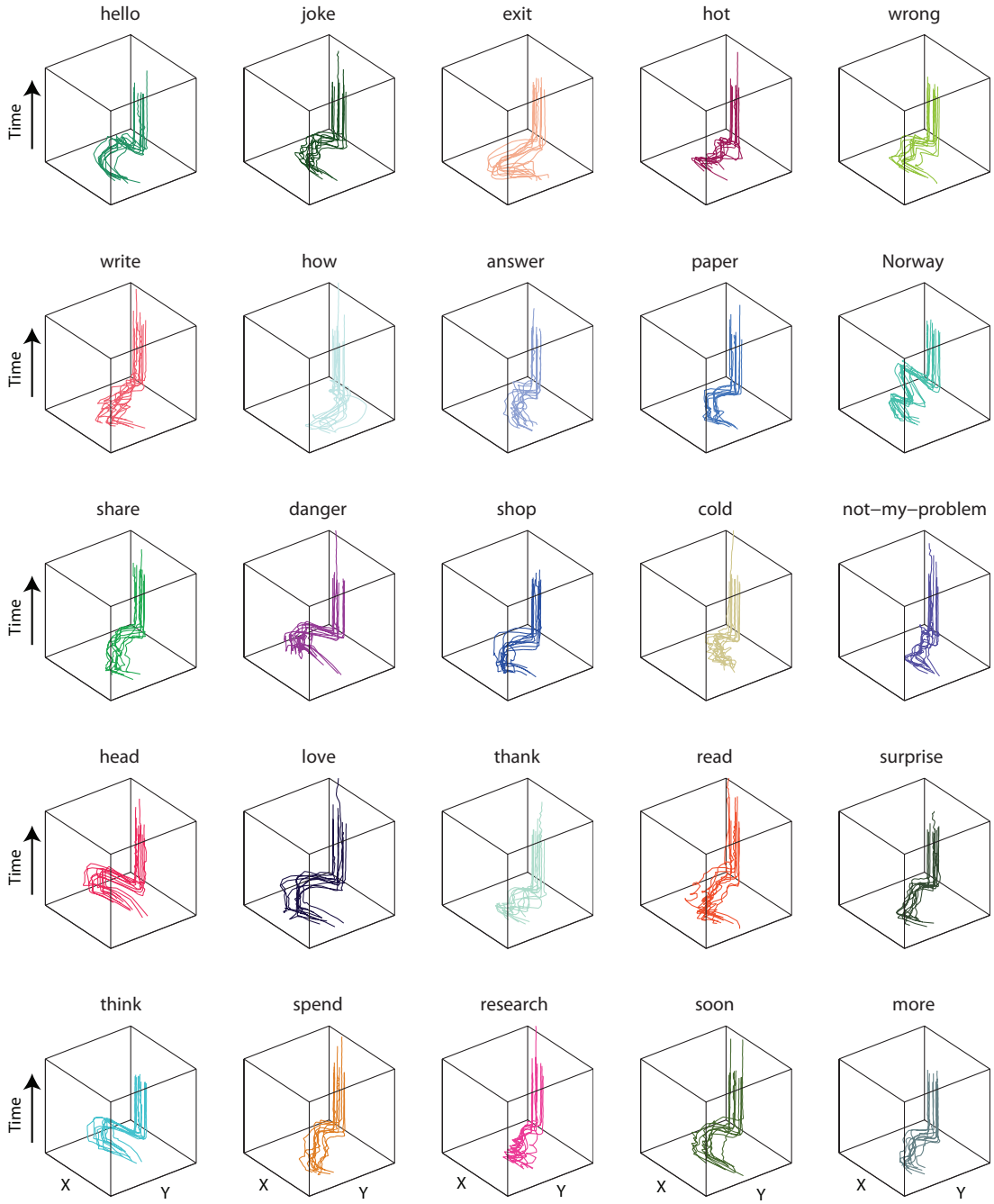
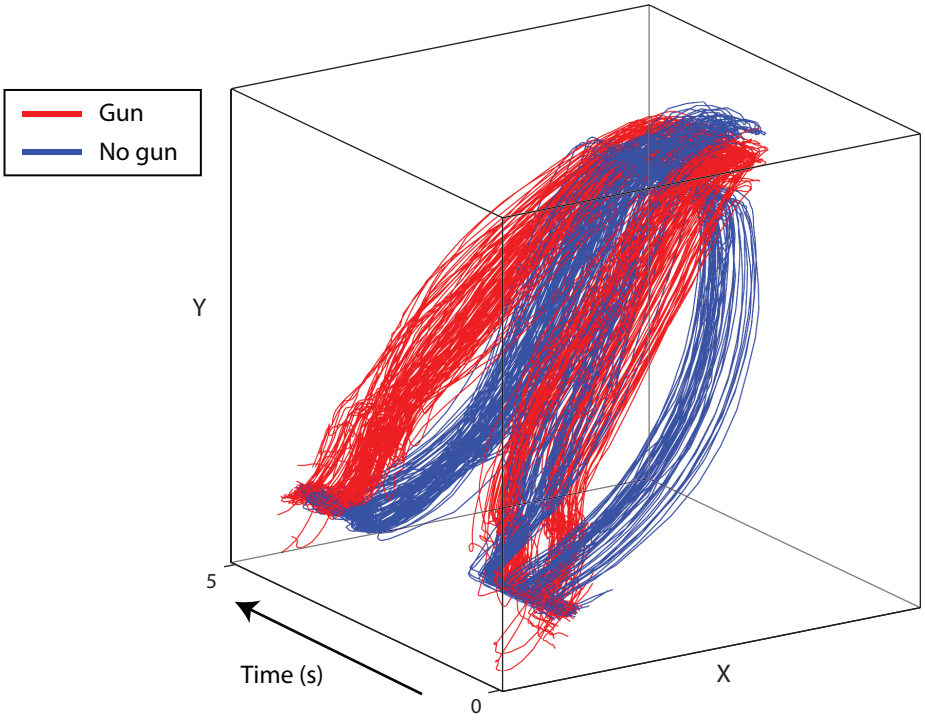


Figure 4.9: Australian Sign Language dataset: right-hand trajectories for a random selection of 25 classes (out of a possible 95), with 10 examples shown per class. See text for further details.

(a)



(b)

Figure 4.10: Gun Point Dataset. (a) Example video frames for the drawing/pointing of a gun (reproduced from [108]). (b) Resulting hand trajectories, which correspond to pointing with a gun (red) and without (blue). See text for further details.

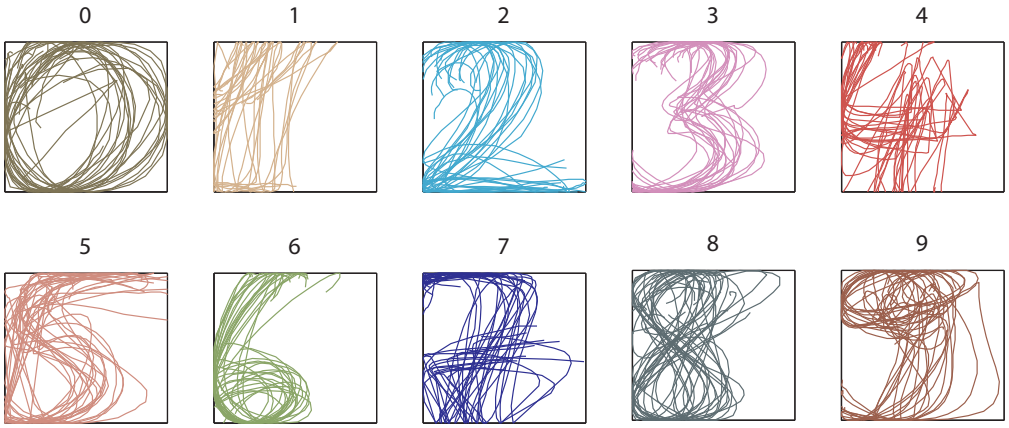


Figure 4.11: Example trajectories from the Pen-digits dataset. See text for further details.

4.4.2 How does trajectory representation affect separability?

The separability measures reviewed in Section 4.3 provide a way to estimate the overlap of a set of classes in a given representational space. Using these measures, we now evaluate the separability of the four trajectory datasets described in the preceding section, for each of the parametric trajectory representations reviewed in Section 4.2. Since each representational strategy allows a trajectory to be described with an arbitrary number of parameters, we measure the separability of each dataset/representation over a range of dimensionalities (specifically, between 8 and 32 dimensions).

Although Section 4.2 proposes the concatenation of the two M dimensional parameter vectors $\vec{C^X}$ and $\vec{C^Y}$ (representing original coordinate sequences $\vec{X}$ and $\vec{Y}$ respectively) with the total time taken t_N to yield a single $2M+1$ dimensional vector, we omit t_N from the trajectory representation vectors used in the following experiments. Since the resulting vectors would need to be rescaled if time were incorporated, this allows the separability of different representations to be compared independently of any particular choice of rescaling strategy.

For each representation and its corresponding range of dimensionalities, Figure 4.12 shows the separability values obtained using J_{RNG}, the proportion of weighted edges joining members of the same class in a relative neighbourhood graph spanning each dataset. Similarly, Figure 4.13 shows the values of J_{NN}, the leave-one-out 1-nearest-neighbour classification rate. For completeness, the within-vs-between class scatter values, J_S, are also shown in Figure 4.14, although for reasons discussed in Section 4.3 these do not form the basis of our discussion. In all figures, each row shows results for a given dataset: the left hand plot shows results for representations parametrised by time while the right hand plot shows results for those parametrised by arc-length; plots within each row have identical Y-axis scale/limits to facilitate the

comparison of time/arc-length parametrisations.

Which representation works best? It can immediately be seen that in most cases the differences in separability values are small, with the observed values typically spanning a range of around 0.05 (the J_{RNG} and J_{NN} measures assign values between 0 and 1). The results do not, therefore, indicate that the choice of representation makes more than a small difference to class separability. A natural question to ask, however, is whether there is any persistent "ordering" corresponding to different representations within the small range of observed variation. In most cases, the Haar representation appears to improve upon the DFT representation, and the highest separability values are yielded by either the Chebyshev or Spline representations.

For the NGSIM and CAVIAR datasets, the Chebyshev representation appears to yield the best separability values. Noting that these datasets contain widely varying class-proportions, it is likely that J_{NN} will provide the most sensitive measure of separability as it takes into account differences in class size (in contrast the graph based J_{RNG} measure *is* affected by class proportions as classes with fewer members contribute fewer graph edges). A potential explanation for the comparatively high separability afforded by the Chebyshev representation on these datasets is that the coefficients for first two Chebyshev basis functions describe a straight line approximation to the trajectory: since the trajectories in these datasets have been labelled according to their start and end points, this may provide a substantial contribution to the separation of their locations in parameter space.

In contrast, for the ASL, Pen-digits, and Gun-point datasets, the Spline representation appears to yield the highest separability values. This difference is most apparent when trajectories are parametrised by arc-length: while trend this is less clear for temporal parametrisation, the highest separability values are provided by arc-length for these datasets (this is discussed next).

Which parametrisation strategy works best? The impact of the two different curve parametrisation strategies (ie. either proportion of total time or of total arc-length) can be examined by comparing the left/right hand plots for each dataset. For the CAVIAR, Pen Digits and Gun-point datasets, parametrisation by arc-length appears to improve separability in all cases. For the ASL and NGSIM datasets this trend is less clear, although in both cases the highest observed separability values correspond to arc-length parametrisation. As discussed earlier, J_{NN} provides a particularly useful measure for the CAVIAR and NGSIM datasets, as it is corrected for class proportions. In both cases the relative ordering of J_{NN} is very similar to that observed for the J_{RNG} plots, and

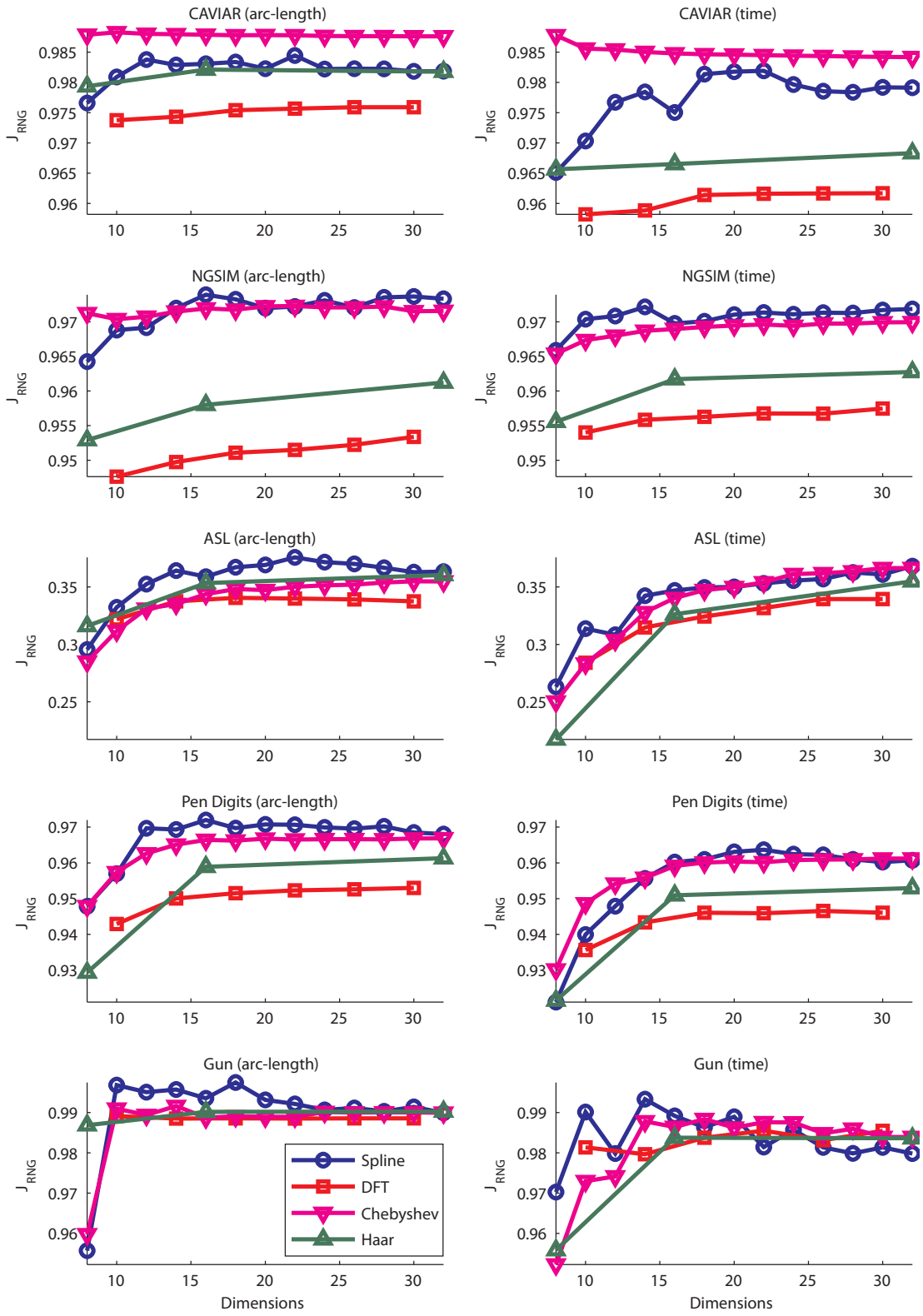


Figure 4.12: Graph-based separability measure J_{RNG} calculated for each dataset, using different trajectory representations over a range of dimensionalities. (Please note that the Y axis scale differs between plots.)

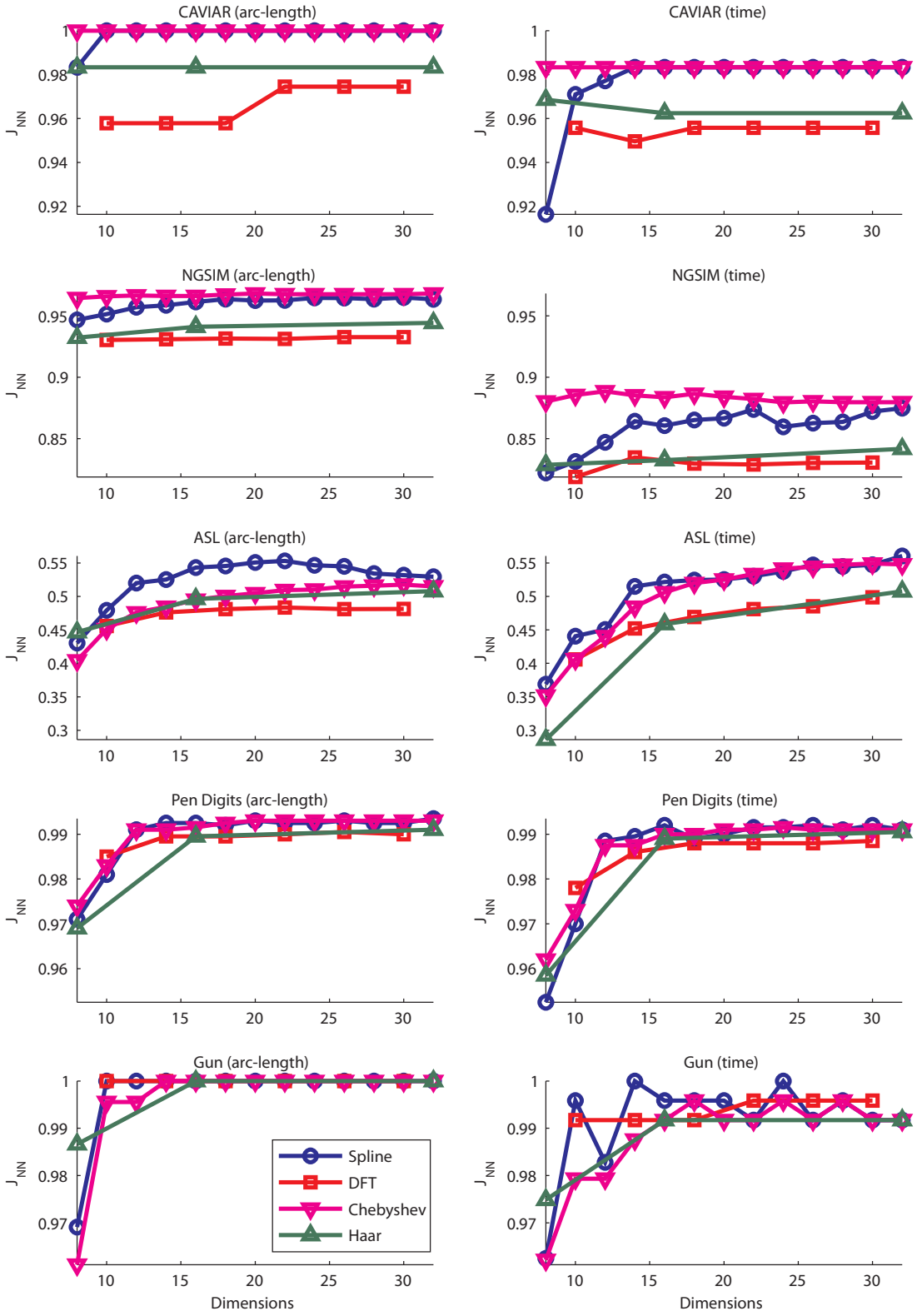


Figure 4.13: Leave-one-out nearest neighbour classification performance for each dataset, using different trajectory representations over a range of dimensionalities. (Note that the Y axis scale differs between plots.)

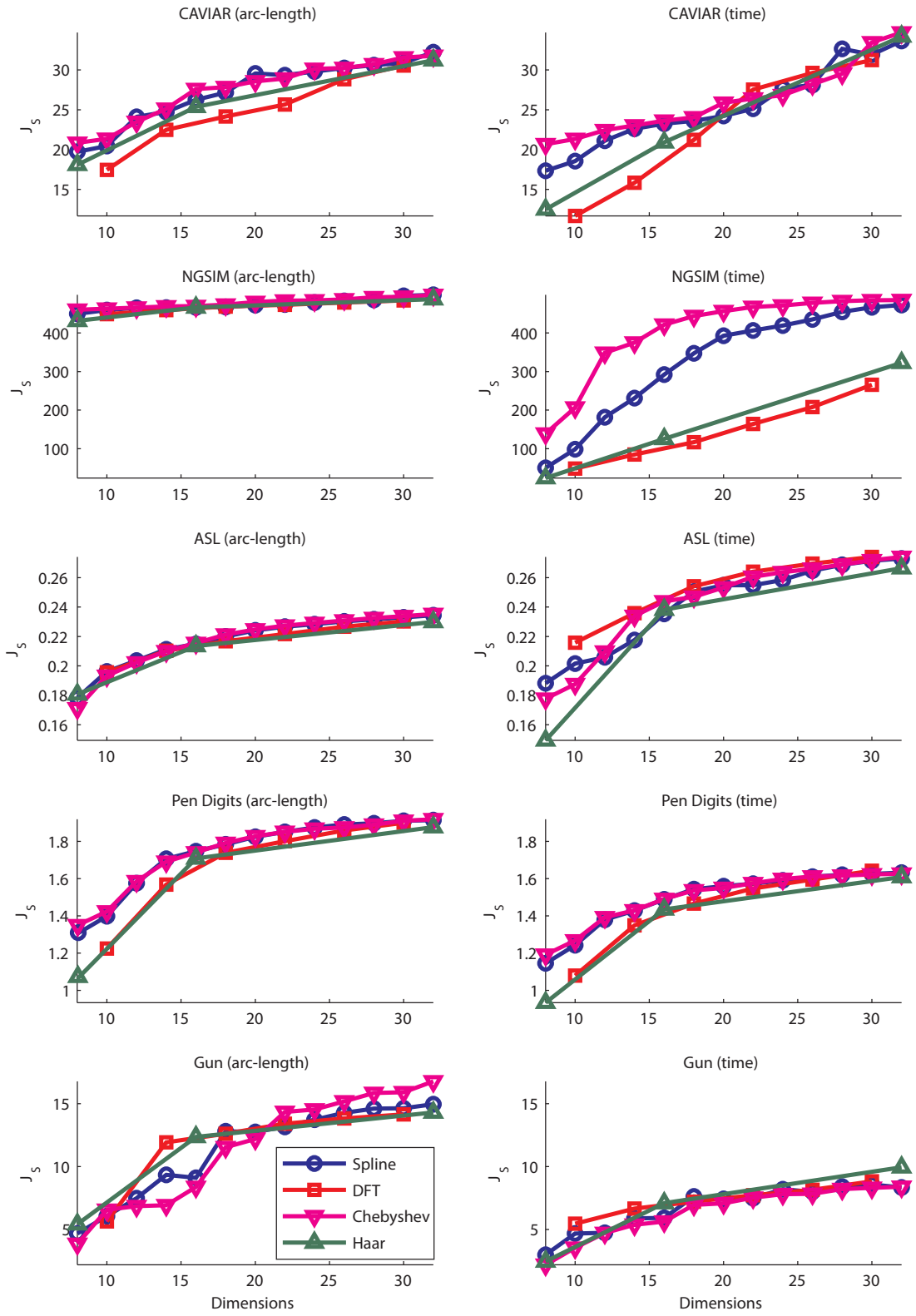


Figure 4.14: Within-vs-between-class scatter ratio J_S for each dataset, using different trajectory representations over a range of dimensionalities. (Please note that the Y axis scale differs between plots.)

arc-length parametrisation yields an improvement over time-based parametrisation: however, this is most significantly manifested for the NGSIM dataset where arc-length parametrisation yields a large ($\sim 10\%$) improvement in classification rate.

