Trajectory Clustering with Mixtures of Regression Models

Technical Report No. 99–15
Department of Information and Computer Science
University of California, Irvine

Scott Gaffney and Padhraic Smyth
Department of Information and Computer Science
University of California, Irvine
CA 92697-3425
[sgaffney,smyth@ics.uci.edu]

March 1999

Abstract

In this paper we address the problem of clustering trajectories, namely sets of short sequences of data measured as a function of a dependent variable such as time. Examples include storm path trajectories, longitudinal data such as drug therapy response, functional expression data in computational biology, and movements of objects or individuals in video sequences. Our clustering algorithm is based on a principled method for probabilistic modelling of a set of trajectories as individual sequences of points generated from a finite mixture model consisting of regression model components. Unsupervised learning is carried out using maximum likelihood principles. Specifically, the EM algorithm is used to cope with the hidden data problem (i.e., the cluster memberships). We also develop generalizations of the method to handle non-parametric (kernel) regression components as well as multi-dimensional outputs. Simulation results comparing our method with other clustering methods such as K-means and Gaussian mixtures are presented as well as experimental results on real data sets.

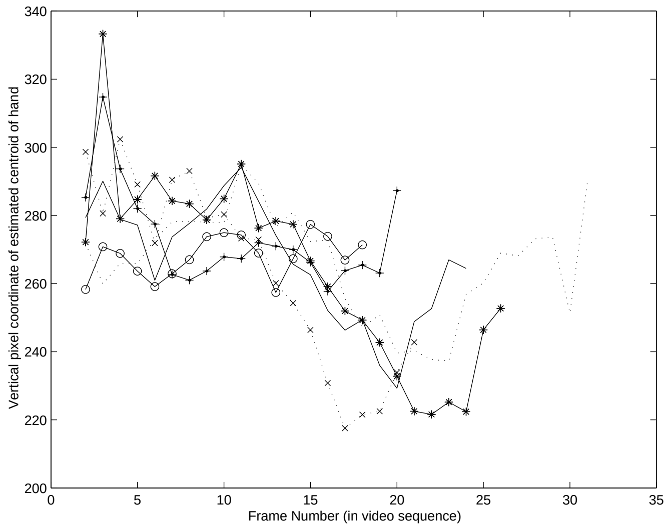


Figure 1: Trajectories of the estimated vertical position of a moving hand as a function of time, estimated from 6 different video sequences.

1 Introduction

In this paper we investigate the problem of clustering sets of measurements Y which are measured as a function of an independent variable x. Typically the x variable represents time and we have data on M different individuals, where for each individual we have measurements of a response variable y (possibly multi-dimensional) over time. Figure 1 shows an example of such data for a set of 6 different y measurements. The x-axis represents time and the y-axis is the vertical location in pixel coordinates (relative to a fixed coordinate frame), giving the centroid of a person's hand as estimated from a sequence of images. Each curve represents the (noisy) estimated trajectory of a particular individual performing a particular simple hand movement.

We investigate the problem of clustering such trajectory data into distinct groups. This type of data can arise in a variety of applications where repeated measurements are available on individual "objects" over time, e.g., response curves in drug therapy monitoring, experimental gene expression data in protein modeling, individual responses to stimuli in animal behavior experiments, and so forth.

Note that there are some complications which arise for data such as that in Figure 1 which make it difficult to apply standard clustering techniques. For example, the trajectories are of different lengths, and thus, one cannot simply convert the trajectories to fixed-length vectors and apply a clustering technique such as the K-means algorithm in a fixed-dimensional space. We will return to these issues and to the video data set in more detail later in the paper.

In statistics, this type of data is often referred to as *longitudinal* data or *repeated measures* data. Even though such data could be thought of as time-series data for each individual, it is typically the case that the data records per individual are too short to be amenable to conventional time series modeling techniques (i.e., typically one may only have on the order of 10 measurements per individual). For example, a simple longitudinal data set might consist of 240 measurements giving the weight of twenty different rabbits, weighed once a month, for a year. Datasets of this type are very common in medical research, where the individuals being measured are separated into several different groups, and the difference in "group behavior" is statistically analyzed. The most common example of this type of analysis is realized through the use of the so called treatment and control groups. Methods and procedures for dealing with these types of analyses are thoroughly discussed in Jones (1993). In a slightly more general context the term *functional data* (Ramsay and Silverman (1997)) is also used to describe these data sets, where in this case x need not necessarily be a time index and it is explicitly assumed that y's are a smooth function of the x's.

In this paper we will use the term *trajectory data* as a general term to refer to this type of data, emphasizing the notion that the data for each individual is assumed to be a smooth trajectory in y space as a function of an independent variable x. We are interested in being given trajectory data and determining if the data can be naturally clustered into groups. In traditional approaches to modeling trajectory data it is assumed that any group or cluster structure on the data is known a priori. For example, Jones (1993) describes a common technique used in longitudinal data analysis in which a line is fit to the data within each group using simple linear regression, and then the regression coefficients are used as summary measures to compare the difference in group behavior. We are not aware of any prior published work on probabilistic or statistical clustering methods for trajectory data.

