Online Clustering of High-Dimensional Trajectories under Concept Drift

Georg Krempl[1], Zaigham Faraz Siddiqui[2], and Myra Spiliopoulou[2]

[1] University of Graz
georg.krempl@uni-graz.at
[2] University of Magdeburg
{myra,siddiqui}@iti.cs.uni-magdeburg.de

Abstract. Historical transaction data are collected in many applications, e.g., patient histories recorded by physicians and customer transactions collected by companies. An important question is the learning of models upon *the primary objects* (patients, customers) rather than the transactions, especially when these models are subjected to drift.

We address this problem by combining advances of online clustering on multivariate data with the trajectory mining paradigm. We model the measurements of each individual primary object (e.g. its transactions), taken at irregular time intervals, as a trajectory in a high-dimensional feature space. Then, we cluster individuals with similar trajectories to identify sub-populations that evolve similarly, e.g. groups of customers that evolve similarly or groups of employees that have similar careers.

We assume that the multivariate trajectories are generated by drifting Gaussian Mixture Models. We study (i) an EM-based approach that clusters these trajectories incrementally as a reference method that has access to all the data for learning, and propose (ii) an online algorithm based on a Kalman filter that efficiently tracks the trajectories of Gaussian clusters. We show that while both methods approximate the reference well, the algorithm based on a Kalman filter is faster by one order of magnitude compared to the EM-based approach.

Keywords: On-Line clustering, incremental clustering, high-dimensional trajectories, clusters of trajectories, high-dimensional clustering, stream clustering.

1 Introduction

Finding clusters on a mixture of multivariate distributions is a well-investigated problem. A typical assumption is that the data can be represented as vectors. Although this requirement seems fairly easy to fulfil, it is counter intuitive for many applications. For example, consider tasks like learning the progress of some chronicle disease or the seasonal purchase habits of customers. The data at hand are measurements of fixed dimensionality for each individual, e.g. transactions performed by customers or recordings of a patient's blood pressure. However,

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part II, LNAI 6912, pp. 261–276, 2011.

we cannot assume that there is the same number of measurements for each individual, nor that the measurements have been performed at the same intervals for them. Hence, vector representation becomes too restrictive. In this study, we propose to model the measurements of individuals over time as ongoing *trajectories* in a high-dimensional space, and learn patterns over the evolving data by clustering together individuals whose trajectories deploy similarly.

Gaffney, Smyth and colleagues have proposed probabilistic methods for the clustering of trajectories of measurements [6,2,3]. However, they assume that the trajectories are available in their entirety, hence the algorithm can exploit all the data to learn the model. There are two problems with this approach. First, there are many cases where model learning is done *while* data are being recorded: not an ultimate model over all data is then of interest, but rather knowledge on how models change from one time point to the next. Second, a model over all data may smooth away concept drifts that are valuable to the observer. Some later works, like [7,14] address the first problem by adapting the model as time progresses and new data arrive. However, the problem of adapting to drift is treated in a rather rudimentary way, namely by considering only special aspects of drift (e.g. cluster split).

In this study, we propose an adaptive learning approach over the trajectories of individuals *as* the trajectories receive further measurements. We allow for measurements that are multi-variate data, and recorded at irregular intervals. We assume that these measurements are generated from a mixture of Gaussians, but allow for gradual and abrupt concept changes (drifts and shifts). For example, consider aeroplanes from Heathrow to Montreal and from Madrid to Montreal. They fly two clusters of routes, with the former being closer to Iceland than the other. After May 21, this clusters' route changes due to the volcanic eruption. While the cluster is still there, its aeroplanes move differently for a part of their route. This is an example of "abrupt concept change". For "gradual concept change", consider a cluster A of elderly women and B of women in their late forties. Members of A show a more and more intensive preference for books in big font; this cluster is drifting as a whole, indicating that its members become older. Further, some members of cluster B show an increasing preference for sports and gradually build a separate cluster C, i.e. there is drift inside the cluster B.

We approach model adaptation by using an incremental, but offline EM-algorithm for clustering trajectories. We further propose a method that tracks the trajectories of the Gaussian clusters (as whole entities) using Kalman filter. This second method is fast and adapts well - it is appropriate for online processing. We compare this algorithm to the offline EM-algorithm as a reference algorithm which can exploit all data for model learning, and we show that the model quality of our method is comparable to the reference, while the execution time is lower by one order of magnitude.

The paper is organised as follows. In the next section, we discuss related work. In section 3 we first discuss Gaussian mixture models and the Kalman filter and then describe our two adaptive methods. The experiments are reported in

section 4. The last section concludes our paper with a summary and a list of future research issues.

2 Related Work

Our work is on learning mixture models for the multi-variate data trajectories of evolving objects. Related work includes mixture models, object tracking methods, and methods that cluster objects which move in a conventional geometric space (e.g. cars, persons, vessels).

Mixture models: The work of Gaffney and Smyth [6] was one of the earliest to consider the problem of clustering trajectories without a vector-based representation. The approach implicitly assumes that the objects and their trajectories follow some basic model, i.e. the Gaussian Mixture Model, and uses a variant of Expectation-Maximisation algorithm to cluster sets of trajectories. Our approach is inspired by that study. However, we do not limit our data to time series, as in [6], but allow for multivariate measurements that constitute a stream rather than a closed time series. Further, the number of measurements per individual may vary from one individual to the other.