A clear potential reason for the sharp improvement observed when using arc-length parametrisation to represent the NGSIM dataset pertains to the wide range of different spatio-temporal profiles present in this data: since this data contains vehicle trajectories from a road segment which includes traffic lights etc., it is likely to yield a widely differing set of spatio-temporal profiles for a given route. While one would not automatically rule out the possibility of a single representation which captures spatial and spatiotemporal characteristics, the results obtained for NGSIM dataset indicate that a single spatio-temporal representation may be inadequate if a sufficiently large number of different spatial and spatiotemporal categories are present - regardless of the particular method used.

What impact does dimensionality have? It is clear from Figures 4.12 and 4.13 that the dimensionality of the chosen trajectory representation affects class separability. In the majority of cases, separability appears to increase when trajectories are represented with more parameters. Although this appears to be at odds with conventional wisdom regarding over-fitting (see [53, 12] etc.), it is important to note that the range of dimensionalities examined here (at most 16 parameters to describe an X or Y co-ordinate sequence) does not allow more than a coarse approximation of most of the trajectories considered here. In the majority of cases a peak level of separability appears to be reached between 14 and 20 dimensions, followed by a plateau.

In certain cases, however, increasing the dimensionality of the trajectory representation reduces the resulting level of separability. One example is the Chebyshev representation of the CAVIAR dataset, where maximum separability appears to be reached for dimensionalities below 10. As discussed earlier, a potential reason for this is that the first two Chebyshev basis functions (corresponding to 4 dimensions in the final representation) capture the most important discriminative information for this dataset: variation in the additional dimensions may serve to move members of a given class further apart in the resulting feature space. It is also worth noting, however, that in cases where separability for a given representation/dataset is observed to decrease with increasing dimensionality, the decrease is never catastrophic, with near-peak separability levels being maintained in all cases for up to 32-dimensions.

4.4.3 When might temporal parametrisation be necessary?

Although the preceding results indicate that arc-length parametrisation yields higher separability than temporal parametrisation, it is clear that the class structure in the datasets addressed so far is largely spatial in nature. It is therefore useful to provide some indication of the potential utility of temporal curve parametrisation. Despite the preceding results, it is possible that arc-length parametrisation may be inadequate for many trajectory classification purposes: for surveillance applications, the spatiotemporal characteristics of trajectory may capture the most important information for behaviour classifications.

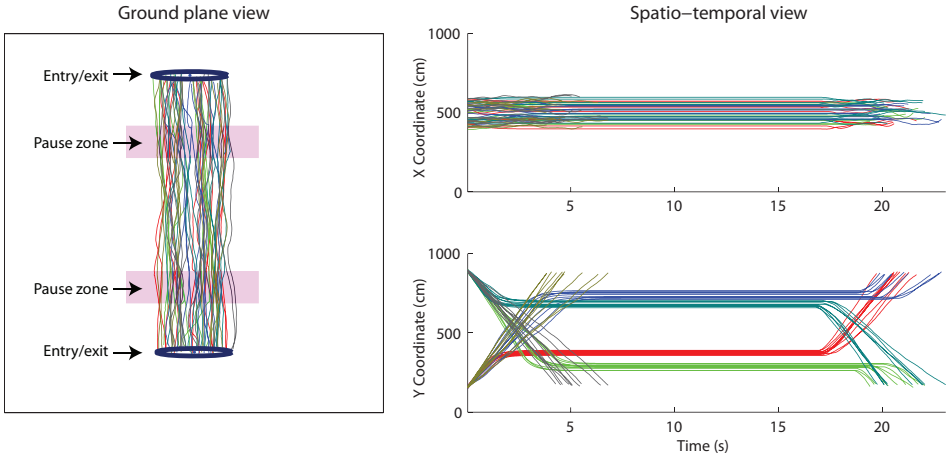
To illustrate this point, Figure 4.15a shows a simple synthetic dataset where simulated motion trajectories are generated between two possible entry/exit zones. Each traversal of this scenario (ie. up or down with respect to the ground plane view in Figure 4.15a), is associated with three different temporal profiles (either moving continuously, or pausing in one of two possible locations), leading to 6 distinct spatiotemporal classes. The spline representation from the preceding section (see Section 4.2.1 for details) was used to represent this dataset. Figure 4.15b shows the separability values (calculated using the J_{RNG} edge-weight statistic) for this dataset across a range of dimensionalities, using both the spatial and spatiotemporal spline representations: it is clear that the spatiotemporal representation separates the 6 classes almost perfectly, while the spatial representation only allows partial separation.

While the inclusion of total time elapsed could allow both representations to distinguish between trajectories with and without pauses, it would not enable the location of those pauses to be distinguished. The ability to discriminate between trajectories involving pauses in different locations is likely to be important in a surveillance context (where loitering in different locations may have different implications), indicating the potential utility of temporal trajectory parametrisation for certain applications.

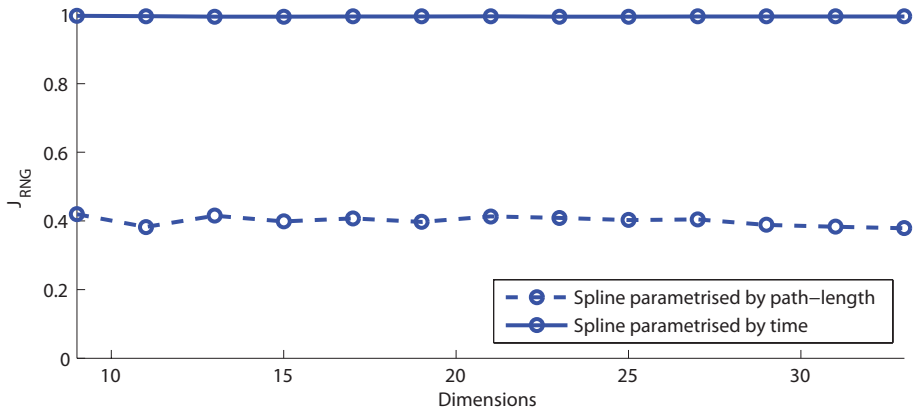
4.5 Discussion

This chapter has reviewed several different ways to parametrise motion trajectories with fixed-length vectors. Using the separability measures described in Section 4.3, the impact of trajectory representation on class separability has been examined for several real trajectory datasets.

The results did not, in general, indicate that the choice of trajectory representation made more than a small difference to class separability. Nonetheless, the Chebyshev and Spline representations appeared to provide the best class separability, with the former yielding the highest values for trajectories classified according to their



(a) Simulated trajectories with 6 distinct spatio-temporal profiles. Trajectories follow two possible directions; for each direction there are 3 possible temporal profiles: the simulated pedestrian may either pause (once, in a designated location) or progress without pausing.



(b) Resulting J_{RNG} separability values for cubic B-Spline trajectory representations of the above data, fitted using both time and path-length as the curve parameter.

Figure 4.15: The potential impact of spatial vs spatio-temporal trajectory representation on the separability of different behaviour classes (see text for description).

start/end points and the latter for trajectories classified according to their shape. Furthermore, a comparison of the separability of trajectories represented using both arc-length and temporal curve parametrisation indicated that arc-length parametrisation yielded higher separability values for the majority of datasets considered. However, a further experiment with synthetic data indicated that temporal parametrisation may be essential for trajectory classification in a surveillance scenario, implying that both strategies are worthy of consideration depending on the particular task at hand.

The experiments presented in this chapter have measured class separation based on Euclidean distances between raw/unprocessed feature vectors; it is thus possible that further processing of feature vectors corresponding to a given representation (eg. by re-weighting or rescaling attributes) may yield improved separation. It is in principle possible that such weightings could be learned from a given dataset, which highlights the problem of attempting to define separability as a quality distinct from a given learning method. However, given the density-based one-class learning/classification approach proposed in Chapter 3, where class labels within the set of "normal"-labelled data are unavailable, the definitions of separability employed here are likely to provide an adequate reflection of the impact different trajectory representations on classification performance.

It should be noted that the set of trajectory representations examined in this chapter is by no means exhaustive. For example, the choice of Haar wavelets (rather than one of many other possible wavelet families - see [137, 21]) relates to the fact that this wavelet has been previously used in the trajectory classification literature [120]; it is possible that other wavelet families may deliver very different, potentially improved, results. Similarly, the trajectory datasets considered here might not constitute an adequate characterisation of the different class-separation scenarios that may arise in real-world surveillance data (particularly when considering the problem of anomaly detection). Nonetheless, the experiments have shown that the choice of trajectory representation does have an impact on class separability as defined in this chapter.

The over-arching purpose of this chapter was to determine an appropriate fixed-length vector representation so that the distribution of normal trajectories could be incrementally modelled using the algorithm proposed in Chapter 3. To this end, the results presented in this chapter suggest that the Spline and Chebyshev representations are appropriate candidates, and that 14 (15 if time is included) dimensions is sufficient to achieve near-peak class separability for the datasets considered here. In this light we adopt a 15 dimensional Spline representation for the trajectory modelling experiments conducted in the next chapter, noting that it appears to provide a good compromise between capturing route and shape information.

Chapter 5

Semi-supervised Learning for Anomalous Trajectory Detection

5.1 Introduction

The review of existing approaches to behaviour learning/classification presented in Chapter 2 indicated that there is a paucity of techniques for incrementally learning models of normal/typical behaviour. This chapter attempts to address this issue by combining the incremental one-class learning algorithm proposed in Chapter 3 with the "hard-coded" parametric trajectory representation approach explored in Chapter 4, to provide a method for incrementally modelling the distribution of motion trajectories corresponding to normal behaviour. The resulting incremental trajectory modelling strategy then provides the foundation for a novel solution to the problem of incorporating labelled and unlabelled data in the learning of normal behaviour models, which forms the key focus of this chapter.

As discussed in Chapter 2, the most common strategy underlying systems for detecting anomalous behaviour is to model the distribution of observations from a given feature space in an unsupervised fashion. Here the term "anomalous behaviour" refers to behaviour worthy of scrutiny in a surveillance context: an intuitive assumption is that such behaviour is rare, and therefore not reflected in models which capture the underlying distribution of a large collection of behavioural data. Outside the unprecedented circumstance of having a representative corpus of anomalous behaviour examples (or having a rule based model eg. [32]), unsupervised learning is a necessary precursor to anomalous behaviour detection. The resulting models - which capture the underlying distribution of a set of training data - can then be used for behaviour classification, based on the assumption that a low probability (or an equivalent heuristic in the case of non-probabilistic models) is assigned to examples of anomalous behaviour.

The potential violation of this principle, however, may present a risk if behaviour classification models based on unsupervised learning are regularly deployed in different real-world settings. In each case, learning equates to modelling a finite sample of unlabelled data, which is assumed to be representative of the true distribution of behaviours that occur in a given scenario. It is possible, therefore, that on some occasions the available selection of unlabelled training data may contain multiple instances of a particular, relatively unusual, behaviour which could distort subsequent normal-vs-anomalous classification boundaries.

We propose to address this issue by implementing a semi-supervised training framework, where a model of normal behaviour is constructed incrementally in a largely unsupervised fashion with the following important exception: the approval of a human operator is requested before any seemingly unusual behaviour pattern (according to the model constructed so far) is used to train the model. This training strategy provides a way to make parsimonious use of human attention as a "safety net" to ensure that only normal examples are used to construct the resulting behaviour classifier. Moreover, given that current automated surveillance approaches are intended to assist, rather than replace, human operators, the proposed strategy provides an intuitive way to exploit the opportunity for occasional human feedback when training such systems.

The proposed trajectory-based behaviour modelling strategy is described in detail in Section 5.2.2, and its relationship to several existing machine learning approaches is discussed in Section 5.2.3. Finally, experimental results characterising the behaviour of the proposed algorithm are presented and discussed in Sections 5.4 and 5.5.

5.2 Incremental Semi-supervised Normal Behaviour Modelling

5.2.1 Behaviour Representation

The work presented in this chapter concerns an algorithm for incrementally learning from/classifying a sequence of whole trajectories, as produced by a pedestrian/vehicle tracking algorithm (see [174] for a review of existing strategies). While the requirement that *whole* trajectories are used as training/test data could be regarded as a practical shortcoming, it nonetheless provides a useful initial basis for exploring the characteristics of the proposed learning framework.

As discussed in Chapter 2, incremental behaviour learning algorithms require either a specific feature learning stage or an appropriate hard-coded behaviour representation. Here we employ a hard-coded behaviour representation, adopting the parametric trajectory representation approach discussed and examined in Chapter 4. The

comparison of trajectory representations presented in Chapter 4 indicated that a 14-dimensional (7 control point) cubic B-Spline representation provided a good trade-off between separability and dimensionality for a range of different datasets. This representation, illustrated in Figure 5.1, is used as the basis for behaviour modelling in this chapter: a key assumption of our approach is that differences between trajectories corresponding to different behaviours will be reliably reflected by distances in this representational space. Since Chapter 4 indicated that temporal and arc-length parametrisation may yield improvements depending on the nature of the classification task at hand, we initially explore both techniques in this chapter.

To remove temporal scaling invariance, each trajectory is represented by concatenating its corresponding 14 dimensional vector of spline control points with the total time taken t_N to yield a single 15 dimensional vector. In the absence of further processing, however, it is clear that the Euclidean distance between two trajectory vectors will depend on the particular units used to describe spatial coordinates and time. In order to remove this dependence, we rescale each dataset - once it has been parametrised with a given representation - so that its attributes all lie on the interval $[-1,1]$. We rescale the first M attributes according to a global maximum/minimum value for those attributes, rather than rescaling each one separately: this avoids - in the case of Spline and Haar wavelet representations - parameters corresponding to different regions of the trajectory being arbitrarily emphasised.

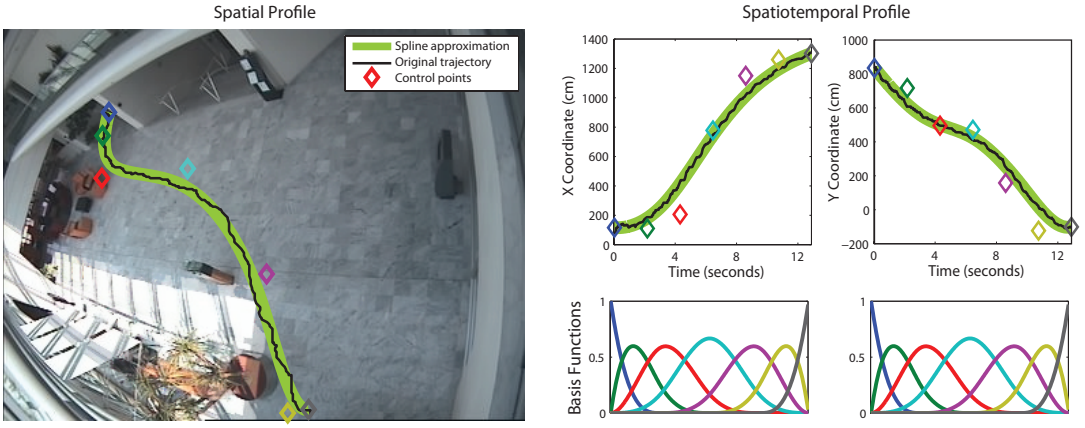


Figure 5.1: Spatiotemporal trajectory approximation using cubic B-spline curves.

5.2.2 Learning Algorithm

The learning algorithm proposed in Chapter 3 provides a way to incrementally estimate the probability density function underlying a given dataset, building a deliberately over-complex Gaussian mixture model that can be used to define conservative

classification boundaries (see Chapter 3 for details) for outlier detection at any stage during training. We use this technique to incrementally model the distribution of a sequence of parametrised trajectories corresponding to normal behaviour in a given scenario, ie. to estimate $p(\vec{Z}|normal)$ where $\vec{Z}$ is an instance of the 15-dimensional B-Spline trajectory representation described in Chapter 4.

In the proposed framework, illustrated in Figure 5.2, trajectories are assumed to arrive in a sequence, where each new trajectory is classified according to the current estimate of $p(\vec{Z}|normal)$. By default, in the absence of any training data, all trajectories are classified as anomalous. If a new trajectory is classified as anomalous, it is brought to the attention of a human operator, who may either choose to label the trajectory as normal (in the case of a new instance of normal behavior not yet represented in the model), or to label the trajectory as anomalous - thereby excluding it from the model - in the case of a genuine anomalous event. Conversely, if a trajectory is classified as normal, it is automatically used to update $p(\vec{Z}|normal)$ without requiring human approval (this is known as "self-training" - see Section 5.2.3).

In summary, new trajectories can only be used to update the model representing $p(\vec{Z}|normal)$ if one of the following two conditions are met:

1. The trajectory is classified as normal according to the current model.

2. The trajectory is deemed to be normal by a human operator.

It is important to note that Condition 2 only occurs if Condition 1 is not met, which means that requests for human approval are not made unless the trajectory is deemed anomalous according to the current model. The proposed mechanism thus enables a human operator to filter the training data used to construct a normal-behaviour model, by only drawing their attention to the most unusual examples.

While it could be argued that the self-training framework provides no safe-guard against the accidental inclusion of misclassified anomalous instances it is clear that, by using a human to filter a certain proportion of training data, the proposed framework provides a potential improvement over an entirely unsupervised training scheme and would - at worst - yield a classifier which performs equivalently to one trained in an unsupervised fashion.

By allowing training to occur in a largely unsupervised fashion while making use of occasional human effort to confirm the labels of unprecedented - but potentially normal - events, the proposed approach provides a natural way to incorporate labelled and unlabelled data in a one-class learning problem in a cost-effective manner.

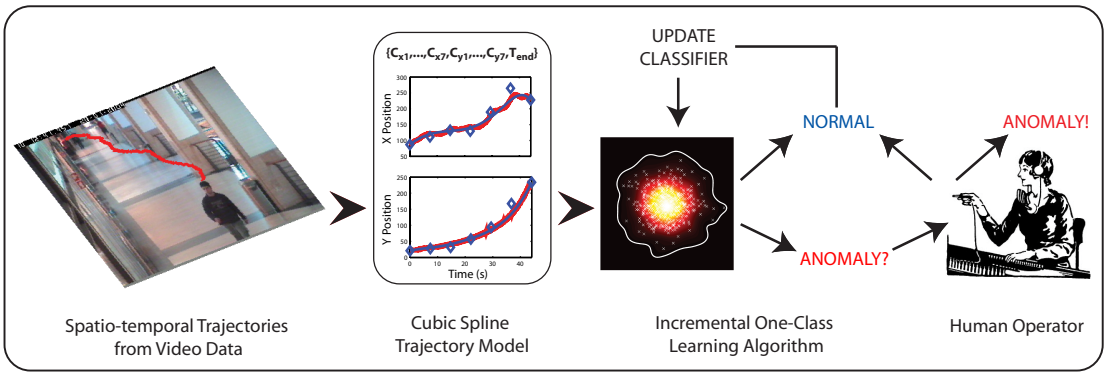


Figure 5.2: **Self-training framework for (normal) motion trajectory modelling:** Trajectories produced by a tracking algorithm are represented with vectors of cubic spline control points (see Chapter 4) and then assessed by a classifier. New examples classified as normal are automatically used to train the classifier (see Chapter 3), while anomalous examples are passed to a human operator for approval.

5.2.3 Related Approaches

There is a wide body of research exploring the incorporation of labelled and unlabelled data in the solution of classification problems. To place the proposed learning framework in context, this section briefly discusses its relationship to two relevant classes of machine learning strategy.

Semi-supervised Learning The proposed approach provides a way to integrate labelled data (normal examples confirmed by an operator) with unlabelled data in order to learn a classification rule. The combination of labelled and unlabelled data to solve learning/classification problems is known as Semi-Supervised Learning [26, 180], and can be implemented in a variety of different ways.

The proposed approach corresponds to a form of semi-supervised learning known as *Self Training*. This technique is a wrapper method for supervised learning algorithms which allows unlabelled data to be used for training through repeated iterations of the following procedure: given a set of labelled and unlabelled data, a classifier is trained using the available set of labelled data and then used to provide labels for the set of initially-unlabelled data. Those items of unlabelled data which were classified with high confidence are then added to the original labelled training set, which is then used to iteratively retrain the original classifier. This procedure was originally proposed by Yarowsky in [172] for the text classification, and has recently been applied to the problems of audio classification (Moreno and Agarwal [87]) and object detection (Rosenberg et al. [117]). In all cases the underlying supervised learning al-

gorithm is trained on examples with hard (ie. discrete, non-probabilistic) labels, but classifies new examples on the basis of a continuous value (eg. posterior probability) which can be used as an accompanying "confidence" measure for the classification. In [172] a threshold is placed on this confidence value in order to determine which unlabelled examples can be used for self-training, while in [87, 117] the example classified with the highest confidence value at each iteration is used.

The implementations of self-training in [172, 117, 87] differ in certain respects: in [117, 87] self-labelled examples retain their label for all subsequent training iterations, while in [172] such examples can lose their labels if their corresponding confidence measure drops below the original labelling threshold. A number of related approaches [96, 7] assign unlabelled examples with "soft" probabilistic labels which are utilised during training and re-estimated at each iteration: cast in this form, the self-training process is equivalent to the *Expectation Maximisation* algorithm. However, the self-training component of the proposed approach most closely resembles that of [117, 87] as hard labels are assigned/used during self-training and these cannot subsequently change: the latter is unavoidable as the proposed algorithm is required to learn incrementally given a single presentation of each example. The need to train on singular examples means that the proposed approach also resembles [172], as a classification/confidence threshold is used to determine whether or not a new example is suitable for self-training.