The paper is organized as follows. In Section 2 we discuss the difficulties with using standard clustering techniques for such data, motivate the use of probabilistic model-based clustering, and discuss related prior work. In Section 3 we propose a generative probabilistic framework for trajectory data, namely that each cluster can be modeled in a regression manner as a smooth function of x with additive noise, and each cluster has a different such function associated with it. Section 4 illustrates how the expectation-maximization (EM) algorithm can be used to learn the cluster parameters. Extensions to non-parametric regression functions (using kernel regression for example) are also presented, as well as the generalization to the case where $\mathbf{y}$ is a multi-dimensional measurement. In Section 5 we present a systematic analysis of EM-based trajectory clustering on simulated data, with comparisons to alternative vector-based methods. Section 6 illustrates how the method can be used to learn clusters of hand trajectories from video data, in a completely unsupervised manner. Section 7 contains a brief discussion of various extensions of the general approach and Section 8 presents general conclusions.

2 Background and Related Work

An obvious way that one might go about clustering trajectory data is to take all of the n_j measurements for an individual and form a vector $\mathbf{y}_j$ of dimension n_j. Assume for the moment that each individual has the same number of measurements (i.e., $n_j = n$ for all individuals j) and these measurements were all taken at exactly the same x values. We can then treat the set of $\mathbf{y}_j$ trajectories as a set of n-dimensional vectors in an n-dimensional space and use any of a variety of the many clustering methods which operate in vector-spaces.

While this may be a reasonable approach in some applications, it will not always be applicable or appropriate. For many data sets, the trajectories will be of different lengths and may be measured at different time points. In addition, the $\mathbf{y}$ measurements may be multidimensional (e.g., 3d position estimates in tracking the dynamics of a moving object), in which case there is no natural vector representation.

Perhaps more fundamentally, if one converts the data to a vector representation there is a fundamental loss of information about the data, i.e., if we believe from the underlying physics of the data-generating process that the $\mathbf{y}$'s are a smooth function of the x's, then this smoothness information is lost when we convert a sequence of n numbers to an n-dimensional vector of numbers. Thus, intuitively, retaining the notion of trajectory smoothness in our clustering procedure, should generate better data models compared to throwing away this information.

Thus, we will investigate *model-based* clustering of trajectories, where each cluster will be modeled as a prototype function with some variability around that prototype. A distinct feature of this model-based approach to clustering is the fact that it produces a descriptive interpretable model for each cluster. Since we are estimating smooth functions from noisy data it will be natural to use a probabilistic framework. Specifically we will use mixtures of regression models as the basis for clustering.

Mixtures of multivariate (vector-based) distributions and densities have been widely used for probabilistic clustering in the past (e.g., McLachlan and Basford (1988), Banfield and Raftery (1993), Cheeseman and Stutz (1996), Smyth et al (1997), Fraley and Raftery (1998), and Smyth, Ide, and Ghil (in press)). However, none of this work extends directly to trajectory clustering unless one uses a vector representation for the data. The work of Stanford and Raftery (1997) on mixture models for finding clusters in two-dimensional spatial point patterns is similar in spirit to the approach proposed here. However, the problem of clustering trajectory data is somewhat different to that of two-dimensional curves, since there is an explicit dependence on an independent variable x present in the trajectory case.

The "mixtures of experts" models popular in neural networks research, as originally proposed by Jordan and Jacobs (1994), is mathematically quite similar to our cluster models (in fact, in mixtures of experts the weights for the clusters are allowed to vary smoothly, which is a more flexible model than what we will propose here). However, there are also some distinct differences between mixtures of experts and our trajectory mixtures. In mixtures of experts, the emphasis is on prediction rather than clustering, and there is no explicit focus on obtaining clusters. Furthermore, for mixtures of experts there is an input space and an output space of fixed dimensionalities, and one is learning a mapping from one to the other. There is no notion of having *sets* of trajectories, of possibly different lengths and measured at different points. Thus, although mathematically similar, the focus of trajectory clustering and the focus of mixtures of experts models are quite different.

3 A Generative Mixture Model for Clusters of Trajectories

3.1 A Brief Review of Standard Mixture Model Clustering

In probabilistic clustering we assume that the data are being generated in the following "generative" manner:

- An individual is drawn randomly from the population of interest.
- The individual has been assigned to cluster k with probability w_k, $\sum_{k=1}^{K} w_k = 1$. These are the "prior" weights on the K clusters. We will implicitly assume that K is fixed in this paper, but in general we may also wish to learn K from the data.