The method of Han et al. [7] does trajectory clustering by using the mixture model for automatic speech recognition. They report on the general problem of different initial cluster assignments leading to different EM clusters. To sidestep this problem, their variant of EM increases the number of clusters incrementally. The algorithm starts by computing the best fitting polynomial for the complete data set and then successively splits the cluster with the largest weight of the mixture densities until K clusters are obtained.

The work of Xiong et al. deals with the problem of tracking objects from a video stream [14]. They model these objects using a Gaussian mixture model. For the first frame, the parameters are initialised using the traditional mixture model. For each subsequent frame these parameters are predicted with the help of Kalman filter. Unlike the method of Han et al. [7], which utilises splitting during initialisation, Xiong et al. integrate "split", "merge" and "delete" operation into their dynamic Gaussian mixture model.

The follow-up work of Cadez, Gaffney and Smyth [2] deals with the problem of clustering individuals that are associated with non-vector, non-multi-variate data measurements. The example of such data can be patients at a hospital which are associated with multiple but varying number of time-series. The proposed solution is based on generative mixture models and can accept the data in its native form, i.e. with varying data sizes and types across the individuals.

Object Tracking: Kalman filtering [8] is a widely used technique in signal analysis, computational visualistics and other domains. It is an iterative state estimation filter that supports the estimation of current and subsequent states, when the exact nature of the modelled system is undetermined. The power of a Kalman filter comes from its ability to estimate the subsequent states in the presence of noise. However, calibrating the parameters such as process and measurement noise covariance is not trivial and even a simple predictions model can outperform Kalman filter in the absences of an accurate model [5].

Initially the filter was designed for spacecraft navigation but has been used widely for multi-object tracking [9,12,1]. The method discussed in Pathan et al. [12] tracks vehicles whose paths overlap or get merged. They denote it as a confusion state and make use of a Kalman filter to identify the trajectories of the individual objects. Mederios et al. [11] presented a comprehensive approach on tracking objects, using a sensor network of cameras. A Kalman filter is used for aggregating the data distributively, which is then shared among sensors to propagate the state of the objects as they move. As stated earlier, Xiong et al. [14] keep track of their elliptical objects represented by Gaussian clusters through a Kalman filter. In addition, they use their filter in order to predict the parameters of the clusters in the next timepoint.

The work of Ellis et al. [4] uses Gaussian process regression to model the trajectories of pedestrian trajectory patterns. They couple the Gaussian process with Monte Carlo methods and a Kalman filter to achieve a long term prediction. They show in their empirical evaluation that their Monte Carlo method has a much better performance than their Kalman filter over long term predictions. The poor performance of their Kalman filter is predictable as it needs measurements for calculating the error and updating the estimates. However, for very short term prediction, Kalman filter shows competitive performance.

Clustering moving objects: Our approach is based on the trajectory clustering paradigm introduced by Gaffney and Smyth [6] and has very similar problem definition with that of Cadez et al. [2]. Our first algorithm, which is used for initial cluster identification, is based on the EM algorithm by Gaffney and Smyth [6]. However, we extended the original univariate algorithm to handle multi-variate data. Unlike the method of Cadez et al. [2], where individuals are associated with time-series only, in our setting individuals are associated with multi-dimensional objects. Individuals can also vary in the number of their associated observations (also denoted objects). The EM algorithm of [6] has the drawback of not being online. This is due the fact that all historic observations of an individual are required in re-fitting a mixture regression model. This might cause high computational costs and might be susceptible to sudden shift, i.e. sudden, non-smooth changes in the position of cluster centres. Therefore, we introduce an online cluster tracking algorithm for the trajectory clustering paradigm. This algorithm is based on Kalman filter technique and can be easily initialised by using the parameter estimates obtained by the EM algorithm.

Related to our work is the method of Xiong et al. [14]. However, there are some important differences. First, their method is not online and performs adjustment of Gaussian mixture models at each frame. The input to their methods are the individual points and the target is to find the trajectories, whereas in our method data points are already associated to a trajectory and the task is 1) to cluster them and 2) to estimate the path of the true cluster centre over time. Moreover, its not entirely clear what they use as input to their algorithm, i.e. whether they use individual points or a batch of points.

Table 1. List of Symbols

	Symbol	Description
General	D	Dimensionality of feature space (e.g. $D = 2$ in the bivariate case)
	P	Order of the polynomials in the underlying mixture regression problem
	K	Number of clusters in the mixture
	N	Total number of observations (over all individuals)
	n	Number of observations for a given individual
	M	Total number of individuals
EM-Algorithm	z	D-dimensional feature vector (measurement)
	s, t_s	Discrete time step s at continuous time t_s
	Θ,θ_k	Parameter set (of the complete model or of cluster k)
	α_k, x_k, Σ_k	Mixing proportion, mean, and covariance of cluster k
	β_k	Regression coefficients (including cluster means x_k) for cluster k
	$\beta_{k\,dp}$	p-th regression coefficient for feature d in cluster k
Kalman Filter	x_s, z_s	True state and measurement at (discrete) time step s
	A	State transition matrix
	w_s, Q	Process noise at time step s, process noise covariance matrix
	H	Measurement (or state-to-signal) matrix
	v_s, R	Measurement noise at time step s, measurement noise cov. matrix
	K	Kalman gain matrix
	$P_{\bar{s}}$, P_s	Estimate covariances (a priori, and posterior)

3 Method TRACER

The main contribution of this paper is the extension of the trajectory clustering method provided by Gaffney and Smyth [6] to multivariate data streams which require online algorithms. We suggest the following approach: First, an extended variant of the EM algorithm of [6] is used to perform an initial clustering. As a result of this algorithm, parameter estimates for the underlying mixture regression are available. These estimates are then used to initialise a Kalman filter for tracking the clusters and updating the model parameter estimates regularly.