The algorithms discussed thus far all correspond to discriminative problems where training data is provided from multiple classes: the proposed approach is unusual in that the underlying supervised learning algorithm assumes labelled data from one class only. In this light there is another seemingly relevant set of semi-supervised learning approaches concerned with learning from *Positive and Unlabelled Data* eg. [36, 78]. In these approaches classifiers are trained on unlabelled data accompanied by labelled data from a single "positive" class with the aim of identifying whether or not new members belong to that class. In [36], Denis et al. build a classifier by estimating $P(x)$ from the entire dataset and $P(x|positive)$ from the subset of positive-labelled data (in [36] these are Naive Bayes models of word frequencies in documents): then, given knowledge of $P(positive)$ it is possible to estimate $P(x|negative)$ and thus to generate a classifier. In [78], Liu et al. use EM to iteratively refine models for positive and negative documents, while fixing the labels of positive documents during EM iterations. Both of these approaches rely on the assumption that the set of unlabelled data contains a significant proportion of non-positive data: in contrast, the proposed learning framework operates on the assumption that anomalous data is extremely rare, and so must be able to learn from unlabelled data which may contain no

anomalous examples.

There are many other interesting semi-supervised learning techniques (eg. graph based methods, transductive SVMs, etc. [26, 180]) that have not been discussed here, as they are not readily applicable given the specific task constraints (incremental one-class learning) posed in this thesis. Similarly techniques such as co-training (Blum and Mitchell [14]) and tri-training (Zhou et al. [179]), which make use of multiple classifiers trained on different features to provide labels for unlabelled data, are not applicable as the whole-trajectory behaviour representation adopted here does not naturally decompose into separate independent features.

Active Learning Since the proposed learning framework relies on the ability to request labels for certain data points, it can also be regarded as a form of Active Learning (see Settles [129] for an overview). The term "Active learning" refers to a class of machine learning strategies where the underlying learning algorithm can select/generate unlabelled queries to be labelled by a human operator (or "oracle" [129]). Thus, unlike the preceding semi-supervised approaches - which make use of unlabelled data - all the training examples used in an active learning framework are labelled. The key problem addressed by active learning algorithms is how to choose examples that will improve classification performance when labelled and added to the training set.

Many active learning approaches assume the existence of a large set of unlabelled examples, and operate on the basis of metrics for ranking this set of examples in order to chose the "best" example to query; in contrast, the proposed algorithm must make a sequence of choices about whether to query the labels of single examples. Nonetheless, two querying strategies from the active learning literature bear a strong resemblance to the approach adopted here: one is "uncertainty sampling" (Lewis et al. [77]) which chooses examples for which classification confidence is lowest; another related strategy, intended to deal with a stream of examples, defines a "region of uncertainty" in state space (Cohn et al. [28]) and queries new examples lying within this region. In the proposed framework we classify new examples (and thereby choose those to query) by thresholding a value which quantifies how likely it is that a new example belongs to the normal behaviour class: it is clear that this value can be interpreted as a measure of one-class classification confidence/uncertainty; equally, placing a threshold on this value can be viewed as defining a region of uncertainty.

5.3 Datasets and Preliminary Experiments

5.3.1 Datasets

This section describes three different trajectory datasets, and how they are used to test the proposed behaviour learning/classification algorithm. In order to give an indication of the relative separability of the normal and anomalous instances in each dataset, Figures 5.7 and 5.8 show scatter plots corresponding to 2-dimensional Principal Components Analysis and Fisher Linear Discriminant projections for each dataset: while these plots cannot provide a conclusive measure of separability, they suggest that the NGSIM dataset may present the most challenging classification problem as there is no 2D projection where normal and anomalous examples are well separated.

CAVIAR "INRIA" Dataset The publicly available CAVIAR dataset[1] consists of video footage and tracking data for a range of behaviours performed by actors in the entrance lobby of INRIA Labs, and contains around 60 complete tracks. We selected a subset of 21 tracks to represent normal behaviour, consisting of people walking directly from one exit point to another. We then selected a further subset of 19 tracks to define anomalous behaviour, consisting of actors fighting, falling down, and leaving/collecting packages. Although this data, shown in Figure 5.3, clearly encapsulates the type of problem an anomalous trajectory detection algorithm should be able to solve, it is not sufficient for testing our algorithm as there are no further examples available for training.

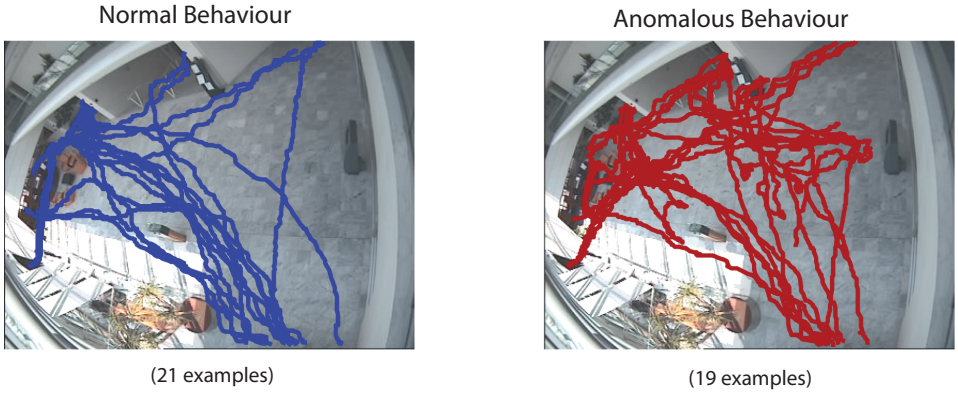


Figure 5.3: Test trajectories from the CAVIAR "Inria" dataset. See text for description.

The absence of additional training data is addressed by simulating a large set of ordinary walking behaviour between the entry/exit points featured in the test sets. For each pair of entry/exit locations, a route is hand-defined by a set of elliptical re-

[1] Available at: `http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/`

gions which represent entry/exit points and way points. This allows us to generate a diverse collection of possible paths by drawing sets of samples from these regions. Each set of samples is then interpolated to form a series of subgoals, which are used to generate a realistic trajectory in conjunction with the model for instantaneous pedestrian dynamics proposed by Helbing and Molnar in [56]. We define a route model for each of the 11 entry-exit pairs which appear in the test data and generate 100 simulated tracks for both traversal directions of each route, resulting in a total of 2200 tracks. Figure 5.4 shows the elliptical regions defining one of the 11 routes (in ground plane coordinates) between two exit points and a random subset of simulated tracks (10 from each route) projected onto the image plane.

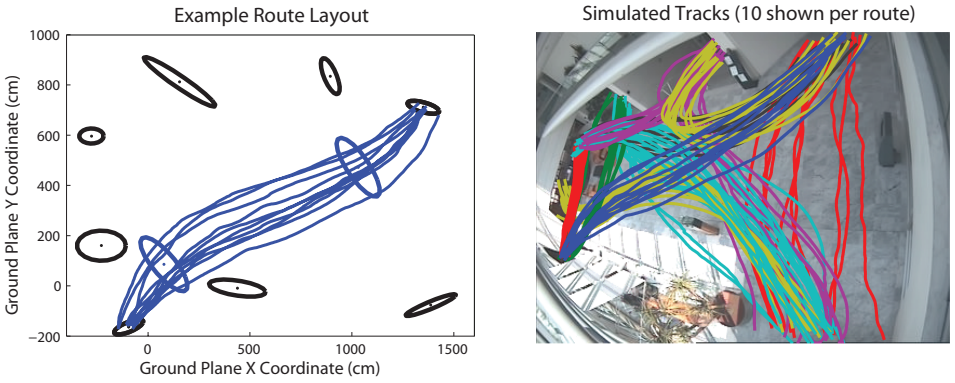


Figure 5.4: Normal behaviour simulation for CAVIAR scenario. Left: ellipses indicate all entry/exit points, and way points for a given route. Right: example simulated tracks coloured by route.

Leeds Carpark Dataset While the preceding dataset provides a means to test whether the proposed model can be used to classify real test data, the resulting performance will be highly dependent on the nature of the simulated training data. It is thus important to ascertain that similar behaviour can be obtained when training on real tracking data instead of simulations.

To establish this we use another dataset created by Dee and Hogg [32, 34], which contains examples of pedestrian and vehicle trajectories obtained by applying a tracking algorithm (proposed by Magee in [81]) to video footage obtained from a car-park scenario. The resulting dataset, shown in Figure 5.5, consists of 262 trajectories corresponding to ordinary walking/driving behaviour, and a further set of 6 - deliberately circuitous - trajectories corresponding to the behaviour of actors.

While this is a relatively small set of data, it affords us the possibility of examining the classification performance/intervention rates attained during the early stages of training with the proposed algorithm, without requiring simulated training data. We split the data so that 235 (ie. 90%) of the normal examples are used for training

our algorithm and 27 are retained for testing, along with the 6 anomalous examples provided by actors.

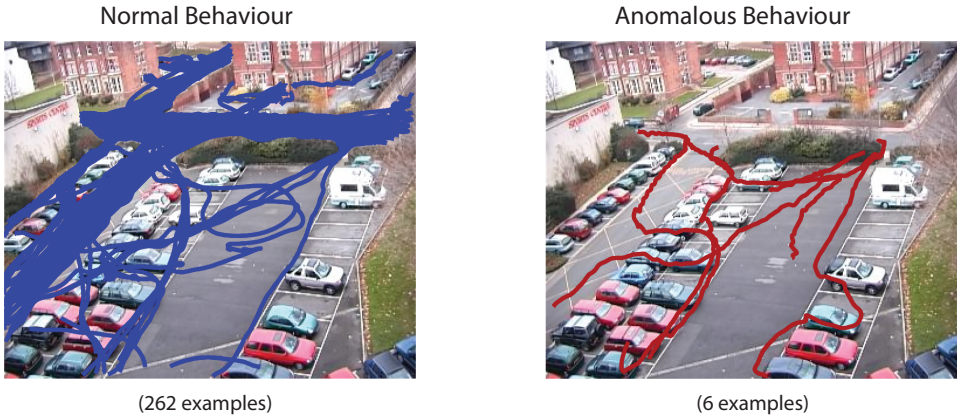


Figure 5.5: Trajectories from the Leeds carpark dataset. See text for description.

NGSIM "Peachtree Street" Dataset Finally we adapt the NGSIM "Peachtree Street" vehicle trajectory dataset [23] (used in Chapter 4 for comparing trajectory representations) to provide another normal-vs-anomalous classification problem accompanied by real training data. As described in Chapter 4, this dataset contains over 2000 different vehicle trajectories, which are labelled according to their entry/exit locations, and can be divided into classes on this basis. As in Chapter 4, we disregard any trajectory lasting for fewer than 2 seconds, which yields a set of 2007 trajectories.

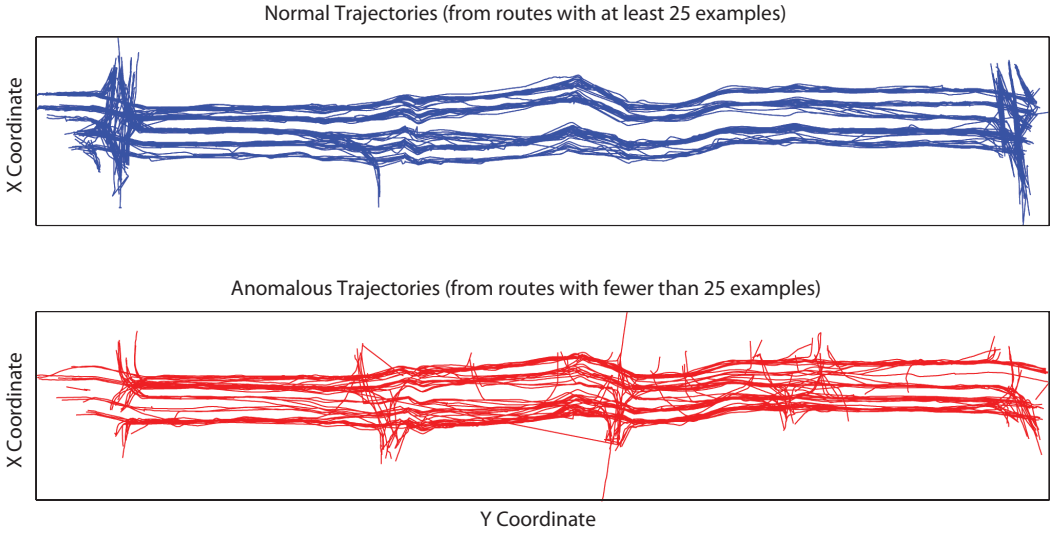


Figure 5.6: NGSIM "Peachtree Street" dataset. See text for description.

We use this dataset to provide a normal-vs-anomalous classification problem by treating trajectories belonging to classes with at least 25 members as normal, and all

others as anomalous. This yields a set of 1758 "normal" trajectories (33 classes), of which 1582 (approximately 90%) are used for training, and the remaining 176 are used for testing, together with the 249 (91 classes) designated "anomalous" trajectories. Figure 5.6 provides an illustration of this dataset.

5.3.2 How should the classification boundary be defined?

In order to apply the one-class learning algorithm proposed in Chapter 3 within the semi-supervised learning framework described in Section 5.2.2, it is necessary to pre-specify a classification threshold. Chapter 3 highlighted two different methods for defining a classification boundary by to do this.

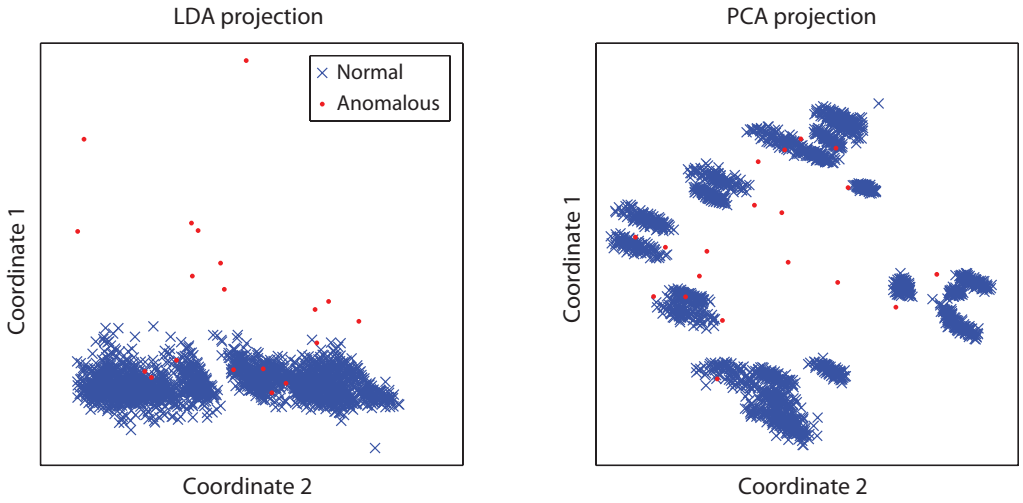
1. Impose a density threshold, chosen such that a certain proportion (a user specified value) of training data are deliberately misclassified as anomalous: if the probability density at the location of a new example falls below this threshold it is classified as anomalous.

2. Impose a component-wise threshold based on Extreme Value Theory [109]: if the deviation of a new example from its closest component has sufficiently low probability (a user specified value), it is classified as anomalous.

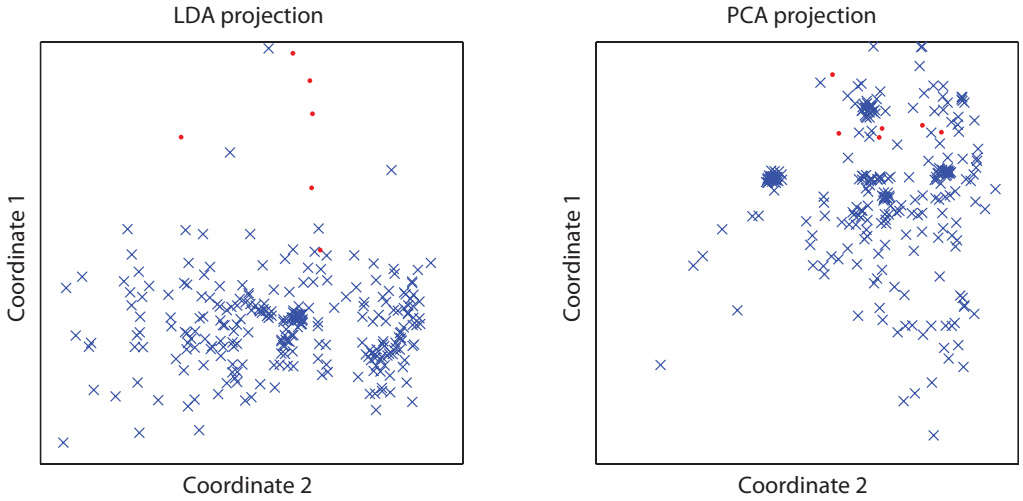
Rather than pre-selecting the best classification method and threshold parameter for each individual dataset, we attempt to determine a reasonable choice for all datasets. Specifically, we evaluate both classification methods, using several different thresholds, for all three datasets[2]. Figures 5.9 and 5.10 (corresponding to trajectory representations based on time and arc-length respectively - see Section 5.3.3) show how classification performance changes as training progresses for each dataset/classification method: each plot shows 8 different curves corresponding to either EVT (thin lines) or density boundaries (thick lines) with one of the following thresholds: 0.5 (black); 0.2 (blue); 0.1 (purple) and 0.05 (red). In each Figure, the left hand plot show the True Positive Rate (TPR - the proportion of correctly classified normal trajectories) while the right hand plots show the False Positive Rate (FPR - the proportion of incorrectly classified anomalous trajectories).

From the set of 8 classification boundaries tested, the EVT-based boundary with a classification threshold set at $P < 0.2$ appears to provide the best all-round performance. For the Leeds and CAVIAR datasets (temporal trajectory representation) this boundary gives the highest TPR values when FPR equals zero, while also providing a

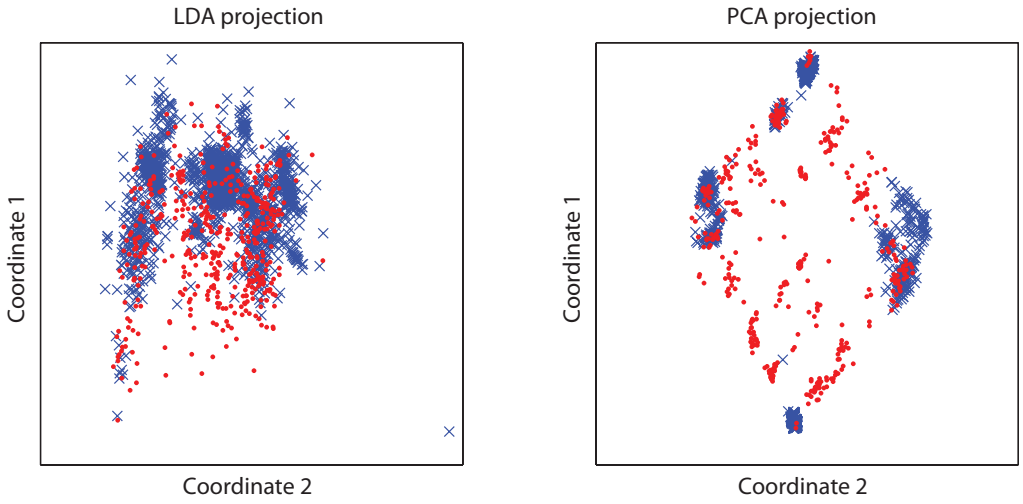
[2]In each experiment the L_{IQR} function (see Chapter 3) is used to estimate the kernel parameter σ and, as in Chapter 3, the maximum number of mixture components is set to 100 in all experiments.



(a) CAVIAR "INRIA" Dataset

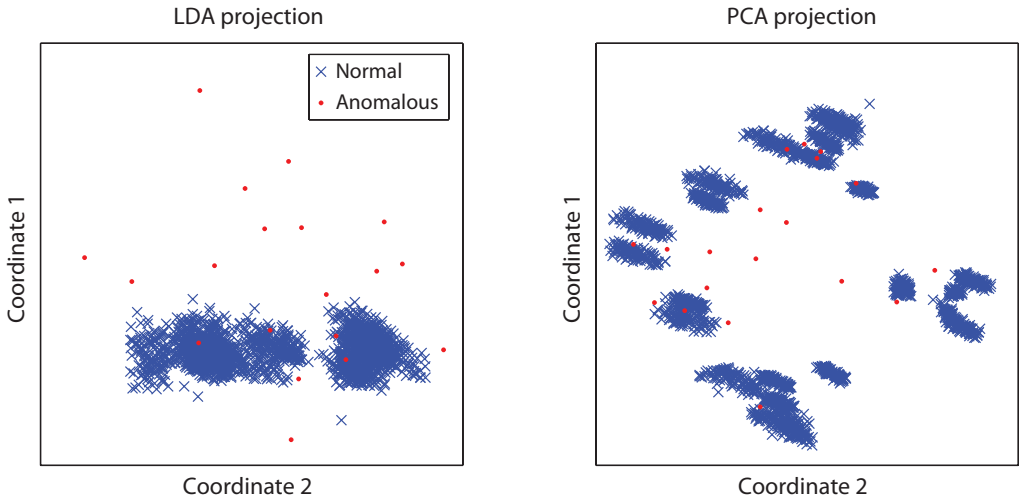


(b) Leeds Carpark Dataset

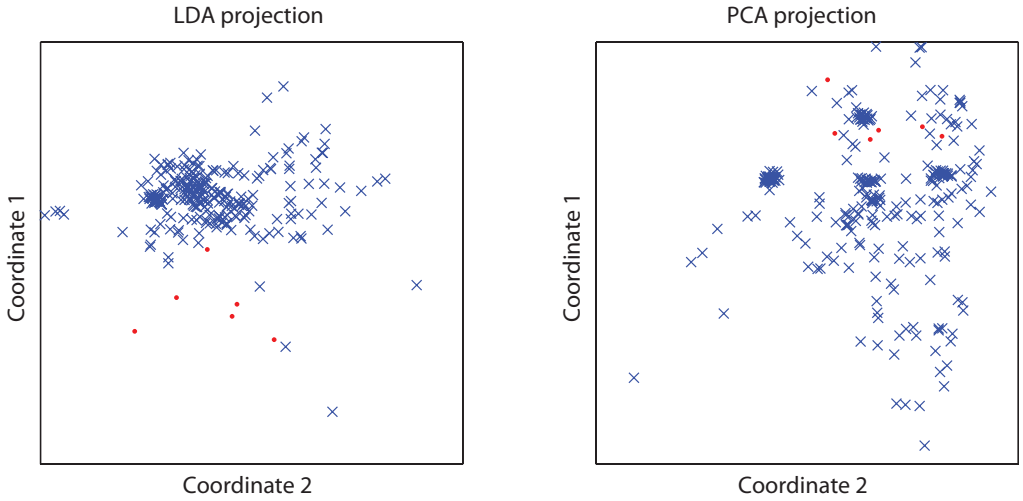


(c) NGSIM "Peachtree Street" Dataset

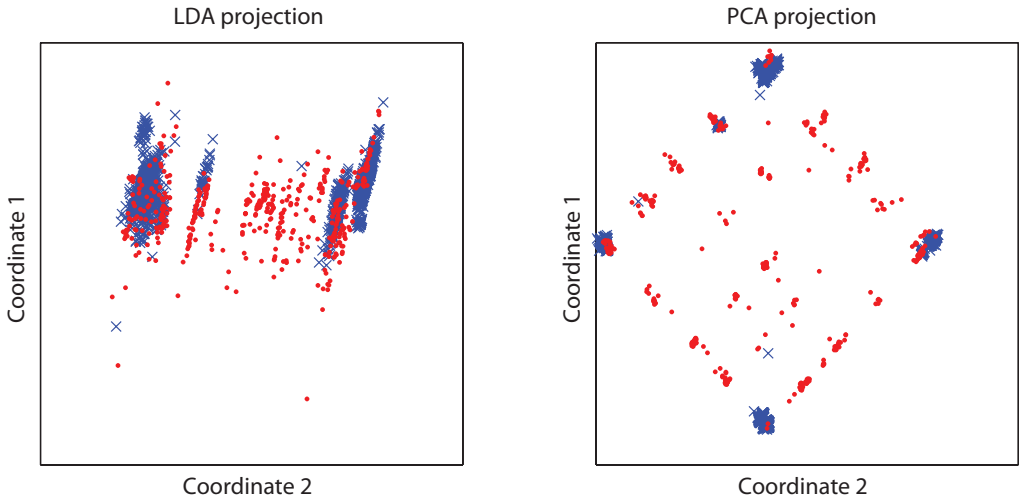
Figure 5.7: 2D Visualisation of trajectory datasets using PCA and LDA projections (from 15-dimensional spline representation with temporal parametrisation).