- Given that an individual belongs to cluster k, there is a density function $f_k(y_j|\theta_k)$ which generates observed data y_j for individual j.

From this generative model, it follows that the observed density on the y's must be a mixture model, i.e., a linear combination of the component models:

$$P(y_j|\theta) = \sum_k^K f_k(y_j|\theta_k) w_k, \tag{1}$$

Thus, if we observe the y_j's, and we assume a particular functional form for the f_k components, we can try to *estimate* from the data what the most likely values of the parameters θ_k and the weights w_k are. The EM algorithm (e.g., McLachlan and Krishnan, 1997) is a general procedure for finding the maximum likelihood estimates of the parameters of a mixture model.

In practice, if each $\mathbf{y}_j$ is a multivariate vector, it is common to use simple functional forms for the component models, e.g., multivariate Gaussian bumps. Under the Gaussian assumption, the EM algorithm is used to find the means (locations) and covariances ("shapes") of the Gaussian bumps in the vector space where the $\mathbf{y}$'s are measured. The component models (as estimated from the data) can then be treated as the clusters, and Bayes' rule can be used to determine the probability of membership in the learned clusters for any data point $\mathbf{y}_j$.

A feature of the mixture model approach is that it allows uncertainty in cluster memberships, or equivalently, overlap in the cluster models. In addition, it can effectively "learn" the appropriate distance metric in the $\mathbf{y}$ space for each cluster (for Gaussians for example this will be the Mahalonobis distance defined by the covariance and mean of each component), rather than relying on any predefined (and possibly inappropriate) notion of distance. Finally, the probabilistic framework allows one to address issues such as finding the best number of clusters in a principled and relatively objective manner (e.g., see Fraley and Raftery, 1998).

3.2 Mixtures of Regression Models

We can straightforwardly generalize the multivariate mixture model in the last section (Eq. (1)) to define *mixtures of regression models*, where we have measurements y which are a function of some known x. Each component now is a *conditional* density function of the form $f_k(y|x,\theta_k)$. Assume for now that y and x are each 1-dimensional. Typically we will assume a standard regression relationship between y and x, e.g.,

$$y = g_k(x) + e, \tag{2}$$

where e is zero-mean Gaussian with standard deviation σ_k, and $g_k(x)$ is a deterministic function of x. Thus, in the case of Gaussian noise (which we assume henceforth in this paper) we have that the conditional density $f_k(y|x,\theta_k)$, given that y belongs to the kth group, has mean $g_k(x)$ and standard deviation σ_k. Here θ_k includes both the parameters of the model $g_k(x)$ and the noise deviation σ_k. The noise model e could easily be generalized to be dependent on x, rather than being constant. Such a generalization may be appropriate in specific situations and can be handled in the mixture model in a straightforward fashion. For simplicity of notation, however, we will assume a constant noise term in this paper.

We are now ready to define a probabilistic cluster model for sets of trajectories. Let our data set $\mathcal{S}$ consist of n_j measurements for each of M individuals, $1 \leq j \leq M$. We will refer to these measurements as being a function of time (i.e. x is synonymous with time), although this is not strictly necessary. Let the trajectory of measurements for the jth individual be denoted as y_j, with the ith measurement of y_j denoted as $y_j(i)$. Furthermore, suppose that the trajectory of measurements y_j were taken at the times in x_j. Finally, let each trajectory in $\mathcal{S}$ belongs to one of K groups.

The probability of observing a particular measurement $y_j(i)$, given $x_j(i)$ and component model k, is defined as $f_k(y_j(i)|x_j(i), \theta_k)$, and is assumed to be a conditional regression model as discussed above. We can then define the probability of a complete trajectory, given a particular component model k as

$$P(y_j|x_j, \theta_k) = P(y_j(1), \ldots, y_j(n_j)|x_j(1), \ldots, x_j(n_j), \theta_k) = \prod_i^{n_j} f_k(y_j(i)|x_j(i), \theta_k). \tag{3}$$

Here we make the standard regression assumption that, conditioned on the model and the x values (and, thus, the means for the y's are known), the noise is independent at different x points along the trajectory. Dependent noise could be modeled if appropriate for a particular application.

When we don't know which component generated that trajectory (as is the case in practice for clustering), the conditional density of the observed data $P(y_j|x_j)$ is a mixture density:

$$P(y_j|x_j, \theta) = \sum_k^K f_k(y_j|x_j, \theta_k) w_k, \tag{4}$$

where $f_k(y_j|x_j, \theta_k)$ are the mixture components, w_k are the mixing weights, and θ_k is the set of parameters for component k.