In the subsequent sections, we first give a brief overview of the Gaussian mixture models and the Kalman filter for completeness before outlining our tracking algorithm. A list of parameters and symbols is given in Table 1.

3.1 Gaussian Mixture Model

The objectives of our tracking algorithm are first to track clusters of individuals over time. Second, the parameters estimates of the assumed underlying data generating process should be updated. The input to clustering in this context are sets of individuals that are observed over a longer time span, e.g., patients and customers. Each individual is associated with a multiple of measurements (or observations). The measurements which have been recorded for an individual result in a trajectory. The number of measurements per individual as well as the observation time, i.e. the time at which a measurement is recorded, can differ between the individuals. For example, customers could make a varying number of transactions and not all customers will make the transactions at the same time. Furthermore it is reasonable to assume that the population of individuals is not homogeneous, but consists of sub-populations (clusters). Each sub-population might evolve differently over time. When a sub-population is affected by drift,

the behaviour of its individuals might also change. This results in a mixture regression model. If the distribution of measurements of a sub-population at a given moment in time is assumed to follow a Gaussian distribution, the following model of a mixture of Gaussians is obtained:

Algorithm 1: Incremental Clustering Of Trajectories using EM Algorithm

Input : new batch of data Z_i, old model $\mathcal{C}_{i-1}$ old estimates Θ_{i-1}
Output: new model $\mathcal{C}_i$, new estimates Θ_i
1 Computer the membership probabilities for the individuals in Z_i using $\mathcal{C}_{i-1}$.
2 Update the likelihood estimates
3 Repeat until re-convergence for new estimates and new clustering.

Let $z_i = z_{i1}, z_{i2}, \cdots, z_{in}$ be the n observations of the i-th individual. These observations follow a Gaussians mixture model with cluster means $x_1, x_2, \cdots, x_K$ and probability density function:

$$p(z_i; \Theta) = \prod_{l=1}^{n} \sum_{k=1}^{K} \alpha_k p(z_{il}; \theta_k) \tag{1}$$

where α_k is the mixing proportion of the k^{th} cluster such that $\sum_{k=1}^{K} \alpha_k = 1$. The density $p(z; \Theta)$ of the k-th cluster in the mixture with mean x_k follows a normal probability distribution

$$p(z; \theta_k) = (2\pi)^{-\frac{D}{2}} |\Sigma_k^{-1}|^{\frac{1}{2}} exp\left\{ -\frac{1}{2}(z - x_k)^T \Sigma_k^{-1} (z - x_k) \right\} \tag{2}$$

where $\theta_k = x_k, \Sigma_k$ is the centroid and covariance matrix for k^{th} cluster at a given moment in time, respectively.

For a given set of M independent individuals $Z = z_1, z_2, \ldots, z_M$, the log-likelihood with respect to the Gaussian mixture is given by

$$logp(Z; \Theta) = log \prod_{i=1}^{M} p(z_i; \Theta) = \sum_{i=1}^{M} \sum_{l=1}^{n_i} log \sum_{k=1}^{K} \alpha_k p(x_{il}; \theta_k) \tag{3}$$

Θ is complete set of parameters needed to define the complete mixture model, i.e., $\Theta = \{\alpha_1, \ldots, \alpha_k, x_1, \ldots, x_k, \Sigma_1, \ldots, \Sigma_k\}$. However, the maximum likelihood estimate of Θ cannot be computed analytically. The task here is to determine the individual mixture components from the joint density, using the given set of individuals Z.

3.2 Expectation-Maximisation Algorithm

Let us assume the cluster means in the mixture model above follow a polynomial functions of order P. In this mixture regression problem, the objective is to estimate the regression coefficients $\beta_k \; \forall \quad k = 1, \cdots, K$ as well as the cluster membership probability for each individual. EM is the widely used technique

for computing the maximum likelihood estimates for this problem, where the cluster membership probabilities are hidden. In this particular case, where all measurements of an individual are dependent, the EM-variant of [6] can be used for an initial clustering.

3.3 Kalman Filter

An elaborate overview about Kalman filter is presented in [13], we describe it briefly for completeness. Kalman filter addresses the problem of trying to estimate the state $x \in \mathcal{R}^N$ of a discrete-time controlled process that is governed by the following linear-time difference equation:

$$x_s = Ax_{s-1} \tag{4}$$

with a measurement $z \in \mathcal{R}^D$ that is

$$z_s = Hx_s + v_s \tag{5}$$

where A represents the transition matrix, x_s is the state (i.e. the true cluster centre) at time step s. H is the measurement matrix, z_s is the observed measurement (of an individual belonging to this cluster). w_s and v_s are random variables that represent process and measurement noise, respectively. They are mutually independent, white and with normal probability distributions, i.e., $p(w) \sim N(0, Q)$ and $p(v) \sim N(0, R)$.

Kalman filter basically estimate the state vector by using system sensors and measurement data, which are corrupted by noise. This estimation is done by using a type of feedback control: first the filter estimates the state of the process at a given time step s and then obtains feedback in terms of measurements which are assumed to be noisy. Eq. 4 and 5 describe a linear model at time s. Since x_s cannot be measured directly, the information provided by z_s is used to update the x_s.