(a) CAVIAR "INRIA" Dataset



(b) Leeds Carpark Dataset



(c) NGSIM "Peachtree Street" Dataset

Figure 5.8: 2D Visualisation of trajectory datasets using PCA and LDA projections (from 15-dimensional spline representation with arc-length parametrisation).

Name	Interpretation
True Positive Rate (TPR)	Proportion of normal examples classified as normal
False Positive Rate (FPR)	Proportion of anomalous examples classified as normal
True Negative Rate (TNR)	Proportion of anomalous examples classified as anomalous
False Negative Rate (FNR)	Proportion of normal examples classified as anomalous

Table 5.1: Summary of the terms used to describe classification performance.

reasonable trade-off between TPR and FPR for the NGSIM dataset (none of the strategies allow zero FPR for this dataset). An arrow next to each plot indicates the final FPR/TPR values obtained with this boundary: unless otherwise stated, this classification strategy/threshold is used in the remainder of this chapter.

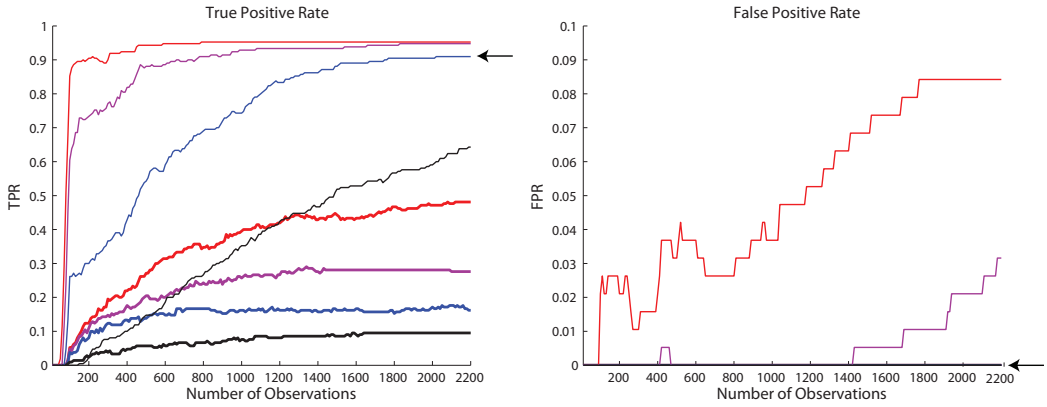
Since it is possible that the range of thresholds $[0.05, 0.5]$ measured in the preceding experiment does not encompass the best possible density-based threshold (whose values correspond to the proportion of observed training data that should be rejected, ie. classified as anomalous), Table 5.2 shows TPR and FPR values at the end of training for the best EVT threshold (0.2) and two further density thresholds (0.01, where 1% of training data is rejected and *min*, where no training data is rejected). The results show that the EVT (0.2) threshold provides better classification performance than the density (0.01) threshold on all datasets. While the density (*min*) threshold yields improvement over EVT (0.2) for the NGSIM and CAVIAR datasets, it nonetheless yields substantially reduced classification performance for the Leeds dataset, confirming that the EVT (0.2) is indeed an appropriate choice.

This experiment illustrates a key problem with using a density based threshold: It is, of course, possible that there exists a density value - outside the range present in the training data - that would yield better performance on the Leeds dataset than the chosen EVT boundary, but there is no way to pre-determine such a threshold in terms of a parameter that is not specific to the dataset itself.

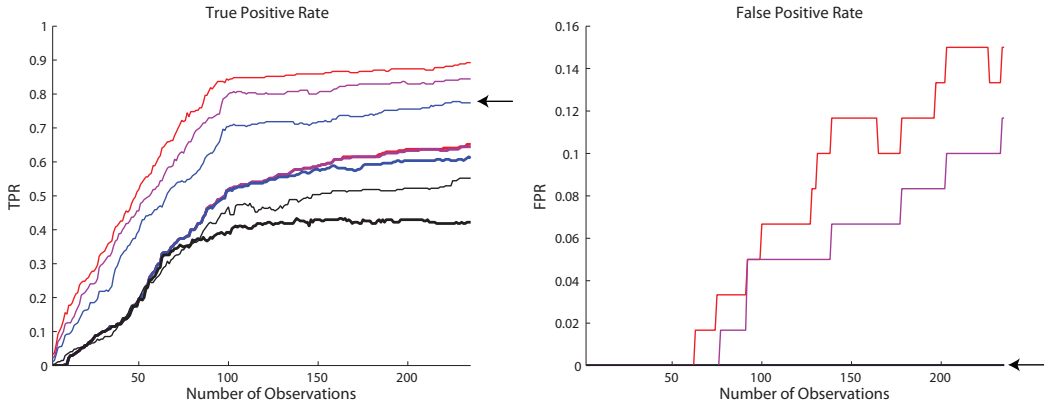
5.3.3 Spatial or temporal trajectory parametrisation?

The results presented in Figures 5.9 and 5.10 indicate that trajectory representations based on temporal parametrisation yield the best results for the CAVIAR and Leeds datasets, confirming the potential importance of spatiotemporal trajectory representations for anomalous behaviour detection suggested in Chapter 4. Conversely, as predicted by the results presented in Chapter 4, trajectory representations based on arc-length parametrisation yield the best results for the NGSIM dataset.

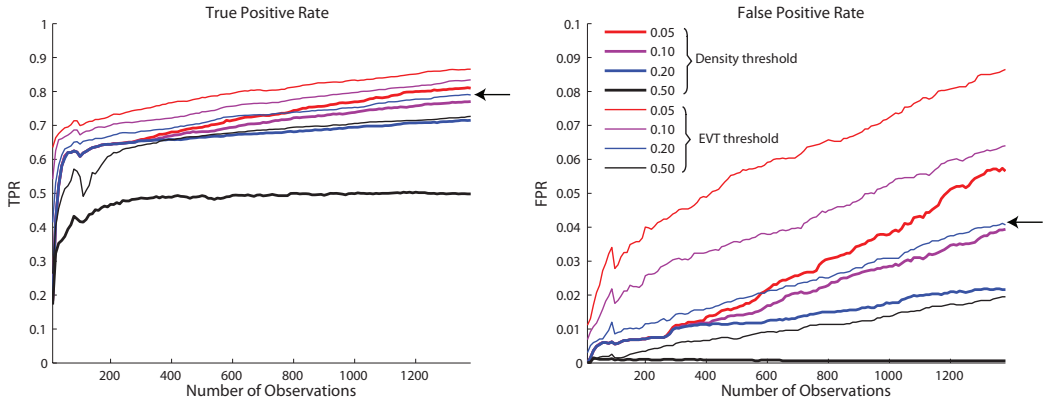
Since the primary concern of this chapter is to assess the learning framework



(a) CAVIAR "INRIA" dataset.

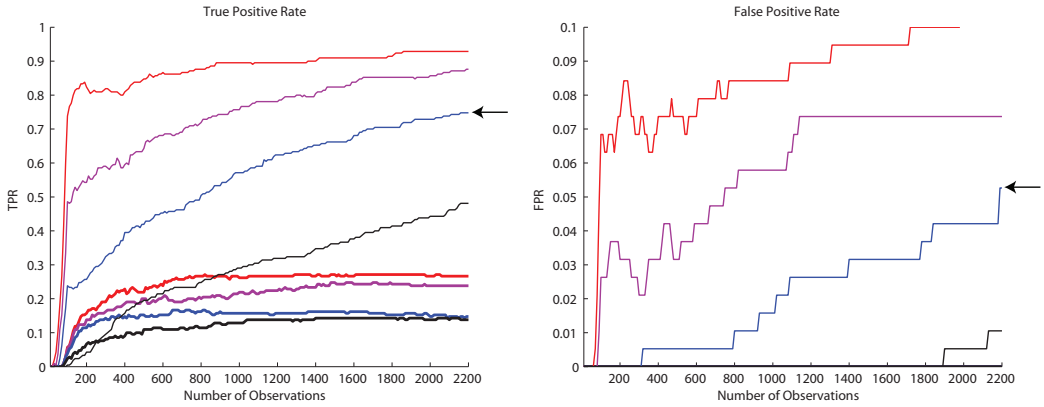


(b) Leeds carpark dataset.

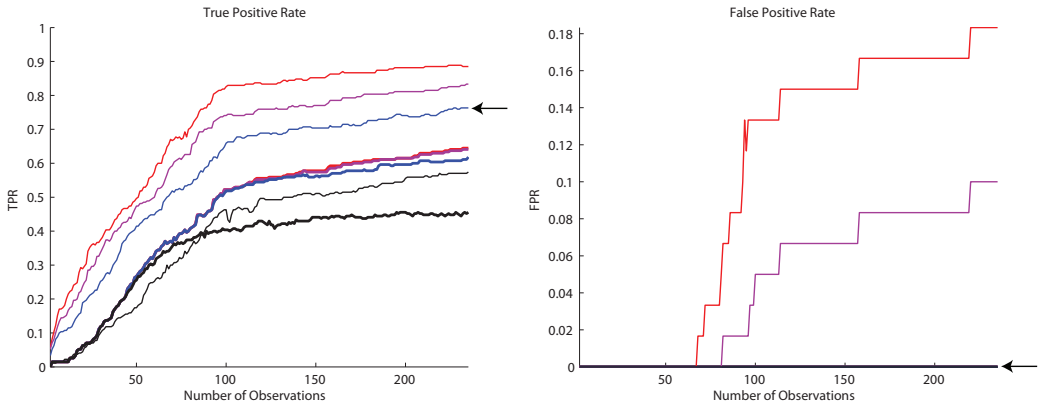


(c) NGSIM "Peachtree Street" dataset.

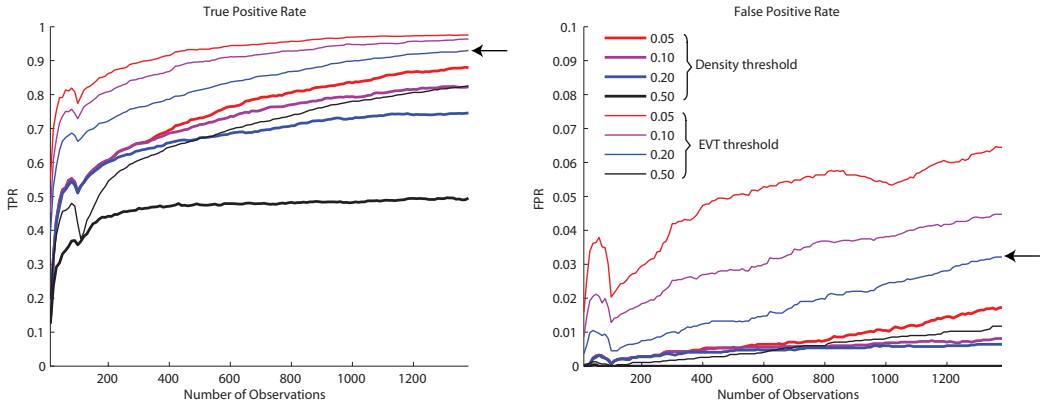
Figure 5.9: The impact of different boundary definitions (and their corresponding threshold values) on classification performance during incremental learning. Trajectories are represented using a 15-dimensional spline representation parametrised by time. Each curve represents the mean value over 10 trials. Note that the Y-axis scales/limits for left/right hand plots do not match: the (right hand) plots of False Positive Rates only cover a subset of the full range $[0,1]$ in order to facilitate comparison of the different curves (for FPR plots where only a subset of the curves are visible, the remaining curves are equal to zero).



(a) CAVIAR "INRIA" dataset.



(b) Leeds carpark dataset.



(c) NGSIM "Peachtree Street" dataset.

Figure 5.10: Repeat of experiment shown in Figure 5.9. Trajectories are represented using a 15-dimensional spline representation parametrised by arc-length. See text for discussion.

		Caviar (temporal)		NGSIM (spatial)		Leeds (temporal)	
		TPR	FPR	TPR	FPR	TPR	FPR
Density	*min*	0.9524	0	0.9604	0.0452	0.6556	0
Density	0.01	0.8190	0	0.9299	0.0325	0.6519	0
EVT	0.2	0.9095	0	0.9299	0.0321	0.7741	0

Table 5.2: Is there a density-based boundary that works better than the chosen EVT boundary? This table shows classification performance at the end of training for two extra density thresholds: 0.01, where 1% of training data is rejected, and *min* where no training data is rejected. Results are shown for the best curve parametrisation strategy for each dataset.

described in Section 5.2.2, in all subsequent experiments each dataset will be represented using the best method as identified in this section (temporal parametrisation for CAVIAR and Leeds datasets, and spatial parametrisation for the NGSIM dataset). While the results presented here and in Chapter 4 indicate certain limitations of parametric trajectory representations, it is important to note that the over-arching principle explored in this chapter be could applied to any suitable form of parametric behaviour representation.

5.4 Experiments

5.4.1 How do performance and intervention rate change with time?

Using the classification boundary identified in the Section 5.3.2, we now measure the number of operator interventions that would be incurred during training on each dataset, and examine how classification performance and the rate of user-intervention requests change as more training examples are observed. As in the preceding section, the proposed algorithm is trained on 10 different random permutations of normal trajectory data (split 90% train / 10% test). In addition to the True / False Positive Rates (which correspond to classification performance on the test data), we measure the proportion of the last 20 training examples that would have required human approval before being incorporated into the normal behaviour model. This quantity, referred to hereafter as the Intervention Rate, is also equivalent to the False Negative Rate calculated as a moving average from the training data.

Figure 5.11 shows the classification performance and intervention rates obtained for the three datasets. The True Positive Rate rises rapidly during the first 100-200 observations before increasing steadily at a slower rate, yielding final TPR values of

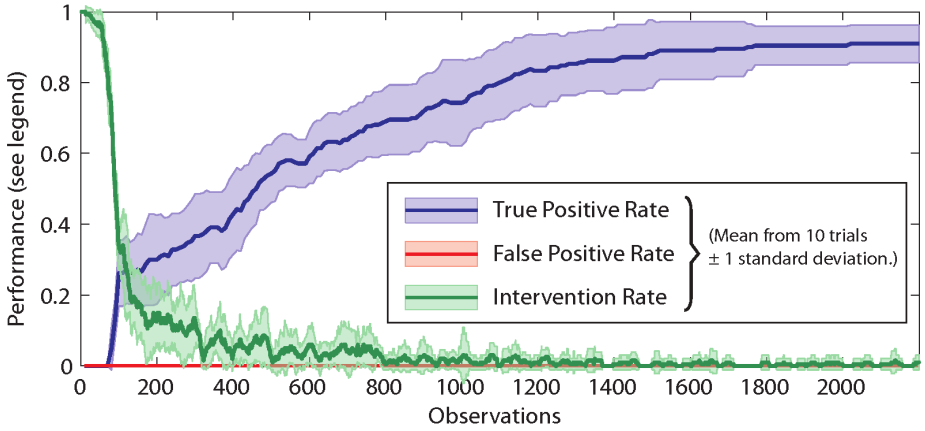
		Leeds	NGSIM	CAVIAR
235 Obs.	TPR	0.77 ± 0.095	0.74 ± 0.054	0.31 ± 0.11
	FPR	0	0.0077 ± 0.0037	0
1386 Obs.	TPR	-	0.93 ± 0.023	0.86 ± 0.094
	FPR	-	0.032 ± 0.0076	0
Final	TPR	0.77 ± 0.095	0.93 ± 0.023	0.91 ± 0.052
	FPR	0	0.032 ± 0.0076	0

Table 5.3: Classification performance obtained at different stages of training. (See text for discussion.)

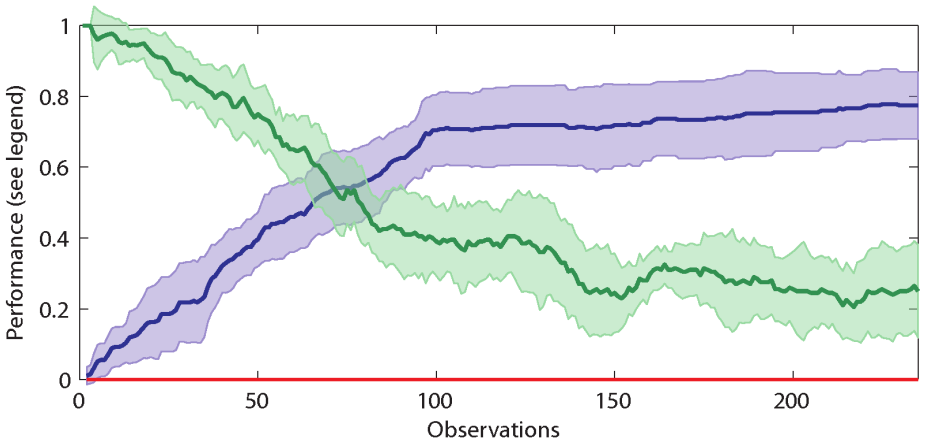
0.77 ± 0.095 (mean ± 1 standard deviation) for the Leeds dataset, 0.93 ± 0.023 for the NGSIM dataset, and 0.91 ± 0.052 for the CAVIAR dataset. For the Leeds and CAVIAR datasets, the False Positive Rate remains at zero for the duration of training, while for the NGSIM dataset FPR gradually increases to a (low) value of 0.032 ± 0.0076 at the end of training (the potential causes/implications of this increase are examined in Section 5.4.4).

Table 5.3 shows classification performance at the end of training, after 235 observations (the largest number where all datasets can be compared), and after 1386 observations (the largest number where the CAVIAR and NGSIM datasets can be compared). After 235 observations, classifiers trained on the NGSIM and Leeds datasets yield similar TPR values, while those trained on the CAVIAR dataset yield a much lower TPR value: a potential explanation is that the distributions of the synthetic training data and real test data used for this dataset are different, meaning that a larger quantity of training data is required to ensure that the resulting classification boundary encompasses the region of feature-space occupied by the training data. However, after 1386 observations, similar TPR values are obtained for both the NGSIM and CAVIAR datasets.

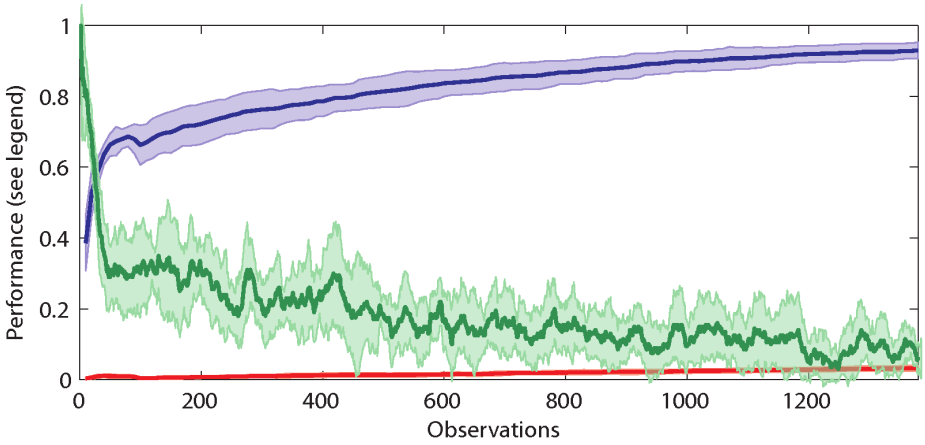
The Intervention rate follows a similar (opposite) pattern to TPR, dropping rapidly at the start of training before decreasing more slowly. For the Leeds and NGSIM datasets, the Intervention rate corresponds roughly to $1 - TPR$, as would be expected (since the Intervention rate is a moving average of $FNR = 1 - TPR$ calculated from the training data). However, for the CAVIAR dataset, the Intervention rate is lower than would be predicted from the TPR values: again, this may reflect the discrepancy between the distributions of the synthetic training data and real test data. Table 5.4 shows the number of interventions requested during different stages of training (for ranges of observations that can be compared between datasets). During the first 235



(a) CAVIAR "INRIA" dataset.



(b) Leeds carpark dataset.



(c) NGSIM "Peachtree Street" dataset.

Figure 5.11: Measuring classification performance and intervention rate as a sequence of training examples is presented. Curves/error bars correspond to the mean/standard deviation obtained for 10 different random permutations of normal training/test data. (See text for discussion.)

	Leeds (235)	NGSIM (1386)	CAVIAR (2200)
Obs. 1-235	103.6 ± 8.7 (44.1%)	78.8 ± 5.97 (33.5%)	100.7 ± 4.5 (42.9%)
Obs. 236-1386	-	166.6 ± 33.1 (14.5%)	37.3 ± 16.5 (3.2%)
Obs 887-1386	-	51.3 ± 10.9 (10.3%)	6.4 ± 3.4 (1.3%)
Total Interventions	103.6 ± 8.7 (44.1%)	245.4 ± 37.5 (17.7%)	141.9 ± 21.1 (6.5%)

Table 5.4: The number of operator interventions that would be incurred during different stages of training. (See text for discussion.)

observations (the total number available for the Leeds dataset) the number of interventions requested for each dataset is similar (with the NGSIM dataset incurring fewest interventions); thereafter (between 236-1386 observations) the intervention rate for the CAVIAR dataset falls to $\sim 3.2\%$, compared to $\sim 14.5\%$ for the NGSIM dataset. For the last 500 observations in the NGSIM dataset (887-1386), the intervention rates for the CAVIAR and NGSIM datasets are $\sim 1.3\%$ and $\sim 10.3\%$ respectively.

It is possible that the low intervention rate observed for the CAVIAR dataset reflects the simplicity/uniformity of the distribution of synthetic training data, and may thus provide an unrealistic indication of the intervention rate that could be achieved at this stage in training (ie. when 1000-2000 training examples have been observed). Indeed the TPR values observed for both datasets are similar at the end of training, indicating that the intervention rate for the CAVIAR dataset (as predicted by $1 - TPR$) may potentially be similar to that observed for the NGSIM dataset if real training data were used (conversely, however, such training data could have yielded a higher TPR and correspondingly low intervention rate).

The results presented here indicate that, depending on the nature of the dataset in question, an intervention rate between 1% and 10% may be obtained after training the proposed algorithm on ~ 1000 example trajectories, with 10% being the most realistic figure when taking into consideration the relative realism of each dataset. It is clear that, in a real-world scenario, this represents an unreasonably high intervention rate. However, since the results illustrated in Figure 5.11 show a steady improvement in classification performance for each dataset, they do not rule out the possibility that low intervention rates could be obtained given sufficient training data.

5.4.2 How many interventions are necessary?

While the preceding experiment examined how the proportion of training examples requiring user interventions might change as training progresses, the impact of user interventions on classification performance was not determined. To address this issue,

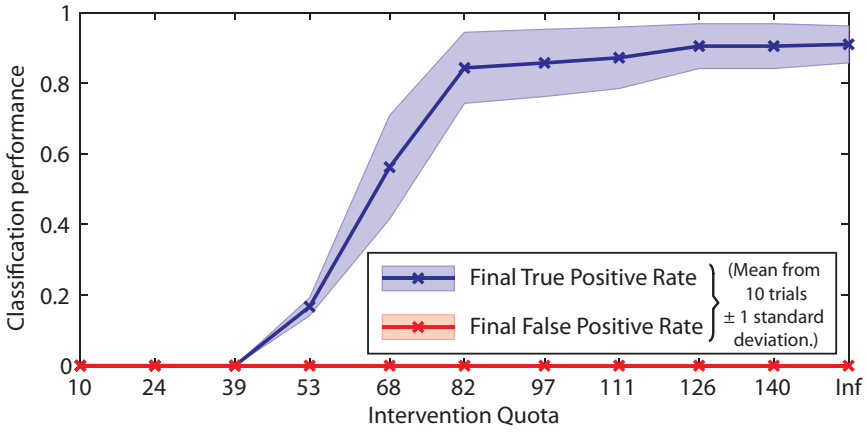
a further experiment was conducted where the maximum number of interventions available during a given training run was restricted. For each dataset, the proposed algorithm was trained using several different intervention "quotas" ranging (in uniformly spaced steps) from 10 interventions to (a number close to) the mean number of interventions identified in the preceding experiments. During training, once the maximum number of interventions is reached, further training examples which are classified as anomalous are discarded.

For each dataset, Figure 5.12 illustrates the classification performance obtained at the end of training using range of different intervention quotas. When only 10 user interventions are available, final classification performance is significantly reduced ($TP = 0/FP = 0$ for the CAVIAR dataset, $TP = 0.12 \pm 0.067/FP = 0$ for the Leeds dataset, and $TP = 0.44 \pm 0.095/FP = 0.001 \pm 0.001$ for the NGSIM dataset) compared to classifiers trained with unlimited interventions (see Table 5.3 for performance figures). As shown in Figure 5.12, classification performance increases proportionally to the number of available interventions: in each case, below a certain critical number of interventions, increases in the intervention quota make a large difference to classification performance; thereafter, increasing the intervention quota yields a relatively small increase in classification performance.

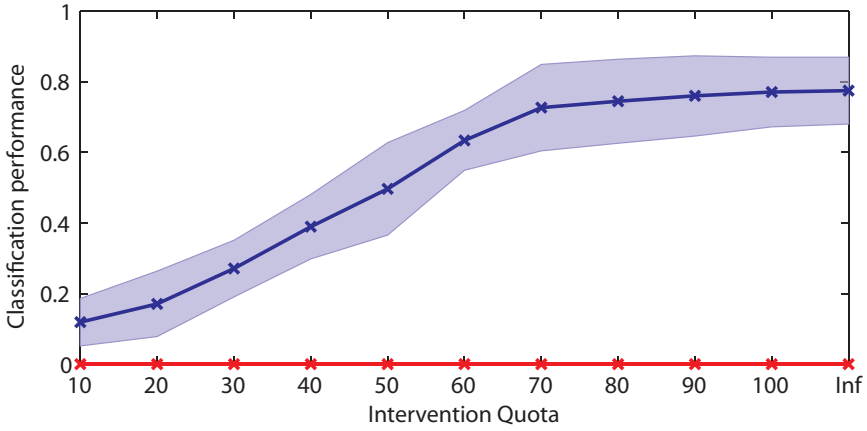
The results shown in Figure 5.12 serve to illustrate the role of user interventions in the proposed semi-supervised learning framework: it is clear that without the approval of certain key items of training data, the proposed framework successfully prevents the normal behaviour model from encompassing the corresponding "non-approved" regions of feature space. The relationship observed between classification performance and intervention quotas also serves to indicate that a reasonable level of performance can be achieved even if only a subset of intervention requests are addressed: a critical quantity of ~ 100 interventions seems to be necessary for the datasets considered here.

5.4.3 What difference does training on unlabelled examples make?

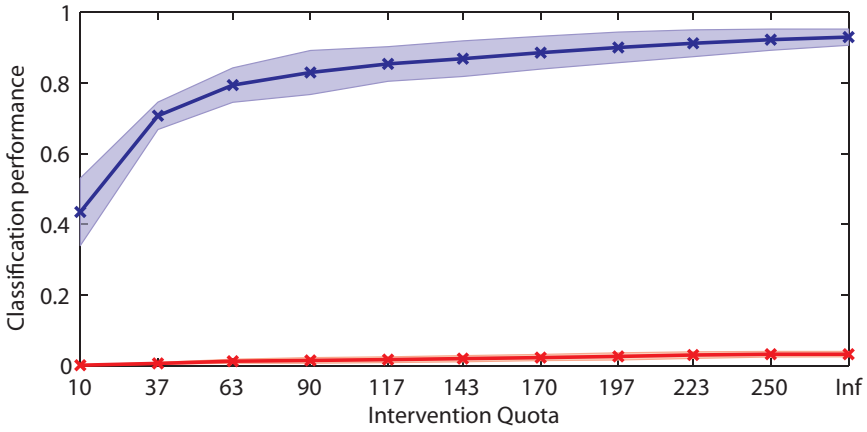
The preceding experiments have shown that the proposed framework allows a high level of classification performance to be achieved while only requiring labels for a minority of the training examples presented. A related issue, unresolved by these experiments, is determining the extent to which classification performance is improved by self-training on unlabelled examples (ie. those classified as normal). We examine the impact of unlabelled examples by modifying the experiment presented in Section 5.4.1 so that any unlabelled data classified as normal is discarded rather than being used for training. In order to ensure that there is no difference in the behaviour of



(a) CAVIAR "INRIA" dataset.



(b) Leeds carpark dataset.



(c) NGSIM "Peachtree Street" dataset.

Figure 5.12: Measuring the classification performance that is obtained when only a finite "quota" of user-interventions is made available. The final point on each plot corresponds to an "infinite", ie. unrestricted, intervention quota. (See text for discussion.)

the underlying learning algorithm in this experiment compared to that presented in Section 5.4.1, unlabelled data is only discarded after the first 100 examples have been presented: specifically, this ensures that the same kernel width (which is estimated from the first 100 training examples and affects learning/generalisation behaviour - see Chapter 3) is used in both experiments.

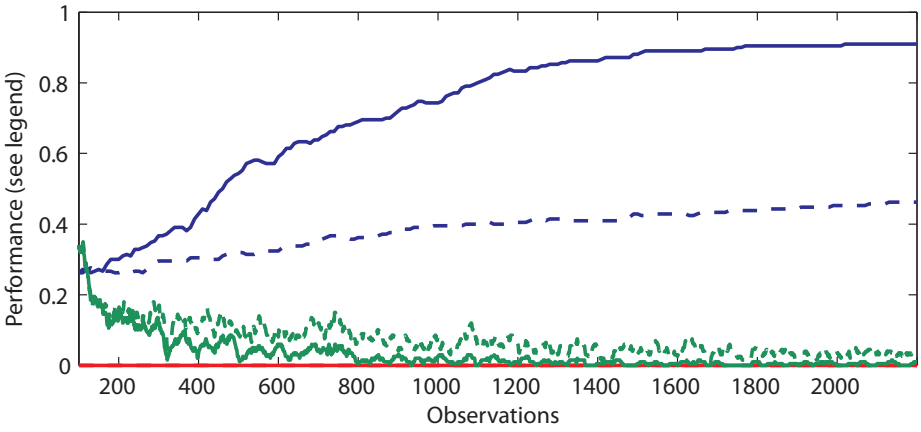
For each dataset, Figure 5.13 shows how classification performance changes as more observations are presented, both with and without the option of self-training on unlabelled data. For the CAVIAR dataset, removing the option to self-train on unlabelled data yields a large decrease in TPR (a mean value of 0.46 compared to 0.91), and a corresponding increase in the number of interventions (230.6 compared to 141.9). The same pattern - lower TPR and higher intervention rate in the absence of unlabelled data - can be observed for the NGSIM and Leeds datasets: however, in these cases, the difference is less extreme (see Table 5.5 for exact figures) than observed for the CAVIAR dataset. For the NGSIM dataset, training without unlabelled data causes a small reduction in FPR (a mean value 0.025 compared to 0.032), although this difference is smaller than the corresponding decrease in TPR. In this light it is also interesting to note that decreasing the quantity of *labelled* data, as explored using intervention quotas in the preceding section (see Figure 5.12), also causes FPR to decrease for the NGSIM dataset.

Overall, the option to self-train on unlabelled data appears to yield an improvement in classification performance and reduction in intervention rate for all datasets. While this effect is clearly observable for all datasets, its extent appears to depend on the dataset in question: the extreme difference observed for the CAVIAR dataset may be related to the discrepancy between synthetic/real train/test data (eg. it is possible that more unlabelled examples are required to ensure sufficient overlap between the training/test distributions in this case).

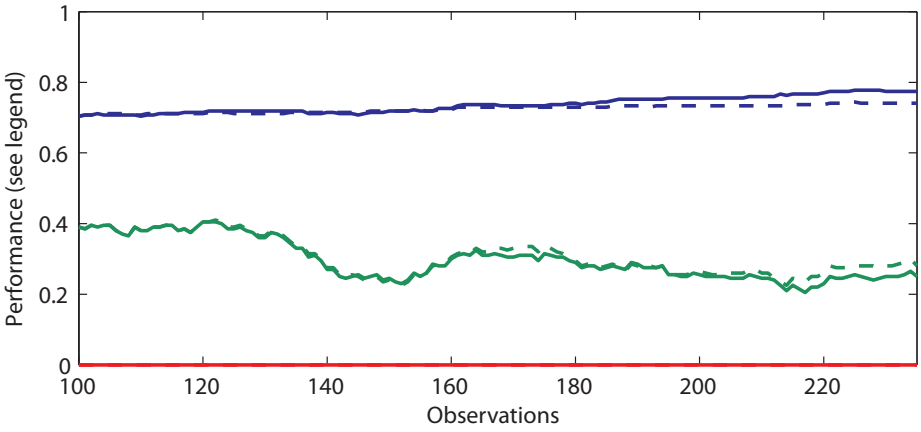
5.4.4 Will anomalous examples inevitably contaminate the model?

A key concern with proposed learning framework is that self-training may occur for incorrectly classified unlabelled training examples (ie. anomalous examples misclassified as normal), ultimately reducing the model's ability to distinguish between normal and anomalous examples. This concern is particularly pertinent in the case of the NGSIM dataset, where FPR appears to rise steadily as more training examples are observed; it is indeed possible that the same would occur for the Leeds and CAVIAR datasets given sufficient training examples.

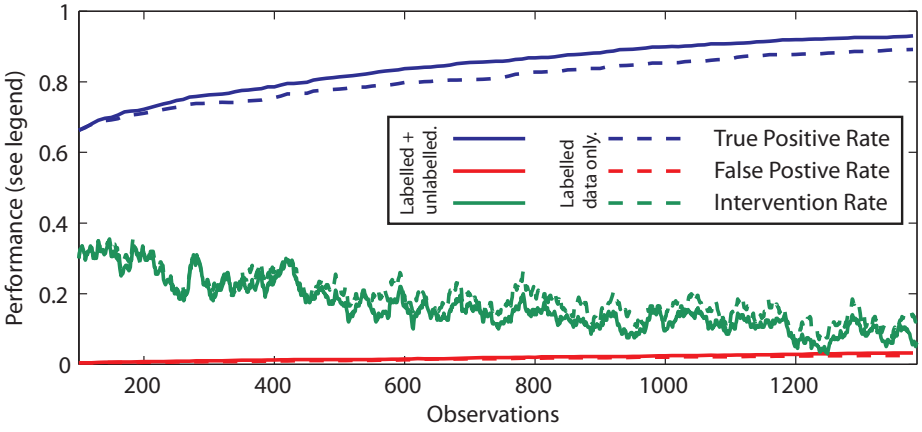
Here we conduct a further experiment to determine the extent to which classification performance might be damaged by training the model on unlabelled exam-



(a) CAVIAR "INRIA" dataset.



(b) Leeds carpark dataset.



(c) NGSIM "Peachtree Street" dataset.

Figure 5.13: The difference between the classification performance and intervention rates obtained when training with (solid lines) and without (dashed lines) the option to self-train on unlabelled data. (See text for discussion.)

	Data	Leeds	NGSIM	CAVIAR
TPR	L+U	0.77 ± 0.095	0.93 ± 0.023	0.91 ± 0.052
	L	0.74 ± 0.10	0.89 ± 0.023	0.46 ± 0.11
FPR	L+U	0	0.032 ± 0.0076	0
	L	0	0.025 ± 0.0097	0
Interventions	L+U	103.6 ± 8.7 (44.1%)	245.4 ± 37.5 (17.7%)	141.9 ± 21.1 (6.5%)
	L	140.1 ± 8.0 (59.6%)	283.4 ± 39.7 (20.5%)	230.6 ± 33.7 (10.5%)

Table 5.5: The difference between the classification performance obtained and total number of intervention incurred after training with (L+U) and without (L) the option to self-train on unlabelled data. (See text for discussion.)

ples containing a large proportion of anomalous examples. For each dataset, we take trained classifiers from Section 5.4.1 (trained with unlimited interventions), and present a series of further sequences of unlabelled training data. Each sequence consists of the entire set of anomalous test data and an identical quantity of randomly chosen (normal) training data, with noise (drawn from a Gaussian distribution with $\mu = \vec{0}$ and $\Sigma = 0.01 \cdot I$) added to each example (for the first presentation of the anomalous test set, no noise is added). We train the models on unlabelled data generated in this manner[3] until at least 10,000 examples have been presented.

During this additional training stage new data can only be incorporated into the normal behaviour model via self-training, ie. only if it is classified as normal: by presenting anomalous test data as unlabelled training data, we can determine whether or not occasional self-training on misclassified anomalous items leads to a catastrophic deterioration in classification performance. Figure 5.14 shows how classification performance changes for each dataset as the extra unlabelled data is presented (these plots can be regarded as a continuation of those shown in Figure 5.11), while Table 5.6 shows the TPR and FPR values obtained before and after the extra data is presented. For each dataset, training on the additional data causes a gradual increase in FPR. Although this reflects a deterioration in the ability to detect anomalous instances, the increase in FPR is matched by a corresponding increase in TPR for all datasets. Although a seemingly large increase in FPR ($0 \rightarrow 0.12$) is observed for the Leeds dataset, it is important to note that there are only 6 anomalous test examples for this dataset: in fact, the value of 0.12 corresponds to a single anomalous example being misclassi-

[3] An earlier experiment was conducted where unlabelled data was generated from a uniform distribution within a hypersphere enclosing the dataset (using the method proposed by Tax and Duin in [145]). However, perhaps due to the high dimensionality (15 dim) of the chosen feature space, all examples generated this way were classified as anomalous and thus could not be used for self-training.

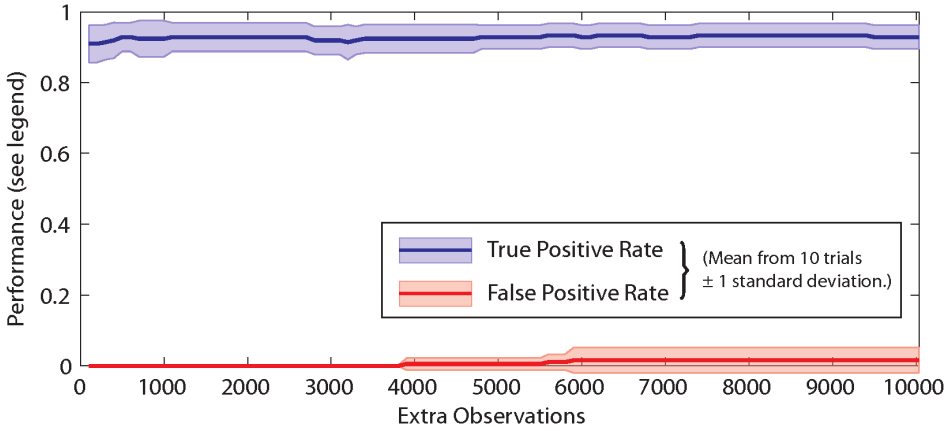
		Leeds	NGSIM	CAVIAR
TPR	Before	0.77 ± 0.095	0.93 ± 0.023	0.91 ± 0.052
	After	0.86 ± 0.096	0.95 ± 0.020	0.93 ± 0.034
FPR	Before	0	0.032 ± 0.0076	0
	After	0.12 ± 0.081	0.051 ± 0.016	0.016 ± 0.036

Table 5.6: Classification performance figures obtained before and after the presentation of 10,000 unlabelled examples, corresponding to the start and end points of the curves shown in Figure 5.13. (See text for discussion.)

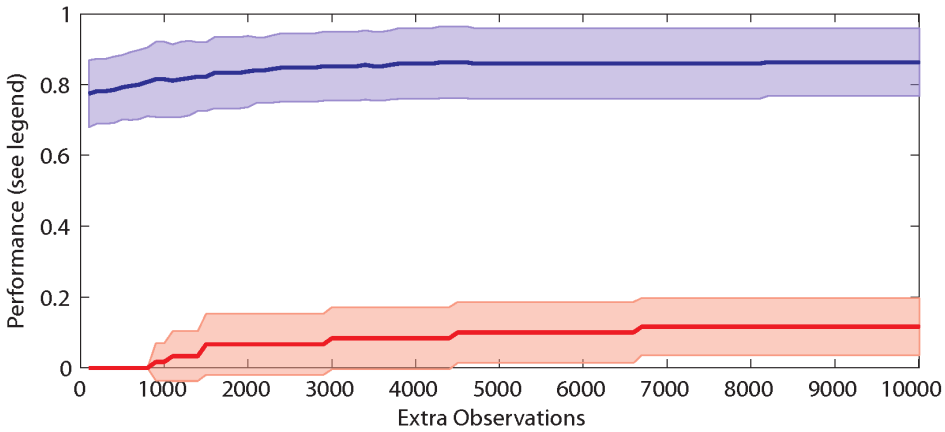
fied in $\frac{7}{10}$ trials. The present results do not conclusively rule out the possibility that a much smaller increase in FPR may have been measured for the Leeds dataset, if a larger selection of anomalous examples were available.

It is interesting to consider the underlying cause of the observed increase in FPR. For the NGSIM dataset, where the FPR was non-zero before the presentation of the additional instances, it is possible that the presentation of anomalous instances could have caused classification performance to deteriorate. For the CAVIAR and Leeds datasets, however, FPR was zero before the addition training data was presented: this means that the anomalous test data could not initially have been used for self-training, implying that self-training on "normal" instances caused the classification boundary to spread sufficiently to encompass some of the anomalous examples. It is difficult to determine whether this is a failing of the underlying learning algorithm, or whether it is an inevitable consequence of the imperfect separation of normal and anomalous instances in the chosen feature space (see Chapter 4). The very gradual nature of the increase in FPR (apparently reaching a plateau for the Leeds and CAVIAR datasets at around 7000 observations), and the fact that it is accompanied by a corresponding increase in TPR, lends some support to the latter hypothesis.

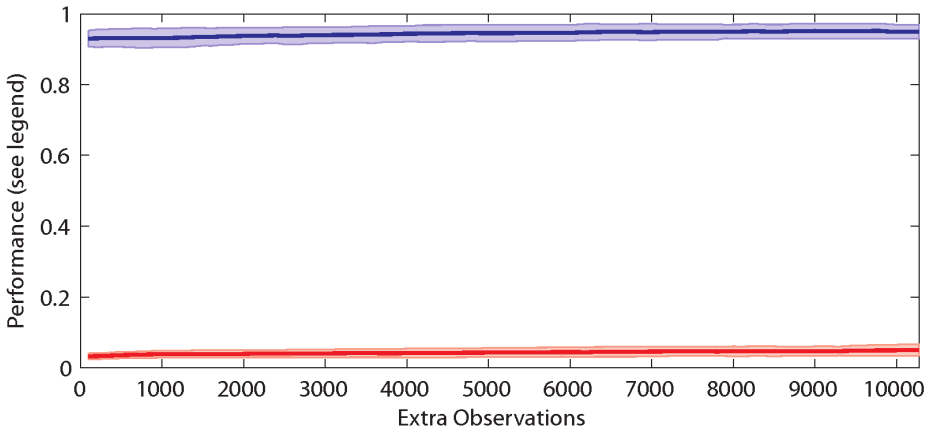
Despite the observed deterioration in classification performance, it is useful to consider the classification performance that would be afforded by an entirely unsupervised learning algorithm if trained on the unlabelled data used in the preceding experiment (ie. containing 50% normal and 50% anomalous instances). It is clear that modelling the distribution of this data would not allow anomalous instances to be detected on the basis of their low probability with respect to the model: in contrast, the proposed learning framework retains a reasonably high level of classification performance despite the large proportion of anomalous training data.