The a priori state estimate $\hat{x}_s^-$ and covariance error P_k^- are calculated using the following time update equations:

$$\hat{x}_s^- = A\hat{x}_{s-1} + w_{s-1} \tag{6}$$

$$P_s^- = AP_{s-1}A^T + Q \tag{7}$$

The measurement update equations provides the feedback and are responsible for adjusting the model based on the new measurement (i.e., the priori estimates are adjusted to obtain a better posterior estimate):

$$S = HP_s^-H^T + R \tag{8}$$

$$K_s = P_s^-H^TS^{-1} \tag{9}$$

$$\hat{x}_s = \hat{x}_s^- + K_s(z_s - H\hat{x}_s^-) \tag{10}$$

$$P_s = (I - K_sH)P_s^- \tag{11}$$

where I = Identity Matrix of similar dimensions as P_s^-, and K_s is the Kalman gain. Using the measurement, a posterior state estimate $\hat{x}_s$ as well as an error estimate P_s are computed. The time and measurement equations are executed iteratively.

3.4 Kalman Filter Initialisation

Before the Kalman filter outlined above can be used, its parameters defining the state transition model as well as the measurement model must be initialised. Using the clusters' β-coefficients of the mixture regression model learnt by the EM-algorithm of Gaffney and Smyth, the position of each cluster centre at time t can be estimated. The coordinates corresponding to this position in the D-dimensional feature space at time t can be written as a vector $f^{(0)}(t) = f_1^{(0)}(t), \cdots, f_D^{(0)}(t)$, where $f_d^{(0)}(t) = \beta_{d0} + t\beta_{d1} + \cdots + t^o\beta_{do}$.

The true state at this time t comprises these coordinates, but also higher order derivatives of $f^{(0)}(t)$ as *meta-features*, as for example speed and acceleration. Let $f^{(l)}(t)$ denote the l-th derivative of $f^{(0)}(t)$ with respect to t. Assuming that all derivatives of orders greater than o are zero, the true state f at the time t can then be written as:

$$f(t) = \Big(f_1^{(0)}(t), \quad f_2^{(0)}(t), \quad \cdots, \quad f_D^{(0)}(t), \quad f_1^{(1)}(t), \quad f_2^{(1)}(t), \quad \cdots, \quad f_D^{(1)}(t), \\ \cdots, \quad f_1^{(o)}(t), \quad f_2^{(o)}(t), \quad \cdots, \quad f_D^{(o)}(t) \quad \Big)^T$$

The initialisation steps for the Kalman state model are outlined in Function `InitKalman()`. The probability distribution for the initial state x_0 is modelled by a mean vector μ_0 and a covariance matrix Σ_0 (Line 1). The mean vector can be estimated by evaluating $f(t_0)$ at the first point in time t_0. As a covariance matrix an identity matrix is used.

Given the $(s-1)$-th state at time t_s, the successive state at time t_s is defined by the state transition process above to be $z_s = Az_{s-1} + w_{s-1}$. Given an initial state z_0, the state transition matrix A as well as the process noise covariance matrix Q are needed to estimate the successive states.

Let $\Delta = t_s - t_{s-1}$ be the sampling interval, corresponding to the time between two samples are drawn from successive states $s-1$ and s. Using the Taylor series up to o-th order for defining $\delta_s = \frac{\Delta^s}{s!}$, one can write the state-transition matrix $A = [a_{ij}]$, where

Function InitKalman

Input : model $\mathcal{C}_i$, estimates Θ_i

1 Infer initial state using GMM parameters (β). `/* Init of Kalman State Space */`
2 Determine state transition matrix A (given sampling interval Δ).
3 Compute the covariance for the process noise Q.
4 Compute state-sensor matrix H `/* Init of State-Sensor Model */`
5 Compute the covariance for the sensor noise R .

$$a_{i,j} = \begin{cases} \delta_s \; if \quad \exists \quad q \in N: \quad i - D * q - j = 0 \\ 0 \;\; \text{otherwise} \end{cases}$$

For example, the state transition matrix of a 2-dimensional feature space considering up to 3rd order movement is

$$A = \begin{pmatrix} a_0 & 0 & a_1 & 0 & a_2 & 0 & a_3 & 0 \\ 0 & a_0 & 0 & a_1 & 0 & a_2 & 0 & a_3 \\ 0 & 0 & a_0 & 0 & a_1 & 0 & a_2 & 0 \\ 0 & 0 & 0 & a_0 & 0 & a_1 & 0 & a_2 \\ 0 & 0 & 0 & 0 & a_0 & 0 & a_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & a_0 & 0 & a_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_0 \end{pmatrix}$$

with $a_0 = 1$, $a_1 = \Delta$, $a_2 = \frac{\Delta^2}{2}$, $a_3 = \frac{\Delta^3}{6}$.

Assuming a uniform process noise over the feature space, the covariance matrix Q for the process noise can be set equal to the identity matrix, multiplied by a process noise factor $\hat{q}$ (Function `InitKalman()`, Line 3). This parameter $\hat{q}$ has to be tuned using the training data.

As our interest is the update of the regression obtained by the EM-algorithm, we can safely assume no distortion between the true state and the measurement, allowing the measurement matrix H to be set equal to the identity matrix (`InitKalman()`, Line 4). However, there is measurement noise to be considered. In order to obtain the measurement noise covariance matrix R, the covariance in the training data between the state estimates using EM and the observed data points as measurement is calculated (`InitKalman()`, Line 5).