(a) CAVIAR "INRIA" dataset.



(b) Leeds carpark dataset.



(c) NGSIM "Peachtree Street" dataset.

Figure 5.14: Determining whether self training could damages the model: these plots measure classification performance as normal training examples and anomalous test examples - both with added noise - are presented as further unlabelled training data for pre-trained classifiers. (See text for discussion.)

5.5 Discussion

The results presented in this chapter have demonstrated that the learning algorithm proposed in Chapter 3 can be used in conjunction with a parametric trajectory representation (see Chapter 4) to yield high levels of classification performance for three different normal vs anomalous trajectory classification scenarios.

The central aim of this chapter was to test / demonstrate an incremental semi-supervised learning framework where a human operator is used to filter the training examples used to construct a normal behaviour model by providing labels for any new examples classified as anomalous. We have measured how the rate of requests for labels (ie. human interventions) would change during training on three different trajectory datasets: while in each case the intervention rate is seen to decrease steadily as more examples are observed, a usefully low (ie. $\leq 2\%$) intervention rate is only achieved for one case (the CAVIAR dataset). Due to the limited quantity of available training data, it is difficult to speculate about whether a similarly low intervention rate would be achieved for the other datasets given sufficient training examples. Nonetheless, we have also shown (in Section 5.4.2) that it is possible to reach a high level of classification performance if only a finite set of interventions are provided, although it should also be noted that performance is compromised significantly if the quota of available interventions is too small.

As discussed in Section 5.2.3, the proposed framework could be regarded as a combination of active learning and semi-supervised learning: what distinguishes the proposed approach from active learning is that in addition to selecting which examples need to be labelled, the remaining unlabelled examples are also used to improve the classifier via self-training. Section 5.4.3 examined the difference between classifiers trained with and without unlabelled data, and indicated that the ability to self-train on unlabelled data yielded an improvement in classification performance combined with reduction in intervention rate for all datasets. For the CAVIAR dataset, performance degraded significantly in the absence of unlabelled data while a much smaller decrease was observed for the Leeds/NGSIM datasets: in these cases it appears that labelled data makes the most significant contribution to classification performance.

Conversely, Section 5.4.4 attempted to determine whether self-training on unlabelled data could potentially damage the model: trained classifiers were presented with a sequence of 10,000 unlabelled examples (containing an equal mixture of normal and anomalous examples) yielding a small increase in FPR for all datasets, even those for which FPR was initially zero. This increase in FPR was accompanied by a corresponding increase in TPR which suggests either that the normal and anomalous classes may overlap in the chosen feature space, or that the underlying learning

algorithm yields an insufficiently parsimonious classification boundary and "over-generalises" to the normal behaviour class. Nonetheless, when taking into consideration the high proportion (50%) of anomalous examples in the unlabelled data presented, this experiment also provides a demonstration of a key advantage of the proposed learning framework: that it provides a "safety net" to prevent the inadvertent incorporation of anomalous instances. Given an entirely unsupervised learning framework (ie. where all examples are used for training and classification of new examples is based on their probability with respect to the model) the presentation of 10,000 examples containing 50% anomalous behaviour would cause a catastrophic degradation in classification performance.

The learning framework proposed in this chapter provides a potential strategy for using a human operator to filter training examples used to construct a normal-vs-anomalous behaviour classifier. Although the results obtained in this chapter go some way to suggesting that this idea could be effective, comprehensive testing on a larger set of - more extensive - real-world datasets would be necessary to provide a conclusive indication of its potential applicability in real-world scenarios. It is also worth noting that the semi-supervised learning framework we demonstrate in this chapter could be applied as a wrapper to any existing anomaly detection algorithm capable of incremental unsupervised learning: further experiments with different learning algorithms and different types of data may provide a useful indication of its wider applicability and efficacy.

Chapter 6

Conclusions

6.1 Main contributions and their limitations

This thesis has explored a semi-supervised learning framework for building scene-specific models of motion trajectory data that can be used to detect anomalous behaviour. The preceding chapters presented three main contributions:

1. A novel incremental one class learning algorithm which outperforms the best existing approach while only requiring an upper limit on model complexity to be specified.

2. A thorough empirical comparison of the relative class separability afforded by the four parametric trajectory representations that have previously been applied in the literature.

3. A semi-supervised learning framework for anomalous trajectory classification, where the training data used to construct a normal behaviour model is filtered to remove potential anomalies with minimal human labelling effort.

These contributions are summarised in the remainder of this section, together with a discussion of their limitations.

6.1.1 Incremental One-class Learning

Main findings The learning framework proposed in this thesis relies on the existence of a technique for incrementally - and parsimoniously - building a model of normal behaviour, so that anomalous instances can be detected. To address this issue, Chapter 3 proposed a technique for incrementally constructing a density estimate from a sequence of multivariate data - drawn from a single class - for the purpose of outlier

detection. The method worked by initially constructing a maximally-complex Gaussian mixture model where each new data point was represented by a Gaussian kernel function. Thereafter, once a maximum number of components had been added to the model, a pair of mixture components was merged for each new component added (specifically, the pair with lowest KL divergence pre/post merging).

The results presented in Chapter 3 indicated that mixture models generated using this approach - in conjunction with a density-based classification threshold - yielded useful levels of normal-vs-outlier classification performance for a range of datasets, throughout the incremental training process. In all of the experiments presented, the maximum number of mixture components was set to the same value (100). Despite this, the resulting classification performance was equivalent to that of classifiers generated using the Incremental SVDD algorithm [146, 140] where model complexity was determined using cross-validation [143]. Moreover, for low False Positive Rates (ie. below $\frac{1}{50}$), the proposed algorithm was shown to consistently yield higher True Positive rates than Incremental SVDD. Further results presented in Chapter 5 show that the algorithm also works effectively for anomaly detection on the basis of parametrised trajectory data.

Two different classification techniques were tested in Chapter 3: thresholding either 1) the density estimated by the mixture model (so that a user specified portion of training data is rejected) or 2) a component-wise measure of how likely a given deviation is according to the Gumbel distribution [109]. In most cases the density threshold provided the best results, but the component-wise threshold provided improved results on two datasets where it was hypothesised that generalisation due to component overlap was not beneficial. Interestingly, further results presented in Chapter 5 indicated that the component-wise classification boundary can also be useful when the range of density values calculated for the training data does not encompass the best classification threshold. Finally, examination of the role of the kernel function in the resulting mixture model presented in Chapter 3/Appendix A showed that the kernel covariance matrix acts as a bias/smoothing term added to all components in the model. Experiments showed that subtracting the kernel bias term reduces classification performance on many datasets, indicating that it acts to usefully compensate for the relatively low ratio of training examples to mixture components.

Limitations There are a number of potential criticisms that could be levelled at the proposed algorithm: while it has been shown to yield effective classification performance, it has involved a number of arbitrary - albeit consistent - choices. One such choice is the maximum number of mixture components: this has been set to 100 with

the justification that this is a reasonable upper limit on model complexity for most problems. While this choice yields reasonable results for a range of datasets, it would nonetheless be desirable to be able to adapt this quantity according to the dataset in question (eg. by adding an extra component if the current number is insufficient to represent a given dataset). Addressing this issue presents the challenge of evaluating the trade-off between data likelihood and model complexity (eg. using BIC or AIC [20]) without having the whole dataset to hand, while taking into account the need for an initially parsimonious/over-complex model when only a small quantity of data has been observed. Similarly, although the kernel covariance matrix is shown to have a considerable impact on classification performance, it is estimated in a one-off step from the first 100 training examples. Although this appears to be adequate for most datasets considered, it would be better to be able refine this estimate using further training examples (for example, it could be the case that the first 100 examples led to an inappropriately large bias, but there is currently no way this could be corrected).

It is interesting to note that two highlighted weaknesses correspond to aspects of the learning procedure that are arbitrarily prespecified, and that their potential solutions are at odds with a scenario where data arrives sequentially before being discarded. Since the method is designed to operate usefully from the point where the very first training examples are presented, it is impossible to escape the need to prespecify certain assumptions. In this light, perhaps the deepest criticism of the proposed method - its demonstrable classification performance notwithstanding - might be that the meaning/justification of its underlying assumptions is somewhat opaque.

6.1.2 Trajectory Representation

Main findings In order to apply the incremental one-class learning algorithm proposed in Chapter 3 to trajectory data, it was necessary to encode trajectories in fixed-length vector form. To avoid the need for a separate learning step to determine an appropriate representation, a "hard-coded" technique for representing trajectories was required. To address this issue, Chapter 4 presented an empirical comparison of four different representations (each corresponding to a different family of basis functions) where the class separability of several benchmark datasets was measured over a range of dimensionalities. The representations considered had all previously been adopted in the trajectory classification literature and included: the Haar wavelet transform, the Discrete Fourier Transform, Chebyshev polynomial approximation and least-squares cubic B-Spline regression.

Separability was measured using two main criteria: 1) The proportion of Relative-Neighbourhood-Graph edge weights corresponding to within-class edges [181], and

2) The leave-one-out 1-nearest-neighbour classification rate (corrected for differences in class size). These measures were applied to a range of datasets - ranging from pen trajectories to vehicle trajectories - which each contained a number of distinct classes. The results obtained indicate that although there was only a small difference in separability between the representations, the relative separability of the different representations had a persistent ordering in which the Haar representation generally yielded improved separability compared to the DFT representation, and the best separability was provided by either the Chebyshev or Spline representations. Furthermore, separability was found to increase with dimensionality before reaching a plateau at around 20 dimensions, although this trend was less clear for the Spline representation.

Chapter 4 also examined the impact of the underlying curve parametrisation strategy on the relative separability of each dataset/representation, exploring two different curve parameters: 1) Proportion of total time taken and 2) Proportion of total-arc length. Interestingly, curve parametrisation based on arc-length was found to yield the highest separability values for most datasets, implying that spatio-temporal information is largely irrelevant for the datasets considered (given their particular class labels). This difference was most noticeable for the NGSIM dataset, where spatial curve parametrisation yields a $\sim 10\%$ improvement in nearest-neighbour classification rate - explained by the fact that this dataset is divided into spatially defined classes, but contains a wide range of spatiotemporal profiles. However, a further experiment with simulated data confirmed - not unexpectedly - that temporal parametrisation is essential for distinguishing between trajectories which include pauses in different locations. This was confirmed by experiments presented in Chapter 5 which showed that normal/anomalous behaviour classification for the CAVIAR and Leeds trajectory datasets (each containing examples of anomalous behaviour provided by actors) was improved by temporal trajectory parametrisation.

These contradictory findings call into question the feasibility of a single fixed-length representation that captures both spatial and temporal characteristics: the ability to separate temporal characteristics appears to come at the expense of being able to separate spatial ones, despite the fact both characteristics are undoubtedly important for classifiying behaviour in a surveillance context. While reasonably high levels of classification performance are obtained using a spatiotemporal trajectory representation in Chapter 5, the results obtained for the NGSIM dataset hint that this strategy may not scale well to more complex datasets.

Limitations While the results obtained present a useful characterisation of the potential impact of different parametric trajectory representations on the separability of

different motion classes, there are a number of ways in which these results present an incomplete picture of the issues that were sought to be addressed.

The experiments presented measure the utility of each trajectory representation according to separability measures defined in terms of the Euclidean distance between raw coefficient vectors. While this gives a reasonable indication of the relative classification performance that might be afforded by density/neighbourhood based classifiers for different representations, it may not fully characterise the classification performance achievable in each case. For example, given a labelled dataset, it may be possible to learn a weighting for the constituent attributes of a given representation (eg. DFT coefficients) which would provide a relative level of classification performance that is at odds with the trends observed here. In other words, abstract measures of separability - defined independently of any given learning algorithm and dataset - may have limited scope as predictors of achievable classification performance.

In a different vein, it is also important to note that there are many other potential trajectory representations that could have been explored - the selection examined in this thesis merely reflects those that have been adopted in the trajectory classification literature. There are, however, many other possible representations, including a vast array of different families of wavelet functions (eg. Daubechies wavelets, and different orders therein), and different orders of B-Spline functions (eg. linear, quadratic, etc.), among others. Similarly, given the separability framework adopted here, it would also be possible to the compare the class separability afforded by a range of non-parametric trajectory distance measures (eg. Hausdorff distance, Dynamic Time Warping cost, etc.) alongside that of parametric representations.

Finally, although a diverse range of datasets has been used to compare trajectory representations, it is not at all clear that they adequately characterise the type of discrimination task that would need to be addressed when distinguishing between normal and anomalous trajectories in real world data. While the CAVIAR and Leeds datasets (used to test the proposed learning framework in Chapter 5) provide some examples of such data, the number of available examples is insufficient to meaningfully characterise the adequacy of different representations.

6.1.3 Semi-supervised Normal-behaviour Modelling

Main findings The central focus of this thesis was to explore the possibility of a semi-supervised learning framework for incrementally constructing a model of normal behaviour, where unlabelled examples are combined with occasional human labelled examples. The proposed framework used the classification output of an incremental one-class learning algorithm to determine whether to request labels for a sequence

of unlabelled trajectories: labels were requested for trajectories classified as anomalous, while those classified as normal were automatically used as training data. The rationale of this framework is that it allows a human operator to filter a sequence of unlabelled dataset - to ensure that it only contains normal examples - by providing labels for a small set of "unprecedented" examples. Using the one-class learning algorithm proposed in Chapter 3 in conjunction with a 15-dimensional cubic spline representation, classification performance and intervention rate were measured during simulated training runs on three different real trajectory datasets.

The results obtained for each dataset followed a similar pattern of behaviour: the proportion of correctly recognised normal behaviour (True Positive rate) initially rose quickly before continuing to increase more slowly (reaching a level of $\sim 90\%$ for the CAVIAR and NGSIM datasets and a lower level of $\sim 80\%$ for the Leeds dataset, where fewer training examples were available). In all cases the proportion of incorrectly classified anomalous test data (False Positive rate) remained low (never exceeding zero for the Leeds and CAVIAR datasets, while gradually increasing to a low level of 3.2% for the NGSIM dataset). In all cases, the rate of intervention requests initially fell rapidly before continuing to decrease at a slower rate: for the CAVIAR dataset, where the largest number of training examples were available, this reached a usefully low level of $\sim 1\%$ at the end of training.

A further experiment explored the impact of restricting the total number of user interventions, indicating that once a critical number of interventions - around 100 - had been provided, a high level of classification performance (comparable to that obtained with unlimited interventions) could be achieved even when all subsequent intervention requests were ignored. For each dataset, even when unlimited interventions are available, the total number of requested interventions corresponded to a minority of the total training data presented, with automatically incorporated unlabelled data constituting the remainder. To determine the impact of unlabelled data, training was repeated without the ability to automatically incorporate unlabelled data: in each case, the absence of unlabelled data caused a decrease in True Positive rate and an increase in the intervention rate. Interestingly, this magnitude of this change was very large for the CAVIAR dataset (where training/test data distributions may differ), but was only marginal for the other datasets considered.

Finally, we attempted to determine the extent to which the classification performance afforded by the proposed framework might deteriorate given training data that contained a large proportion of anomalous activity. When pre-trained models were presented with a further sequence of 10,000 unlabelled examples comprising 50% normal training data and 50% anomalous test data (all with added noise), only

a small increase in False Positive rate was observed (typically reaching a plateau for the last 3-4000 presented examples). This finding demonstrated the advantage of the proposed framework - as a means to filter/selectively model unlabelled data - over an entirely unsupervised learning algorithm, since any reasonable model for the distribution of the 10,000 unlabelled training examples presented would provide little means to discriminate normal/anomalous activity.

Limitations While the results presented in Chapter 5 provide some indication of the potential efficacy of the proposed learning framework they leave a number of questions unresolved. One key issue is whether the rate of requests for labels is - in general - likely to decrease to a useful level. The one dataset for which this was shown to be the case (the CAVIAR dataset) consisted of simulated training data (as only a small quantity of real test data was available) whose distribution may have been misleadingly simple compared to the equivalent real trajectory data that would have occurred in the same scenario. The only way in which this question could be addressed - aside from developing an improved algorithm that yields lower intervention rates on the available datasets - would be to test the algorithm on larger sets of real trajectory data.

A second issue which has not been conclusively addressed is the potential for classification performance to deteriorate if/when inappropriate unlabelled examples are automatically incorporated as normal training data. While the results obtained only showed a small increase in False Positive rate when a large set of normal and anomalous unlabelled examples were presented, the results do not rule out the possibility for further increases. Although - as discussed earlier - the proposed framework can only improve on its equivalent unsupervised counterpart, the possibility that a declining ability to correctly filter unlabelled data may eventually negate the human labelling effort expended calls into question the logic of the proposed framework. Another potential explanation for the observed increase in FPR lies in the potential inadequacy of the fixed-length vector trajectory representation used (as discussed earlier): in this light a more conclusive demonstration of the proposed framework may require a more accurate picture of the adequacy of the underlying representation for distinguishing normal/anomalous behaviour.

Viewed from a pragmatic perspective, the proposed trajectory modelling framework has a number of shortcomings that would need to be addressed before it could be genuinely useful. If the system is to detect behaviour in real-time, it would be necessary to evaluate trajectories as they unfold, rather than after a given person/vehicle has left the scene: however, as currently implemented, the proposed framework provides no means to accomplish this. In a similar vein, a key issue for the potential

applicability of the proposed framework is its ability to integrate with the output of an automatic tracking algorithm. Since current tracking algorithms (see eg. [174]) are by no means perfect, the trajectory representation strategy and learning algorithm would need to be adapted to take into account the regular occurrence of partial/incomplete tracks. Furthermore, even if perfect tracking were available, the current learning/classification framework does not take into account the reconstruction error implicit in the parametric representation of each trajectory: in reality however, two similar parameter vectors with very different reconstruction errors are unlikely to represent similar behaviours and should not be treated as doing so. Indeed, quantifying the reconstruction error associated with each parametrised trajectory may provide a means to alleviate the seeming overlap between normal and anomalous trajectories in the chosen feature space.

In a different vein, another shortcoming of the proposed learning framework is that is assumes a human operator to be an infallible source of accurate labels. In reality, trajectories could potentially be mislabelled but the current procedure provides no scope to address this issue. It could also be argued that the proposed framework is wasteful, as labelled anomalous examples are discarded: although anomalous behaviours are unlikely to follow any definitive template, incorporating a model for observed anomalies into the proposed framework may serve to improve classification performance (for example by allowing the classification threshold parameter to determined non-arbitrarily).

6.2 Future work

As indicated in the preceding discussion, there are a variety of ways in which the work presented in this thesis could be further developed. Several potential extensions are briefly described as follows.

Classification of unfolding trajectories given current model. The current trajectory modelling / classification framework requires a trajectory to be completed before it can be classified. In a practical setting, it would be important to be able to identify anomalous trajectories as they unfold. It would thus be useful to explore methods for classifying unfolding trajectories, without fundamentally altering the current (whole-trajectory based) model. Given the current model, the key problem would be mapping partially formed trajectories to the parameter space (eg. spline control points) used to represent whole trajectories. One straightforward practical solution to this problem would involve storing a mean trajectory for each mixture component, so that a non-

parametric distance measure (eg. the Hausdorff distance) could be used to match each partial trajectory to its closest mean trajectory: the partial trajectory could then be blended with its closest mean trajectory to generate a hypothetical whole trajectory, which can be parametrised/classified according to current framework.

Dealing with automatic tracking output. Noting that current tracking algorithms are unlikely to consistently generate whole / unbroken motion trajectories, it may not be realistic to construct a model of whole motion trajectories as envisaged in this thesis. It is possible that the current framework could be used to model and classify the *trajectory segments* resulting from automatic tracking, rather than whole trajectories. Nonetheless, one potential problem with directly attempting to model tracking output is that a prohibitively large number of mixture components may be needed to adequately represent the distribution of trajectory segments, whose range of lengths and start/end points will necessarily be much wider than that of whole trajectories.

However, while it would be important to be able to classify all tracking output, it is not necessary to use it all for training. In this light, provided that it is possible to classify partial segments of any length, the set of trajectory segments used for training could be restricted to those of a certain minimum length. The minimum segment length would effectively specify a compromise between the required model complexity (a shorter minimum segment length would require more mixture components) and the number of available data (a greater minimum length would reduce the number of available training examples). Ultimately, however, it is likely that the most appropriate representational strategy will depend strongly on the characteristics of the chosen tracking algorithm and of the particular scenario in question.

Additionally, it is worth noting that focussing solely on trajectory segments would relinquish the ability to capture and classify the global aspects of a given behaviour: for example, a globally unusual trajectory may be comprised of a collection of intrinsically normal segments. It is clear that the detection of such behaviours hinges on the development of improved tracking algorithms, or techniques for associating track segments corresponding to a given identity.

Modelling spatial/temporal characteristics separately. The results presented in Chapter 4 indicated using a single set of parameters to represent the spatial and spatiotemporal characteristics of trajectories may compromise the separation of spatially-defined trajectory classes in the resulting feature space. While this could be alleviated by using a purely spatial representation, spatiotemporal characteristics are likely to be important for classifying surveillance data: a potential solution to this problem is to model

the spatial and temporal characteristics of trajectories separately.

In practical terms this would involve constructing two separate mixture models. One of these would encode the distribution of spatially-parametrised trajectories. A second model would encode the distribution of 1-dimensional distance-vs-time curves. It is clear the spatial and temporal characteristics of trajectories are not necessarily independent (eg. different temporal patterns may be expected on a road vs a pavement). This issue could be resolved by maintaining a matrix representing the correspondence between components in the spatial mixture model with those in the temporal mixture model. Initially - before any component merging has taken place - this would be an identity matrix encoding a one-to-one correspondence. The particular pairings of spatial and temporal mixture components that are updated when each new example arrives would then be used to incrementally update this matrix, thereby defining a separate prior distribution over temporal mixture components for each spatial mixture component.

New trajectories would be classified with respect to the spatial mixture model, and then with respect to a re-weighted version of the temporal mixture model corresponding to the closest spatial component: a given trajectory would be classified as anomalous if found to be anomalous with respect to either model. Indeed, this approach could be extended to a range of other features (eg. pertaining to object appearance or articulated motion).

Anomalous behaviour modelling. As noted in the preceding section, the learning framework proposed in this thesis discards any trajectories that are deemed anomalous by a human operator. However, the proposed framework could be extended to incorporate anomalous examples in various different ways. One straight-forward possibility would be to construct a second mixture model representing anomalous examples, in parallel to the one used to model normal examples. This model should be separate for two reasons: 1) to prevent pairs of normal/anomalous components being merged and 2) so that a large kernel width can be used in order to counteract the sparsity of anomalous training examples. Furthermore, noting the wide range of possible anomalous behaviours, the complexity (ie. number of mixture components) of the anomalous behaviour model may need to be far greater than that of the normal model. The ability to add new components to the mixture model, and determine when it is necessary to do so, would be important in this case (see next item of future work).

Given a pair of mixture models representing both normal and anomalous behaviour, new anomalous examples could be detected using a Likelihood Ratio Test (as adopted by Xiang and Gong for this purpose in [166]) as follows, where M_n and M_a denote the

mixture models (both constructed as described in Chapter 3) representing normal and anomalous behaviour, and τ denotes a pre-specified threshold:

$$R_L = \frac{P(x|M_n)}{P(x|M_a)} \begin{cases} \geq \tau \rightarrow \text{accept } \mathrm{H}_{\text{normal}} \\ < \tau \rightarrow \text{reject } \mathrm{H}_{\text{normal}} \end{cases} \tag{6.1}$$

A key problem is choosing an appropriate threshold τ. If the prior probabilities $P(a)$ and $P(n)$ were known this ratio could be formulated in terms of posterior probabilities to give a decision boundary at $\frac{P(x|M_n)P(n)}{P(x|M_a)P(a)} = 1$, which leads to $\tau = \frac{P(a)}{P(n)}$. Given that $P(a)$ may be very low this approach could inadvertently discount potentially anomalous behaviour: incorporating a loss matrix (see eg. [12]) encoding the relative risks R_n, R_a of misclassifying normal and anomalous behaviour then leads to $\tau = \frac{P(a)R_a}{P(n)R_n}$. However, without any meaningful way to specify R_n and R_a this approach still ultimately equates to specifying an arbitrary τ. In the absence of any prior knowledge a simple choice, which assumes the risks of misclassification are proportional to the rarity of each class, would be $\tau = 1$.

Even given an appropriate value for τ that correctly classifies the types of examples observed thus far, it is important to take into account the possibility that a future - unprecedented - example of anomalous behaviour may have a parameter vector that has a higher likelihood with respect to the normal behaviour model. The only way to identify such instances, therefore, is to retain the existing anomaly detection rules adopted in this thesis and use these in conjunction with the preceding two class approach. An intuitive strategy would be to employ a winner-takes-all rule, so that a new example is deemed anomalous if either classification approach suggests this.

It is interesting to consider the role that the anomalous behaviour model would play. If the two class classification boundary does not overlap with the one-class boundary describing normal behaviour, then the anomaly model does not provide any direct benefit for classification. If the boundaries did overlap, however, then the WTA rule would constrict the resulting classification boundary in regions closer to the observed anomalies: this could be achieved by drawing samples from the normal behaviour model, and choosing a value of τ that causes the two-class decision boundary to overlap with the one-class boundary so that a certain proportion of the normal samples are misclassified.

In addition to directly refining the classification boundary, another important role for the anomalous behaviour model would be to prevent the classification boundary from growing inappropriately during self training. This could be achieved by measuring the change in the overlap between components of the normal/anomalous behaviour models each time the normal behaviour model is updated: if the overlap

between the updated component and any component from the anomalous model (this could be calculated using $\max_i \left(\int G_{N*}(x) G_{A(i)}(x) dx \right)$ where G_{N*} is the updated normal component and $G_{A(i)}$ is the ith anomalous component) increases beyond a threshold then the self-training update should be discarded.

Noting the potential scarcity of anomalous examples, and the fact that an arbitrary degree of overlap between the one- and two-class boundaries can be achieved, the principal advantage of having an anomaly model would - in many cases - lie in preventing inappropriate expansion of the classification boundary due to self-training.

Online model complexity adjustment. As discussed earlier, a key criticism of the learning algorithm proposed in Chapter 3 is that does not provide a means to increase the maximum number of mixture components to cope with new data. Currently the addition of a new component requires an existing pair to be merged, and only occurs if the relative cost (based on KL divergence - see Chapter 3) of the latter is lower than that of updating an existing component with the latest training example. However, even if merging a pair of existing components incurs the lowest *relative cost* it may lead to undesirable over-generalisation.

In such circumstances, it would be desirable to assess the cost of merging a pair of existing components in *absolute* rather than relative terms, so that if the cost were deemed "too high" no merging would occur and the model complexity would be increased by one component. Given the set of training examples for which a given pair of components is responsible, it would be possible to evaluate a complexity-penalised likelihood criterion (eg. BIC or AIC [20, 12]) for 1) the unmerged components and 2) their merged counterpart: if this criterion favours the unmerged components, then the proposed merge should not take place. In the proposed framework, however, no examples are stored: to address this issue the *expected likelihood* given the number of examples represented by the two components could be calculated. This quantity is straightforward to calculate, and has previously been applied to resolve split-vs-merge decisions for Gaussian mixture models by Arandjelovic and Cipolla in [5].

In a similar vein, this approach could also be used to provide an absolute criterion for choosing whether and how to reduce the number of components in the model. Given that an over-fitted model is preferable to an underfitted one in the case of outlier detection, the AIC criterion (which favours more complex models [12]) is likely to be preferable to BIC: future experiments could explore how these strategies would affect classification performance. Nonetheless, the intuitive strategy adopted in this thesis - to initially fit an overcomplex model to unknown data - seems a sensible starting point for any subsequent model complexity adjustment.

Appendix A

Derivations

In this chapter we provide a derivation of the formulae for merging two mixture components used in Chapter 3, and use this to show how the kernel covariance matrix (which accompanies each new example added to the model) manifests itself in the parameters of the resulting mixture model. Finally, we provide a brief theoretical justification for the merging cost function adopted in Chapter 3.

A.1 Merging formulae for two Gaussian mixture components.

The merging formulae stated in Chapter 3 can be derived in a straightforward fashion by hypothesising two separate sets of data $X = \{x_1,\ldots,x_P\}$ and $Y = \{y_1,\ldots,y_Q\}$ represented (using maximum-likelihood parameter estimates) by Gaussian distributions $C_X = \{\mu_X, S_X\}$ and $C_Y = \{\mu_Y, S_Y\}$, and showing that the parameters of a Gaussian distribution $C_{XY} = \{\mu_{XY}, S_{XY}\}$ representing a combination of both sets of data $X \cup Y = \{x_1,\ldots,x_P,y_1,\ldots,y_Q\}$ can be expressed in terms of the parameters (ie. sufficient statistics) of C_X and C_Y.

Maximum likelihood parameter estimates for C_X and C_Y

The maximum likelihood estimates for the parameters of C_X and C_Y are expressed in terms of $\{x_1,\ldots,x_P\}$ and $\{y_1,\ldots,y_Q\}$ as follows:

$$C_X : \left\{ \mu_X = \tfrac{1}{P}\textstyle\sum_{p=1}^{P} x_p, \quad S_X = \sum_{p=1}^{P}(x_p - \mu_X)(x_p - \mu_X)^T \right\}$$
$$C_Y : \left\{ \mu_Y = \tfrac{1}{Q}\textstyle\sum_{q=1}^{Q} y_q, \quad S_Y = \sum_{q=1}^{Q}(y_q - \mu_Y)(y_q - \mu_Y)^T \right\}$$

If C_X and C_Y are components of a mixture model representing a total of N examples, their corresponding prior probabilities π_X and π_Y are given by:

$$\pi_X = \frac{P}{N}, \quad \pi_Y = \frac{Q}{N}$$

Parameters of merged component C_{XY} in terms of C_X and C_Y

The prior probability π_{XY} for the merged component is given by:

$$\pi_{XY} = \frac{P+Q}{N} = \pi_X + \pi_Y$$

The mean μ_{XY} of the combined set of examples $X \cup Y$ can then be expressed in terms of $\mu_X, \mu_Y, \pi_X, \pi_Y, \pi_{XY}$ as follows:

$$\begin{aligned}
\mu_{XY} &= \frac{1}{P+Q}\left(\sum_{p=1}^{P} x_p + \sum_{q=1}^{Q} y_q\right) \\
&= \frac{P}{P+Q}.\left(\frac{1}{P}\sum_{p=1}^{P} x_p\right) + \frac{Q}{P+Q}.\left(\frac{1}{Q}\sum_{q=1}^{Q} y_q\right) \\
&= \frac{P}{P+Q}.\mu_X + \frac{Q}{P+Q}.\mu_Y \\
&= \frac{\pi_X}{\pi_{XY}}.\mu_X + \frac{\pi_Y}{\pi_{XY}}.\mu_Y
\end{aligned}$$

Finally, the covariance matrix S_{XY} for the combined set of examples $X \cup Y$ (now calculated with respect to the combined mean μ_{XY}) can be expressed in terms of $S_X, S_Y, \mu_X, \mu_Y, \mu_{XY}, \pi_X, \pi_Y, \pi_{XY}$ as follows:

$$\begin{aligned}
S_{XY} &= \frac{1}{P+Q}\left[\sum_{p=1}^{P}(x_p-\mu_{XY})(x_p-\mu_{XY})^T + \sum_{q=1}^{Q}(y_q-\mu_{XY})(y_q-\mu_{XY})^T\right] \\
&= \frac{P}{P+Q}.\left[\frac{1}{P}\sum_{p=1}^{P}(x_p-\mu_{XY})(x_p-\mu_{XY})^T\right] + \frac{Q}{P+Q}.\left[\frac{1}{Q}\sum_{q=1}^{Q}(y_q-\mu_{XY})(y_q-\mu_{XY})^T\right] \\
&= \frac{P}{P+Q}.\left[\left(\frac{1}{P}\sum_{p=1}^{P}x_p x_p^T\right) - \mu_{XY}\mu_{XY}^T\right] + \frac{Q}{P+Q}.\left[\left(\frac{1}{Q}\sum_{q=1}^{Q}y_q y_q\right) - \mu_{XY}\mu_{XY}^T\right] \\
&= \frac{P}{P+Q}.\left[\left(\frac{1}{P}\sum_{p=1}^{P}x_p x_p^T\right) - \mu_X\mu_X^T + \mu_X\mu_X^T - \mu_{XY}\mu_{XY}^T\right] \ldots \\
&\quad + \frac{Q}{P+Q}.\left[\left(\frac{1}{Q}\sum_{q=1}^{Q}y_q y_q\right) - \mu_Y\mu_Y^T + \mu_Y\mu_Y^T - \mu_{XY}\mu_{XY}^T\right] \\
&= \frac{P}{P+Q}.\left[S_X + \mu_X\mu_X^T - \mu_{XY}\mu_{XY}^T\right] + \frac{Q}{P+Q}.\left[S_Y + \mu_Y\mu_Y^T - \mu_{XY}\mu_{XY}^T\right] \\
&= \frac{\pi_X}{\pi_{XY}}.\left[S_X + \mu_X\mu_X^T - \mu_{XY}\mu_{XY}^T\right] + \frac{\pi_Y}{\pi_{XY}}.\left[S_Y + \mu_Y\mu_Y^T - \mu_{XY}\mu_{XY}^T\right] \\
&= \frac{\pi_X}{\pi_{XY}}.\left[S_X + \mu_X\mu_X^T\right] + \frac{\pi_Y}{\pi_{XY}}.\left[S_Y + \mu_Y\mu_Y^T\right] - \mu_{XY}\mu_{XY}^T
\end{aligned}$$

A.2 Contribution of the kernel function to the final mixture model.

Given the merging rule derived in the preceding section, we now analyse the impact of the kernel covariance matrix in the Gaussian mixture model that results from the learning algorithm proposed in Chapter 3. Specifically we show that once a given mixture component represents multiple examples $\{z_1,\ldots z_n\}$ (ie. it is no longer just a kernel function associated with a single data point), its covariance matrix S can be expressed as

$$S = Cov(\{z_1,\ldots z_n\}) + K \tag{A.1}$$

where $Cov(\{z_1,\ldots z_n\})$ is the maximum likelihood estimate for the covariance of $\{z_1,\ldots z_n\}$, and K is the kernel covariance matrix.

Scenario 1: Merging two Gaussian kernels, each representing one example.

As discussed earlier, the mixture model initially consists of a set of Gaussian kernel functions. The parameters of a pair of Gaussian kernel functions $C_1$and C_2 representing single instances $x_1$and x_2 within a mixture representing a total of N examples are defined as follows (where K is the kernel covariance matrix):

$$C_1 : \{\pi_1 = \tfrac{1}{N},\ \mu_1 = x_1,\ S_1 = K\}$$
$$C_2 : \{\pi_2 = \tfrac{1}{N},\ \mu_2 = x_2,\ S_2 = K\}$$

Applying the merging formulae from the previous section gives mean $\mu_{1,2} = \frac{1}{2}x_1 + \frac{1}{2}x_2$ and prior probability $\pi_{1,2} = \frac{2}{N}$ for the merged component $C_{1,2}$. The covariance matrix $S_{1,2}$ can then be expressed in terms of K, x_1, x_2 as follows:

$$\begin{aligned} S_{1,2} &= \frac{\pi_1}{\pi_{1,2}}.\left[S_1 + \mu_1\mu_2^T\right] + \frac{\pi_1}{\pi_{1,2}}.\left[S_2 + \mu_2\mu_2^T\right] - \mu_{1,2}\mu_{1,2}^T \\ &= \frac{\pi_1}{\pi_{1,2}}.\left[K + x_1x_1^T\right] + \frac{\pi_1}{\pi_{1,2}}.\left[K + x_2x_2^T\right] - \mu_{1,2}\mu_{1,2}^T \\ &= \frac{1}{2}(x_1x_1^T + x_2x_2^T) - \mu_{1,2}\mu_{1,2}^T + K \\ &= Cov(\{x_1, x_2\}) + K \end{aligned}$$

Scenario 2: Merging a Gaussian kernel representing one example with a Gaussian mixture component representing multiple examples.

In a similar vein, we now show that the covariance matrix $S_{X,y}$ resulting from merging a Gaussian component C_X representing a set of $P \geq 2$ examples $X = \{x_1,\ldots x_P\}$ with a

Gaussian kernel function C_y representing a single example y can still be expressed in the form of Equation A.1:

$$\begin{aligned}
S_{X,y} &= \frac{\pi_X}{\pi_{X,y}}.\left[S_X + \mu_X\mu_X^T\right] + \frac{\pi_y}{\pi_{X,y}}.\left[S_y + \mu_y\mu_y^T\right] - \mu_{X,y}\mu_{X,y}^T \\
&= \frac{P}{P+1}.\left[Cov(x_1,\ldots,x_P) + K + \mu_X\mu_X^T\right] + \frac{1}{P+1}.\left[K + yy^T\right] - \mu_{XY}\mu_{XY}^T \\
&= \frac{P}{P+1}.\left[\left(\frac{1}{P}\sum_{p=1}^{P} x_p x_p^T\right) - \mu_X\mu_X^T + K + \mu_X\mu_X^T\right] + \frac{1}{P+1}.\left[K + yy^T\right] - \mu_{X,y}\mu_{X,y}^T \\
&= \frac{1}{P+1}\left(\sum_{p=1}^{P} x_p x_p^T\right) + \frac{1}{P+1}.yy^T - \mu_{X,y}\mu_{X,y}^T + K. \\
&= Cov(\{x_1,\ldots x_P, y\}) + K
\end{aligned}$$

Scenario 3: Merging two Gaussian components, each representing multiple examples.

Finally, following an identical logic, the covariance matrix S_{XY} resulting from merging two components C_X and C_Y representing sets of *multiple* examples $\{x_1,\ldots,x_P\}$ and $\{y_1,\ldots y_Q\}$ can also be expressed in the preceding fashion:

$$\begin{aligned}
S_{XY} &= \frac{\pi_X}{\pi_{XY}}.\left[S_X + \mu_X\mu_X^T\right] + \frac{\pi_Y}{\pi_{XY}}.\left[S_Y + \mu_Y\mu_Y^T\right] - \mu_{XY}\mu_{XY}^T \\
&= \frac{P}{P+Q}.\left[Cov(x_1,\ldots,x_P) + K + \mu_X\mu_X^T\right]\ldots \\
&\quad + \frac{Q}{P+Q}.\left[Cov(y_1,\ldots,y_Q) + K + \mu_Y\mu_Y^T\right] - \mu_{XY}\mu_{XY}^T \\
&= \frac{P}{P+Q}.\left[\left(\frac{1}{P}\sum_{p=1}^{P} x_p x_p^T\right) - \mu_X\mu_X^T + K + \mu_X\mu_X^T\right]\ldots \\
&\quad + \frac{Q}{P+Q}.\left[\left(\frac{1}{Q}\sum_{q=1}^{Q} y_q y_q^T\right) - \mu_Y\mu_Y^T + K + \mu_Y\mu_Y^T\right] - \mu_{XY}\mu_{XY}^T \\
&= \frac{1}{P+Q}\left(\sum_{p=1}^{P} x_p x_p^T + \sum_{q=1}^{Q} y_q y_q^T\right) - \mu_{XY}\mu_{XY}^T + K. \\
&= Cov(\{x_1,\ldots,x_P,y_1,\ldots y_Q\}) + K
\end{aligned}$$

It is clear that any merging operation that could occur in the proposed learning algorithm will correspond to one of the preceding scenarios, provided that the kernel covariance matrix K remains constant once merging has commenced.

A.3 Justification of Merging Cost Function

When merging a pair of mixture components it is desirable to minimise any decrease in the likelihood of the underlying training data. However, in the proposed incremental learning scenario, it is assumed that historical training data is not retained and so cannot be evaluated.

One way to address this problem is to compare the *expected* log-likelihood of a point drawn from the original pair of components (ie. a two component GMM composed of those components) when calculated with respect to 1) the original pair of components and 2) the proposed merged component. The expected resulting decrease in log-likelihood is captured by the KL divergence $\mathcal{KL}(P||Q) = \int p(x) \log \frac{p(x)}{q(x)} dx = \int p(x) \left(\log p(x) - \log q(x)\right) dx$, which for the proposed scenario corresponds to:

$$\mathcal{KL}((w_i G_i + w_j G_j)||G_{merge(i,j)}) \tag{A.2}$$

Unfortunately there is not a closed form solution for this measure, leaving Monte Carlo sampling as the only means of evaluation. In the proposed learning algorithm, the pair of mixture components merged at a given step is chosen by minimising the following alternative measure [47] which *can* be evaluated in closed form:

$$w_i \mathcal{KL}(G_i||G_{merge(i,j)}) + w_j \mathcal{KL}(G_j||G_{merge(i,j)}) \tag{A.3}$$

This measure can be shown to be an upper bound on the desired measure, Equation A.2, as follows:

$$\begin{aligned}
w_i \mathcal{KL}(G_i||G_{merge(i,j)}) + w_j \mathcal{KL}(G_j||G_{merge(i,j)}) &\overset{?}{\geq} \mathcal{KL}((w_i G_i + w_j G_j)||G_{merge(i,j)}) \\
w_i \int G_i \log \frac{G_i}{G_{merge(i,j)}} + w_j \int G_j \log \frac{G_j}{G_{merge(i,j)}} &\overset{?}{\geq} \int \left(w_i G_i + w_j G_j\right) \log \frac{\left(w_i G_i + w_j G_j\right)}{G_{merge(i,j)}(x)} \\
w_i \int G_i \log G_i + w_j \int G_j \log G_j dx &\overset{?}{\geq} w_i \int G_i \log \left(w_i G_i + w_j G_j\right) + w_j \int G_j \log \left(w_i G_i + w_j G_j\right) \\
\int G_i \log G_i - \int G_i \log \left(w_i G_i + w_j G_j\right) &\geq \frac{wj}{wi} \left(\int G_j \log \left(w_i G_i + w_j G_j\right) - \int G_j \log G_j\right)
\end{aligned}$$

From Gibbs' inequality [79] we have that $\int G_i \log G_i \geq \int G_i \log \left(w_i G_i + w_j G_j\right)$, which implies that L.H.S. ≥ 0 and R.H.S. ≤ 0. The approximation $w_i \mathcal{KL}(G_i||G_{merge(i,j)}) + w_j \mathcal{KL}(G_j||G_{merge(i,j)})$ is thus an upper bound on $\mathcal{KL}((w_i G_i + w_j G_j)||G_{merge(i,j)})$, with the two measures coinciding at 0 when $G_i = G_j$. When the two distributions are perfectly separated (so that $G_i(x) = 0|_{\forall x: G_j(x) > 0}$ and vice versa) it is straightforward to show that the two measures differ by a constant value of $w_i \log(w_i) + w_j \log(w_j)$.

Minimising Equation A.3 thus minimises an upper bound on the expected decrease in log-likelihood when the chosen pair of components is merged. Nonetheless,

it should be noted that there is no guarantee that the ordering of the merging costs for a given mixture model provided by Equations A.3 and A.2 should be identical.

Appendix B

Publications

The research presented in this thesis has also formed the basis for several peer-reviewed conference papers, listed as follows.

1. R. R. Sillito & R. B. Fisher. Incremental One-class Learning with Bounded Computational Complexity. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, LNCS 4668, pp 58-67, 2007.

2. R. R. Sillito & R. B. Fisher. Semi-supervised Learning for Anomalous Trajectory Detection. In *Proceedings of the British Machine Vision Conference (BMVC)*, pp 1035-1067, 2008.

3. R. R. Sillito & R. B. Fisher. Parametric Trajectory Representations for Behaviour Classification. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2009.

Bibliography

[1] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz. Robust real-time unusual event detection using multiple fixed-location monitors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(3):555–560, March 2008.

[2] E. Andrade, S. Blunsden, and R. Fisher. Hidden markov models for optical flow analysis in crowds. In *Proc. 18th International Conference on Pattern Recognition ICPR 2006*, volume 1, pages 460–463, 2006.

[3] E. Andrade, S. Blunsden, and R. Fisher. Modelling crowd scenes for event detection. In *Proc. 18th International Conference on Pattern Recognition ICPR 2006*, volume 1, pages 175–178, 2006.

[4] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

[5] O. D. Arandjelovic and R. Cipolla. Incremental learning of temporally-coherent gaussian mixture models. In *Proc. British Machine Vision Conference*, 2005.