The (estimated) *true* states, including higher order derivatives, can be calculated recursively using the β-coefficients obtained by the EM algorithm, as shown above. However, for the *observed measurement*, only the position $\hat{f}^{(0)}(t)$ is directly available, whereas the derivatives of $\hat{f}^{(0)}(t)$ with respect to t have to be estimated. This is solely possible for individuals with more than one measurement in the training data, as we need $o+1$ measurements per individual to estimate the derivatives of order o. Given three measurements of the $(i-1)$-th derivative $\hat{f}^{(i-1)}$ at times t_{-1}, t_0 and t_1 , respectively, the i-th derivative at time t_0 can be approximated as $\hat{f}^{(i)}(t_0) = \frac{\hat{f}^{(i-1)}(t_2) - \hat{f}^{(i-1)}(t_{-1})}{t_1 - t_{-1}}$. If only two measurements exist, the estimated (i)-th derivative is identical for both points in time, thus it is assumed that the the $(i+1)$-th derivative is zero. The resulting estimation of the (i)-th derivative for the two measurements taken at time points t_0, t_1 is then $\hat{f}^{(i)}(t_0) = \hat{f}^{(i)}(t_1) = \frac{\hat{f}^{(i-1)}(t_1) - \hat{f}^{(i-1)}(t_0)}{t_1 - t_0}$.

Once the difference between the estimated true state and the observed measurements of an individual is calculated, it is used to update the state estimate of a cluster. Two principal approaches can be distinguished: First, the state estimates of all clusters can be updated by using a weight proportional to the cluster membership probability of the instance. In contrast to this *weighted*

Algorithm 2: Tracking Trajectories with Kalman Filter

```
Input : new batch of data Z_i, model C_i, estimates Θ_i
foreach z ∈ Z_i do
    if z belongs to a known individual then
        ẑ ← from the meta-features for z
        Update sensor noise covariance R
        Update state transitions A based on the time elapsed b/w last cluster update δ_t
        Get new estimates for the clusters using the Kalman Filter
    Compute cluster membership probability for z
    Update cluster membership probabilities for the individual.
```

update strategy, a *the-winner-takes-all* update strategy only updates the cluster with the highest cluster membership probability.

While the winner-takes-all strategy is straightforward, the weighted update strategy requires a definition on how weights are used in the Kalman equations. In equation 8 from above the measurement noise is considered by adding its covariance matrix R as an addend. The extent of R is inversely related to the confidence in measurements. By multiplying a factor c with R, the confidence in a measurement can be incorporated into this equation.

$$S = HP_s^- H^T + R * c \tag{12}$$

This factor c could be set to the inverse of the cluster membership probability p, i.e. $c = \frac{1}{p}$. However, the inverse of the squared cluster membership probability, i.e. $c = \frac{1}{p^2}$, can also be used. The motivation is as follows: Assume an observation z does not belong to the cluster which is currently under consideration for update. As the distance of z to the cluster centre increases, the cluster membership probability decreases. Using the inverse of the squared cluster membership probability decreases the weight of very distant points in an over-proportional way, possibly reducing the influence of outliers. However, the effect of the choice of an update strategy is studied further in the experimental evaluation below.

3.5 Update and Clustering

Given an estimated probability distribution of the previous state, the current state on obtaining a new measurement for a cluster can be calculated using equation 10, where for the calculation of the Kalman gain K equation 9 and for the calculation of S equation 12 from above are used.

The individual-to-cluster assignment and the calculation of the higher order derivatives as meta-features can be done as in the calculation of the measurement noise covariance matrix above. This leads to the Algorithm 2. As new data is presented to the algorithm it adapts the model accordingly.

Given a new measurement z of individual i at time t_s and state estimates for time t_{s-1}, the algorithm first determines whether (a) the individual i is *known*, i.e. a prior measurement exists for i or (b) individual i is previously unknown. Only measurements from known individuals are used as sensor input for model update, in order to increase robustness against outliers.

A model update is done by first calculating the difference between sensor input and estimated state position. The sensor input $\hat{z}$ is the combination of the observed position of z with the calculated meta features (Line 3), which can be calculated as described in subsection 3.4 above. The estimated state position is calculated as defined in (Lines 4-5). The new state estimates are then used to re-compute the cluster membership probabilities for the new measurement z (Line 7), by using eq. 1. Lastly, for each cluster the product of the cluster membership probabilities of all measurements of the individual is calculated to obtain the cluster membership probability of the individual (Line 8). For numeric stability reasons, the log of the cluster membership probability can be used. The cluster membership probability of an individual can be updated by a single multiplication (or addition, if its log is used) with the probability of the new measurement. In contrast to EM, each measurement is processed once, which results in an online-behaviour of this Kalman trajectory tracking algorithm.

If the individual i is *new*, i.e., no prior measurements exist, its cluster membership probability is equal to the membership probability of the measurement. This probability can be calculated as explained above (Lines 7-8). An individual is not used for a model update unless further measurements are acquired.

4 Experiments

For the experimental evaluation, the Kalman filter was compared in different settings to the EM algorithm by [6]. To be able to study our algorithms in a controlled environment first, synthetic datasets had to be used. Cluster purity was used as performance measure. In analogy to [10] [page 357, equation 16.1], we define this measure as $purity = \frac{1}{N}\sum_{j=1}^{K} \max_{i=1}^{K} C_{ij}$ Here C is the confusion matrix, and C_{ij} is the number of elements that are in the i-th true cluster which are assigned to the j-th cluster by the algorithm. N is the number of elements in total. This measure is normalised to values in the interval $[0; 1]$, where one is a perfect separation. This normalisation allows averaging over several data sets without scaling issues, which could result when a measure based on the distance between true and predicted cluster states would have been used. However, as purity increases with the number of clusters, the number of clusters between methods should be similar. As the EM algorithm is used in the initialisation of the Kalman filter, the same number of clusters in all solutions is guaranteed.

A Wilcoxon signed rank test was performed to test the statistical significance of differences in clustering quality.

4.1 Data Sets

The data generator[1] used in the creation of the data sets uses a mixture model with K components. For each component a multivariate Gaussian density function is used to generate observations. The component centres are themselves

[1] The data generator and algorithms (implemented in Octave) are available at: `https://bitbucket.org/geos/tracer-trajectory-tracking/overview`

functions of time. For the initial state, the position, speed and acceleration as well as possible higher derivatives are generated at random. Subsequent states are calculates using a state-transition-matrix as described above, and adding random state-transition noise. Furthermore, sudden shift occurs at a given point in time, offsetting the cluster centres by a random vector.

Observations are sampled along this drift path by first calculating the cluster mean at the point in time, and subsequent sampling of observations using the multivariate Gaussian. Each observation is then assigned to an individual of its generating cluster.

The parameters of this data generator are the dimensionality of feature space D, the order of the polynomials used P, the number of clusters K, the number of individuals M, the number of observations N, the extend of the state-transition noise e_w, the strength of the state-to-signal noise e_r and the strength of the sudden shift e_s.

For the experimental evaluation, fifty data sets with five different parameter settings were generated:

1. **Datasets** $A_1, \cdots, A_{10}$: $D = 1$, $K = 3$, $e_w = 25$
2. **Datasets** $B_1, \cdots, B_{10}$: $D = 1$, $K = 3$, $e_w = 5$
3. **Datasets** $C_1, \cdots, C_{10}$: $D = 1$, $K = 3$, $e_w = 1$
4. **Datasets** $D_1, \cdots, D_{10}$: $D = 2$, $K = 3$, $e_w = 5$
5. **Datasets** $E_1, \cdots, E_{10}$: $D = 2$, $K = 3$, $e_w = 1$

$N = 3000$ observations were generated for $M = 1500$ individuals. This results in a very modest expected number of 2 observations per individual.While the first 1000 observations were used for initial training, the subsequent second third of observations was used to evaluate the performance prior to sudden shift. After the twothousandth observation, sudden shift occurred. Thus the last third of observations was used to evaluate the performance in presence of sudden shift.

4.2 Methods

The Expectation Maximisation algorithm was implemented as described in [6]. As all observations of an individual should be taken into account, there is no straightforward extension to an online version of this algorithm, and a batch version was used instead. Therefore, the initial model was fitted on the first one thousand observations. This model was later used for initialising the Kalman filter. For validation, the EM algorithm itself was refitted on all twothousand observations, prior to determining the cluster of any observation in the second batch. Similarly, the model was refitted on all observations before the cluster assignments in the last batch were evaluated.

It should be noted that this behaviour gives some advantage over a Kalman filter, as from the information used in training this would correspond to a Kalman smoother. However, such a comparison is not in the scope of this paper, which aims at a fast online cluster tracking algorithm. Therefore, the Kalman filter described above was initialised on the initial EM model trained solely on the first

Table 2. Cluster purity prior to shift

Data Set	EM	Kalman Tracking					
		K-1	K-2	K-3	K-4	K-5	K-6
A	0.92^{*}	0.83	0.78^{**}	0.83	0.82	0.79	0.81^{**}
	$\pm$ 0.10	$\pm$ 0.11	$\pm$ 0.15	$\pm$ 0.11	$\pm$ 0.13	$\pm$ 0.13	$\pm$ 0.11
B	0.92^{**}	0.73	0.68^{**}	0.72	0.72	0.69^{*}	0.69^{*}
	$\pm$ 0.04	$\pm$ 0.10	$\pm$ 0.10	$\pm$ 0.10	$\pm$ 0.12	$\pm$ 0.12	$\pm$ 0.08
C	0.87^{**}	0.82	0.79	0.81^{*}	0.80	0.78^{*}	0.81^{**}
	$\pm$ 0.13	$\pm$ 0.12	$\pm$ 0.15	$\pm$ 0.12	$\pm$ 0.11	$\pm$ 0.10	$\pm$ 0.12
D	0.98^{**}	0.90	0.81^{*}	0.89	0.89	0.86	0.82^{**}
	$\pm$ 0.03	$\pm$ 0.13	$\pm$ 0.18	$\pm$ 0.13	$\pm$ 0.13	$\pm$ 0.14	$\pm$ 0.17
E	0.96^{**}	0.84	0.79^{**}	0.84	0.84	0.81	0.81^{*}
	$\pm$ 0.11	$\pm$ 0.16	$\pm$ 0.17	$\pm$ 0.17	$\pm$ 0.16	$\pm$ 0.16	$\pm$ 0.15

batch of observations. The subsequent twothousand observations were clustered one-by-one by the Kalman filter.

The performance of both algorithms on the two validation sets is shown in the tables below.