[6] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[7] S. Baluja. Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. In *Advances in Neural Information Processing Systems (NIPS) 11*, pages 854–860, 1998.

[8] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling*. Morgan Kaufmann: San Mateo, CA., 1987.

[9] F. Bashir, A. Khojar, and D. Shonfeld. Real-time motion trajectory-based indexing and retrieval of video sequences. *IEEE Transactions on Multimedia*, 9:58–68, 2007.

[10] F. Bashir, A. Khokhar, and D. Schonfeld. Object trajectory-based activity classification and recognition using Hidden Markov Models. *IEEE Transactions on Image Processing*, 16(7):1912–1919, 2007.

[11] C. M. Bishop. Novelty detection and neural network validation. *IEE Proceedings - Vision, Image & Signal Processing*, 141:217–222, 1994.

[12] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[13] A. Blake and M. Isard. *Active Contours*. Springer, 1997.

[14] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. 11th Annual Conference on Computational Learning Theory (COLT)*, pages 92–100, 1998.

[15] O. Boiman and M. Irani. Detecting irregularities in images and in video. *International Journal of Computer Vision*, 74:17–31, 2007.

[16] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988.

[17] M. Brand and V. Kettnaker. Discovery and segmentation of activities in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:844–851, 2000.

[18] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 994–999, 1997.

[19] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–168, 1998.

[20] K. P. Burnham and D. R. Anderson. Multimodel inference: Understanding AIC and BIC in model selection. *Sociological Methods & Research*, 33:261–304, 2004.

[21] C. S. Burrus, R. A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms*. Prentice Hall, 1997.

[22] Y. Cai and R. Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. In *Proc. ACM SIGMOD*, pages 599 – 610, 2004.

[23] Cambridge Systematics Inc. NGSIM Vehicle Trajectory Datasets. http://ngsim.camsys.com/.

[24] T. Cham and R. Cipolla. Automated B-spline curve representation incorporating MDL and error-minimizing control point insertion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:49–53, 1999.

[25] F. K. P. Chan, A. W. C. Fu, and C. Yu. Haar wavelets for efficient similarity search of time-series: with and without time warping. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):686–705, 2003.

[26] O. Chapelle, B. Scholkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006.

[27] C. Charron, Y. Hicks, P. Hall, and D. Cosker. Incremental learning of dynamical models of faces. In *Proc. British Machine Vision Conference (BMVC)*, 2009.

[28] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 14:201–221, 1994.

[29] S. Coles. *An Introduction to Statistical Modeling of Extreme Values*. Springer, 2001.

[30] A. Datta, M. Shah, and N. Da Vitoria Lobo. Person-on-person violence detection in video data. In *Proc. 16th International Conference on Pattern Recognition*, volume 1, pages 433–438, 11–15 Aug. 2002.

[31] A. Declercq and J. H. Piater. Online learning of gaussian mixture models - a two-level approach. In *Proc. International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 1, pages 605–611, 2008.

[32] H. Dee and D. Hogg. Detecting inexplicable behaviour. In *Proc. British Machine Vision Conference (BMVC)*, pages 477–486, 2004.

[33] H. Dee and S. Valestin. How close are we to solving the problem of automated visual surveillance? *Machine Vision and Applications*, 2007.

[34] H. M. Dee and D. C. Hogg. On the feasibility of using a cognitive model to filter surveillance data. In *IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2005.

[35] A. Dempster and N. M. Laird. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39:1–38, 1977.

[36] F. Denis, R. Gilleron, and M. Tommasi. Text classification from positive and unlabeled examples. In *Proc. International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IMPU)*, pages 1927–1934, 2002.

[37] R. Duin. On the choice of smoothing parameters for Parzen estimators of probability density functions. *IEEE Trans. Computers*, C-25:1175–1179, 1976.

[38] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 726–733, 2003.

[39] K. Etemad and R. Chellappa. Separability-based multiscale basis selection and feature extraction for signal and image classification. *IEEE Transactions on Image Processing*, 7(10):1453–1465, 1998.

[40] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[41] J. Fernyhough, A. Cohn, and D. Hogg. Constructing qualitative event models automatically from video input. *Image and Vision Computing*, 18:81–103, 2000.

[42] R. B. Fisher, J. Santos-Victor, and J. Crowley. CAVIAR test case scenarios. http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.

[43] Y. Freund and R. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14:771–780, 1999.

[44] N. Friedman and S. Russell. Image segmentation in video sequences: A probabilistic approach. In *Proc. Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 175–181, 1997.

[45] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[46] A. Galata, N. Johnson, and D. Hogg. Learning Variable-Length Markov Models of behavior. *Computer Vision and Image Understanding*, 81:398–413, 2001.

[47] J. Goldberger and S. Roweis. Hierarchical clustering of a mixture model. In *Advances in Neural Information Processing Systems (NIPS) 17*, pages 505–512, 2005.

[48] S. Gong and T. Xiang. Recognition of group activities using dynamic probabilistic networks. In *Proc. Ninth IEEE International Conference on Computer Vision*, pages 742–749 vol.2, 2003.

[49] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, 2007.

[50] A. Gueziec and N. Ayache. Smoothing and matching of 3D space curves. *International Journal of Computer Vision*, 12:79–104, 1994.

[51] P. Hall and Y. Hicks. A method to add gaussian mixture models. Technical Report 2004-03, University of Bath, Department of Computer Science, 2004.

[52] P. Hall, D. Marshall, and R. Martin. Merging and splitting eigenspace models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):1042–1049, 2000.

[53] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, 1 edition, 2001.

[54] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[55] P. Hayton, B. Scholkopf, L. Tarassenko, and P. Anuzis. Support vector novelty detection applied to jet engine vibration spectra. In *Advances in Neural Information Processing Systems (NIPS) 13*, pages 946–952, 2000.

[56] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical Review E*, 51:4282, 1995.

[57] Y. A. Hicks, P. Hall, and A. Marshall. A method to add hidden markov models with application to learning articulated motion. In *Proc. BMVC*, 2003.

[58] H. Hoffmann. Kernel PCA for novelty detection. *Pattern Recognition*, 40:863–874, 2007.

[59] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics - Part C*, 34:334–352, 2004.

[60] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank. A system for learning statistical motion patterns. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28:1450–1464, 2006.

[61] W. Hu, D. Xie, T. Tan, and S. Maybank. Learning activity patterns using fuzzy self-organizing neural network. *IEEE Transactions on Systems, Man,and Cybernetics - Part B*, 34:1618–1626, 2004.

[62] N. Japkowicz, H. Mark, and A. Gluck. Nonlinear autoassociation is not equivalent to PCA. *Neural Computation*, 12:531–545, 2000.

[63] N. Japkowicz, C. Myers, and M. Gluck. A novelty detection approach to classification. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 518–523, 1995.

[64] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14:609–615, 1996.

[65] N. Johnson and D. Hogg. Representation and synthesis of behaviour using gaussian mixtures. *Image and Vision Computing*, 20:889–894, 2002.

[66] I. Junejo, O. Javed, and M. Shah. Multi feature path modeling for video surveillance. In *Proc. IEEE International Conference on Pattern Recognition (ICPR)*, pages 716–719, 2004.

[67] I. N. Junejo and H. Foroosh. Euclidean path modeling for video surveillance. *Image and Vision Computing*, 26:512–528, 2008.

[68] M. W. Kadous. *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series*. PhD thesis, School of Computer Science & Engineering, University of New South Wales, 2002.

[69] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for realtime tracking with shadow detection. In *In Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems (AVBS01)*, 2001.

[70] Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *Proc. Tenth IEEE International Conference on Computer Vision ICCV 2005*, volume 1, pages 166–173 Vol. 1, 2005.

[71] T. Kohonen. *Self-Organizing Maps*. Springer, 2001.

[72] M. Kristan, D. Skocaj, and A. Leonardis. Incremental learning with gaussian mixture models. In *Proc. Computer Vision Winter Workshop CVWW*, pages 25–32, 2008.

[73] I. Laptev, B. Caputo, C. Schuldt, and T. Lindeberg. Local velocity-adapted motion events for spatio-temporal recognition. *Computer Vision and Image Understanding*, 108:207–229, 2007.

[74] P. Laskov, C. Gehl, S. Kruger, and K.-R. Muller. Incremental support vector learning: Analysis, implementation and applications. *Journal of Machine Learning Research*, 7:1909–1936, 2006.

[75] O. Ledoit and M. Wolf. Some hypothesis tests for the covariance matrix when the dimension is large compared to the sample size. *The Annals of Statistics*, 30(4):1081–1102, 2002.

[76] D.-S. Lee. Effective gaussian mixture learning for video background subtraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):827–832, 2005.

[77] D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proc. International Conference on Machine Learning (ICML)*, pages 48–156, 1994.

[78] B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. In *Proc. International Conference on Machine Learning (ICML)*, pages 387 – 394, 2002.

[79] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[80] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob.*, volume Vol. 1, pages 281–297. Univ. of Calif. Press, 1967.

[81] D. R. Magee. Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, 22:143–155, 2001.

[82] D. Makris and T. Ellis. Path detection in video surveillance. *Image and Vision Computing*, 20:895–903, 2002.

[83] D. Makris and T. Ellis. Learning semantic scene models from observing activity in visual surveillance. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 35:397–408, 2005.

[84] M. Markou and S. Singh. Novelty detection: a review - part 2: neural network based approaches. *Signal Processing*, 83:2499–2521, 2003.

[85] S. McKenna and H. Nait-Charif. Learning spatial context from tracking using penalised likelihoods. In *Proc. 17th International Conference on Pattern Recognition ICPR 2004*, volume 4, pages 138–141 Vol.4, 2004.

[86] D. C. Montgomery and G. C. Runger. *Applied Statistics and Probability for Engineers*. John Wiley & Sons, 3rd edition, 2003.

[87] P. J. Moreno and S. Agarwal. An experimental study of em-based algorithms for semi-supervised learning in audio classification. In *Proc. ICML-2003 Workshop on the Continuum from Labeled to Unlabeled Data*, 2003.

[88] R. Morris and D. Hogg. Statistical models of object interaction. *International Journal of Computer Vision*, 37:209–215, 2000.

[89] A. Naftel and F. B. Anwar. Visual recognition of manual tasks using object motion trajectories. In *Proc. IEEE International Conference on Video and Signal Based Surveillance AVSS '06*, pages 69–69, 2006.

[90] A. Naftel and S. Khalid. Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space. *Multimedia Systems*, 12:227–238, 2006.

[91] A. Naftel and S. Khalid. Motion trajectory learning in the DFT-coefficient feature space. In *Proc. IEEE International Conference on Computer Vision Systems ICVS '06*, pages 47–47, 2006.

[92] H. Nait-Charif and S. McKenna. Activity summarisation and fall detection in a supportive home environment. In *Proc. 17th International Conference on Pattern Recognition ICPR 2004*, volume 4, pages 323–326 Vol.4, 2004.

[93] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. MIT Press, 1998.

[94] A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems (NIPS) 14*, 2002.

[95] J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. In *Proc. British Machine Vision Conference (BMVC)*, pages 1249–1258, 2006.

[96] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134, 2000.

[97] N. Oliver, B. Rosario, and A. Pentland. A Bayesian computer vision system for modeling human interactions. 22(8):831–843, 2000.

[98] J. Owens and A. Hunter. Application of the self-organising map to trajectory classification. In *Proc. Third IEEE Visual Surveillance Workshop*, pages 77–83, 2000.

[99] T. Petsche, A. Marcantonio, C. Darken, S. J. Hanson, G. M. Kuhn, and I. Santoso. A neural network autoassociator for induction motor failure prediction. In *Advances in Neural Information Processing Systems*, 1996.

[100] C. Piciarelli and G. Foresti. Event recognition by dynamic trajectory analysis and prediction. In *Proc. IEE Symposium on Imaging for Crime Detection and Prevention (ICDP)*, pages 131–134, 2005.

[101] C. Piciarelli and G. L. Foresti. On-line trajectory clustering for anomalous events detection. *Pattern Recognition Letters*, 27:1835–1842, 2006.

[102] L. Piegl. On NURBS: A survey. *IEEE Computer Graphics and Applications*, 11:55 – 71, 1991.

[103] F. Porikli. Learning object trajectory patterns by spectral clustering. In *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, volume 2, pages 1171–1174Vol.2, 27-30 June 2004.

[104] F. Porikli. Trajectory distance metric using Hidden Markov Model based representation. In *Proc. IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS-ECCV)*, pages 15–22, 2004.

[105] F. Porikli and T. Haga. Event detection by eigenvector decomposition using object and frame features. In *Computer Vision and Pattern Recognition Workshop, 2004 Conference on*, pages 114–114, 27-02 June 2004.

[106] W. H. Press, S. A. Teukolsky, W. T. Vettering, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007.

[107] L. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[108] C. Ratanamahatana and E. J. Keogh. Making time-series classification more accurate using learned constraints. In *Proc. SIAM International Conference on Data Mining (SDM)*, pages 11–22, 2004.

[109] S. Roberts. Novelty detection using extreme value statistics. *IEE Proceedings - Vision, Image & Signal Processing*, 146(3):124–129, 1999.

[110] S. Roberts. Extreme value statistics for novelty detection in biomedical data processing. *IEE Proceedings -Science, Measurement and Technology*, 147(6):363–367, 2000.

[111] S. J. Roberts, D. Husmeier, I. Rezek, and W. Penny. Bayesian approaches to gaussian mixture modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1133–1142, 1998.

[112] S. J. Roberts and L. Tarassenko. A probabilistic resource allocating network for novelty detection. *Neural Computation*, 6:270–284, 1994.

[113] N. Robertson and I. Reid. Estimating gaze direction from low-resolution faces in video. In *Proc. European Conference on Computer Vision (ECCV)*, volume LNCS 3952, pages 402–415, 2006.

[114] N. Robertson and I. Reid. A general method for human activity recognition in video. *Computer Vision and Image Understanding*, 104:232–248, 2006.

[115] N. Robertson, I. Reid, and J. Brady. Causal reasoning about human activity in video. *IEEE Transactions on Systems, Man and Cybernetics A*, (submitted).

[116] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25:117–149, 1996.

[117] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *Proc. IEEE Workshop on Motion and Video Computing (WACV/MOTION)*, pages 29–36, 2005.

[118] V. Roth. Kernel Fisher discriminants for outlier detection. *Neural Computation*, 18:942–960, 2006.

[119] D. E. Rumelhart and D. Zipser. Feature discovery by competitive learning. *Cognitive Science*, 9:75–112, 1985.

[120] E. Sahouria and A. Zakhor. Motion indexing of video. In *Proc. IEEE International Conference on Image Processing (ICIP)*, volume 2, pages 526–529, 1997.

[121] I. Saleemi, K. Shafique, and M. Shah. Probabilistic modeling of scene dynamics for applications in visual surveillance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[122] S. Salvador and P. Chan. Towards accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11:561–580, 2007.

[123] M. Sato and S. Ishii. On-line em algorithm for the normalized gaussian network. *Neural Computation*, 12(2):407–432, 2000.

[124] P. J. Schneider. NURB curves: A guide for the uninitiated. *develop, The Apple Technical Journal*, 25:48–74, 1996.

[125] B. Scholkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.

[126] B. Scholkopf, A. Smola, and K. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[127] G. L. Scott and H. C. Longuet-Higgins. Feature grouping by 'relocalisation' of eigenvectors of the proximity matrix. In *Proc. British Machine Vision Conference (BMVC)*, pages 103–108, 1990.

[128] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional SIFT descriptor and its application to action recognition. In *Proc. International Conference on Multimedia*, pages 357–360, 2007.

[129] B. Settles. Active learning literature survey. Technical report, University of Wisconsin Madison, Computer Sciences TR 1648, 2009.

[130] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[131] H. Sidenbladh, M. J. Black, and L. Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *Proc. European Conference on Computer Vision (ECCV)*, volume LNCS 2350, pages 784–800, 2002.

[132] R. R. Sillito and R. B. Fisher. Incremental one-class learning with bounded computational complexity. In *Proc. International Conference on Artificial Neural Networks (ICANN)*, volume LNCS 4668, pages 58–67, 2007.

[133] R. R. Sillito and R. B. Fisher. Semi-supervised learning for anomalous trajectory detection. In *Proc. British Machine Vision Conference (BMVC)*, pages 1035–1044, 2008.

[134] R. R. Sillito and R. B. Fisher. Parametric trajectory representations for behaviour classifcation. In *Proc. British Machine Vision Conference (BMVC)*, 2009.

[135] M. Song and H. Wang. Highly efficient incremental estimation of gaussian mixture models for online data stream clustering. In *Proc. Society of Photo-Optical Instrumentation Engineers (SPIE)*, volume 5803, pages 174–183, 2005.

[136] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:747–757, 2000.

[137] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1995.

[138] N. Sumpter and A. Bulpitt. Learning spatio-temporal patterns for predicting object behaviour. *Image and Vision Computing*, 18:697–904, 2000.

[139] L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady. Novelty detection for the identification of masses in mammograms. In *Proc. IEEE International Conference on Artificial Neural Networks (ICANN)*, pages 442–447, 1995.

[140] D. Tax. DDtools, the Data description toolbox for Matlab. version 1.5.5.

[141] D. Tax and R. Duin. Support vector domain description. *Pattern Recognition Letters*, 20:1191–1199, 1999.

[142] D. Tax and R. Duin. Support vector data description. *Machine Learning*, 54:45–66, 2004.

[143] D. Tax and K.-R. Muller. A consistency-based model selection for one-class classification. In *Proc. IEEE International Conference on Pattern Recognition (ICPR)*, volume 3, pages 363–366, 2004.

[144] D. M. J. Tax and R. P. W. Duin. Outlier detection using classifier instability. In *Advances in Pattern Recognition, Proc. Joint IAPR International Workshops*, volume LNCS 1451, pages 593–601, 1998.

[145] D. M. J. Tax and R. P. W. Duin. Uniform object generation for optimizing one-class classifiers. *Journal of Machine Learning Research*, 2:155–173, 2001.

[146] D. M. J. Tax and P. Laskov. Online SVM learning: from classification to data description and back. In *Proc. 13th IEEE Workshop on Neural Networks for Signal Processing (NNSP)*, pages 499–508, 2003.

[147] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581, 2006.

[148] M. K. Titsias. *Unsupervised Learning of Multiple Objects in Images*. PhD thesis, School of Informatics, University of Edinburgh, 2005.

[149] M. K. Titsias and C. K. I. Williams. Sequential learning of layered models from video. In *Toward Category-Level Object Recognition*, volume LNCS 4170, pages 577–595, 2006.

[150] G. T. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12:261–268, 1980.

[151] J. J. Verbeek, N. Vlassis, and B. Krose. Efficient greedy learning of gaussian mixture models. *Neural Computation*, 15:469–485, 2003.

[152] N. Vlassis and A. Likas. A kurtosis-based dynamic approach to gaussian mixture modeling. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 29(4):393–399, 1999.

[153] N. Vlassis and A. Likas. A greedy em algorithm for gaussian mixture learning. *Neural Processing Letters*, 15:77–87, 2002.

[154] X. Wang and E. Grimson. Trajectory analysis and semantic region modeling using a nonparametric Bayesian model. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[155] X. Wang, X. Ma, and E. Grimson. Unsupervised activity perception by hierarchical bayesian models. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 17-22 June 2007.

[156] X. Wang, X. Ma, and E. Grimson. Unsupervised activity perception in crowded and complicated scenes using hierarchical Bayesian models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31:539–555, 2009.

[157] X. Wang, K. Tieu, and E. Grimson. Learning semantic scene models by trajectory analysis. In *Proc. European Conference on Computer Vision (ECCV)*, volume LNCS 3954, pages 110–123, 2006.

[158] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2004.

[159] L. Wasserman. *All of Nonparametric Statistics*. Springer, 2006.

[160] A. R. Webb. *Statistical Pattern Recognition*. John Wiley & Sons, Ltd., second edition, 2002.

[161] E. W. Weisstein. "kurtosis" from mathworld–a wolfram web resource.

[162] Y.-L. Wu, D. Agrawal, and A. E. Abbadi. A comparison of DFT and DWT based similarity search in time-series databases. In *Proc. 9th International Conference on Information and Knowledge Management (CIKM)*, pages 488 – 495, 2000.

[163] T. Xiang and S. Gong. Beyond tracking: Modelling activity and understanding behaviour. *International Journal of Computer Vision*, 67:21–51, 2006.

[164] T. Xiang and S. Gong. Model selection for unsupervised learning of visual context. *International Journal of Computer Vision*, 69:181 – 201, 2006.

[165] T. Xiang and S. Gong. Activity based surveillance video content modelling. *Pattern Recognition*, 7:2309–2326, 2008.

[166] T. Xiang and S. Gong. Incremental and adaptive abnormal behaviour detection. *Computer Vision and Image Understanding*, 111:59–73, 2008.

[167] T. Xiang and S. Gong. Spectral clustering with eigenvector selection. *Pattern Recognition*, 41:1012–1029, 2008.

[168] T. Xiang and S. Gong. Video behavior profiling for anomaly detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30:893–908, 2008.

[169] T. Xiang, S. Gong, and D. Parkinson. Autonomous visual events detection and classification without explicit object-centred segmentation and tracking. In *Proc. British Machine Vision Conference (BMVC)*, pages 233–243, 2002.

[170] X. L. Xie and G. Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, 1991.

[171] K. Yamanishi, J. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 8:275–300, 2004.

[172] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 189 – 196, 1995.

[173] D.-Y. Yeung and C. Chow. Parzen-window network intrusion detectors. In *Proc. IEEE International Conference on Pattern Recognition (ICPR)*, volume 4, pages 385–388, 2002.

[174] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38, 2006.

[175] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Advanced in Neural Information Processing Systems (NIPS) 17*, pages 1601–1608, 2005.

[176] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowanr. Semi-supervised adapted HMMs for unusual event detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 611 – 618, 2004.

[177] X. Zhang, M. King, and R. J. Hyndman. A Bayesian approach to bandwidth selection for multivariate kernel density estimation. *Computational Statistics & Data Analysis*, 50:3009–3031, 2006.

[178] H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 819–826, 2004.

[179] Z.-H. Zhou and M. Li. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541, 2005.

[180] X. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin Madison, Computer Sciences TR 1530, 2008.

[181] D. A. Zighed, S. Lallich, and F. Muhlenbach. Separability index in supervised learning. In *Principles of Data Mining and Knowledge Discovery*, volume LNCS 2431, pages 241–267, 2002.

[182] Z. Zivkovic and F. van der Heijden. Recursive unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:651–656, 2004.