4.3 Results

The results of the experimental evaluation on the 50 datasets is shown in tables 2,3,4, and 5 below. Single stars * indicate significant ($p < 0.05$) differences compared to K-1, and double stars ** denote highly significant ($p < 0.01$) differences.

Table 3. Cluster purity after shift

Data Set	EM	Kalman Tracking					
		K-1	K-2	K-3	K-4	K-5	K-6
A	0.88	0.87	0.73^{**}	0.86	0.84	0.78^{**}	0.81^{**}
	$\pm$ 0.13	$\pm$ 0.09	$\pm$ 0.11	$\pm$ 0.10	$\pm$ 0.13	$\pm$ 0.13	$\pm$ 0.10
B	0.87	0.80	0.75^{**}	0.80	0.76^{*}	0.70^{**}	0.78
	$\pm$ 0.14	$\pm$ 0.13	$\pm$ 0.12	$\pm$ 0.12	$\pm$ 0.13	$\pm$ 0.12	$\pm$ 0.11
C	0.92	0.89	0.84^{*}	0.88	0.86^{*}	0.78^{**}	0.84^{*}
	$\pm$ 0.12	$\pm$ 0.15	$\pm$ 0.17	$\pm$ 0.14	$\pm$ 0.13	$\pm$ 0.11	$\pm$ 0.15
D	0.96	0.96	0.87	0.96	0.94^{*}	0.88^{**}	0.92^{**}
	$\pm$ 0.10	$\pm$ 0.08	$\pm$ 0.14	$\pm$ 0.08	$\pm$ 0.09	$\pm$ 0.12	$\pm$ 0.09
E	0.96^{**}	0.89	0.82^{*}	0.89	0.88	0.83^{*}	0.86^{*}
	$\pm$ 0.10	$\pm$ 0.14	$\pm$ 0.15	$\pm$ 0.14	$\pm$ 0.14	$\pm$ 0.15	$\pm$ 0.13

The experimental evaluation shows that both algorithms are capable of identifying and tracking the cluster structure correctly. Overall, the performance of the Expectation-Maximisation algorithm is better in terms of purity, with a purity of 0.984 compared to the Kalman filter (0.896) before shift, and 0.964 compared to 0.957 after the shift (overall p-value of $p = 0.035$).

Table 4. Algorithm speed prior to shift

Data Set	EM	Kalman Tracking					
		K-1	K-2	K-3	K-4	K-5	K-6
A	75.9**	6.8	6.4**	6.8	6.8	6.8	6.8*
	± 20.6	± 0.0	± 0.0	± 0.0	± 0.0	± 0.1	± 0.1
B	81.8**	6.8	6.5**	6.8	6.9*	6.8	6.9**
	± 34.2	± 0.0	± 0.0	± 0.0	± 0.0	± 0.0	± 0.0
C	87.1**	7.3	7.0	7.1	6.8	6.8	7.2
	± 61.8	± 1.1	± 0.9	± 0.8	± 0.0	± 0.0	± 0.7
D	64.5**	4.1	3.9**	4.2	4.2	4.1	4.1
	± 26.4	± 0.0	± 0.0	± 0.1	± 0.2	± 0.0	± 0.0
E	54.4**	4.2	3.9**	4.2	4.1	4.1	4.2
	± 13.4	± 0.0	± 0.0	± 0.0	± 0.0	± 0.0	± 0.0

Table 5. Algorithm speed after shift

Data Set	EM	Kalman Tracking					
		K-1	K-2	K-3	K-4	K-5	K-6
A	88.2**	7.0	6.5**	7.0**	6.9*	6.9*	7.0
	± 32.7	± 0.1	± 0.0	± 0.0	± 0.1	± 0.1	± 0.1
B	107.3**	7.0	6.5**	7.0	7.0	7.0	7.1*
	± 73.9	± 0.1	± 0.0	± 0.0	± 0.0	± 0.0	± 0.0
C	126.2**	7.1	7.4	7.9	7.2	8.2	7.0**
	± 124.7	± 0.2	± 1.8	± 1.9	± 0.6	± 2.5	± 0.1
D	73.5**	4.2	4.0**	4.3**	4.2*	4.2*	4.2*
	± 23.6	± 0.0	± 0.0	± 0.0	± 0.0	± 0.0	± 0.0
E	81.9**	4.3	4.0**	4.3*	4.3**	4.3**	4.3
	± 44.8	± 0.0	± 0.0	± 0.0	± 0.0	± 0.0	± 0.0

However, the performance is significantly ($p < 0.001$) worse in terms of computational time, with 65.5 compared to 4.1 seconds for the first validation set and 73.5 compared to 4.3 seconds for the second validations set on a quad core system. One reason might be the offline-behaviour of the EM, which makes use of all 2000 (or 3000) observations for model fitting prior to assigning clusters to the new observations. While this increases the clustering quality, it also leads to ten to twenty times higher computational time compared to the Kalman filter.

The experiments have clearly shown that the weighted cluster update strategy performs best for the Kalman filter. The best strategy is to use squared cluster membership probabilities as weights (K-1). Otherwise observations with low cluster membership probabilities can still have a strong influence on the cluster centre, if they are sufficiently far away. Therefore, using the cluster membership probabilities directly as weights (K-3) leads to a small, but significant ($p = 0.04$) reduction in cluster purity (0.893 compared to 0.896 before and 0.9571 compared to 0.9572 after the shift, in average over all data sets). The hard update strategy using a the-winner-takes-it-all approach (K-2) has shown a highly significant

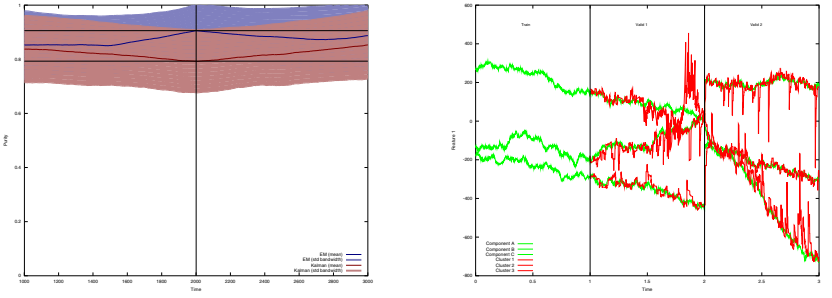

Fig. 1. (a) The development of purity over 30 data sets over time and (b) the distance between true and estimated states for the Kalman filter on one exemplary data set

($p < 0.001$) lower performance (0.808 and 0.869). Further comparisons included a multiplier different from one for the estimated state transition noise e_w (K-4 and K-5), which resulted in worse performance (all results are at least significant with $p < 0.05$). This indicates that the different extend of state transition noise e_w was estimated correctly by the algorithm, making further parameter tuning unnecessary. Finally the meta-features speed and acceleration, which are computed for the movement estimation, were included in the cluster membership probability estimation (K-6). However, this resulted in a highly significant ($p < 0.001$) performance decrease to 0.92 before and 0.82 after shift. In summary, one single parameter setting (K-1) was best on all data sets.

The effect of the sudden shift on the two algorithms is different: The Kalman filter corrects for this shift, and gains from the additional observations over time. This results in a highly significant increase in clustering quality (average purity increases from 0.82 to 0.88, $p < 0.0001$). The EM algorithm is effected differently: A shift constitutes a jump discontinuity in the path of the cluster centre over time, violating the assumed smoothness of this path. The EM algorithm tries to fit a polynomial regression function to the data, which will result in large errors around points of shift. In our experiments, this resulted in a (not significant) decrease in purity from 0.93 to 0.92. This is illustrated in figure 1, where the average purity over a moving window of 1000 observations is plotted over time for both EM and Kalman filter. The horizontal lines indicate cluster purity prior to shift, the vertical line corresponds to the moment shift occurs.

Finally it should be noted that in the experiments only a very modest number of observations per individual was chosen. The results show that both algorithms perform better as more observations per individual become available. Thus the performance in practise can be expected to improve further as the number of available observations increases over time.

5 Conclusions

We studied the problem of clustering trajectories over time when drifts and shifts occur. We proposed TRACER for trajectory cluster tracking in two variants:

an incremental EM algorithm for trajectory clustering and an online tracking algorithm that uses Kalman filter. Our experiments show that both variants of TRACER track clusters over time properly, while the online variant requires significantly less computational time.

Our first experiments have been performed on synthetic data, since we needed insights on the behavior of the algorithms in a controlled environment. We will now experiment with real data on the behavior of individuals (e.g. customers) over time. We further want to improve the robustness of the online TRACER against outliers and to study its behavior in the presence of multiple shifts.

References

1. Buzan, D., Sclaroff, S., Kollios, G.: Extraction and clustering of motion trajectories in video. In: Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004, vol. 2, pp. 521–524 (2004)
2. Cadez, I.V., Gaffney, S., Smyth, P.: A general probabilistic framework for clustering individuals and objects. In: KDD 2000, pp. 140–149. ACM, New York (2000)
3. Chudova, D., Gaffney, S., Mjolsness, E., Smyth, P.: Translation-invariant mixture models for curve clustering. In: KDD 2003, pp. 79–88. ACM, New York (2003)
4. Ellis, D., Sommerlade, E., Reid, I.D.: Modelling pedestrian trajectory patterns with gaussian processes. In: VS 2009, pp. 1229–1234 (2009)
5. Funk, N.: A study of the kalman filter applied to visual tracking. Report (2003)
6. Gaffney, S., Smyth, P.: Trajectory clustering with mixtures of regression models. In: KDD 1999, pp. 63–72. ACM, New York (1999)
7. Han, Y., de Veth, J., Boves, L.: Trajectory clustering for automatic speech recognition (2005)
8. Kalman, R.E.: A New Approach to Linear Filtering and Prediction Problems. Trans. of the ASME – Journal of Basic Engineering 82(series D), 35–45 (1960)
9. Li, X., Wang, K., Wang, W., Li, Y.: A multiple object tracking method using kalman filter. In: IEEE, ICIA 2010, pp. 1862–1866 (2010)
10. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
11. Medeiros, H., Park, J., Kak, A.: Distributed object tracking using a cluster-based kalman filter in wireless camera networks. IEEE TSP 2 (2008)
12. Pathan, S.S., Al-Hamadi, A., Michaelis, B.: OIF - an online inferential framework for multi-object tracking with kalman filter. In: Jiang, X., Petkov, N. (eds.) CAIP 2009. LNCS, vol. 5702, pp. 1087–1095. Springer, Heidelberg (2009)
13. Welch, G., Bishop, G.: An introduction to the kalman filter. Tech Report (1995)
14. Xiong, G., Feng, C., Ji, L.: Dynamical gaussian mixture model for tracking elliptical living objects. Pattern Recognition Letters 27, 838–842 (2006), doi:10.1016/j.patrec.2005.11.015