Conditional independence between trajectories, given the model, amounts to assuming that our individuals constitute a random sample from a population of individuals, and allows the full joint density to be written as:

$$P(Y|X, \theta) = \prod_j^M \sum_k^K w_k \prod_i^{n_j} f_k(y_j(i)|x_j(i), \theta_k). \tag{5}$$

The log-likelihood of the parameters θ given the data set $\mathcal{S}$ can be defined directly from Eq. (5).

$$\mathcal{L}(\theta|\mathcal{S}) = \sum_j^M \log \sum_k^K w_k \prod_i^{n_j} f_k(y_j(i)|x_j(i), \theta_k). \tag{6}$$

4 The EM Algorithm for Mixtures of Regression Models

The task at hand is to pull the mixture components out of the joint density, using $\mathcal{S}$ as a guide, so that the underlying group behavior can be discovered. The problem would be simple if it was known to which group each trajectory belonged. Given the group membership of each trajectory, and assuming some particular form for the density functions f_k (e.g., linear regression models with Gaussian noise), the K models can simply be fit to the grouped data. If, however, the group memberships are hidden, as is the case in practice, more complex procedures are required.

A common approach for dealing with hidden data is to employ the EM algorithm (Dempster, Laird, and Rubin, 1977; McLachlan and Krishnan, 1997). Realizing that if we knew the hidden data, the problem usually becomes a set of much simpler problems (just as above), it makes sense to estimate the hidden data, work out the out the answers to the simpler problems, and then re-estimate the hidden data again using the current answers that we just computed. This process is then repeated until some stabilization occurs. The EM framework gives us a consistent way to estimate the hidden data so that $\mathcal{L}(\theta|\mathcal{S})$ is guaranteed to never decrease.

In Eq. (6), the hidden data corresponds to the unknown group membership for each of the M trajectories. Let $\mathbf{Z}$ be a matrix of indicator vectors $\mathbf{z}_j = (z_{j1}, \ldots, z_{jK})$, such that $z_{jk} = 1$ for some k, and $z_{jt} = 0, \forall\, t \neq k$. So, if $z_{jk} = 1$, then the jth trajectory is said to be *generated* from the kth mixture component. The joint density of Y and $\mathbf{Z}$ given X can be defined as follows.

$$\begin{aligned} P(Y, \mathbf{Z}|X, \theta) &= P(Y|\mathbf{Z}, X, \theta)p(\mathbf{Z}|X, \theta) \\ &= \prod_j^M P(y_j|\mathbf{z}_j, x_j, \theta)p(\mathbf{z}_j) \\ &= \prod_j^M \prod_k^K [f_k(y_j|x_j, \theta_k)w_k]^{z_{jk}} \\ &= \prod_j^M \prod_k^K \left[w_k \prod_i^{n_j} f_k(y_j(i)|x_j(i), \theta_k) \right]^{z_{jk}}. \end{aligned} \tag{7}$$

This follows from our previous conditional independence assumptions on y_j and $y_j(i)$, and the fact that our $\mathbf{z}_j$'s are indepenent. The augmented log-likelihood function (also referred to as the *complete-data likelihood*) follows directly from Eq. (7):

$$\mathcal{L}(\theta|\mathcal{S}, \mathbf{Z}) = \sum_j^M \sum_k^K z_{jk} \log w_k + \sum_j^M \sum_k^K \sum_i^{n_j} z_{jk} \log f_k(y_j(i)|x_j(i), \theta_k). \tag{8}$$

The EM algorithm consists of two steps: (1) the expected value of Eq. (8) is taken with respect to $p(\mathbf{Z}|Y, X, \theta^{t-1})$, where θ^{t-1} is a current set of parameters, and (2) this expectation is maximized over the parameters θ to yield the new parameters θ^t. For the particular form of $\mathbf{Z}$ chosen here, the expectation of $\mathcal{L}(\theta|\mathcal{S}, \mathbf{Z})$ is

$$E[\mathcal{L}(\theta|\mathcal{S}, \mathbf{Z})] = \sum_j^M \sum_k^K h_{jk} \log w_k + \sum_j^M \sum_k^K \sum_i^{n_j} h_{jk} \log f_k(y_j(i)|x_j(i), \theta_k), \tag{9}$$

where

$$\begin{aligned} h_{jk} &= E[z_{jk}] \\ &= p(z_{jk} = 1|y_j, x_j, \theta^{t-1}) \\ &\propto P(y_j|z_{jk} = 1, x_j, \theta^{t-1})p(z_{jk} = 1) \\ &\propto w_k \prod_i^{n_j} f_k(y_j(i)|x_j(i), \theta^{t-1}). \end{aligned} \tag{10}$$

The h_{jk} can be thought of as *soft* ($0 \leq h_{jk} \leq 1$) indicator variables, and the z_{jk} can be thought of as *hard* ($z_{jk} \in \{0, 1\}$) indicator variables. That is, h_{jk} corresponds to the posterior probability that trajectory y_j was generated by component k. Note that all of the measurements for trajectory j share this *membership probability.*

In Eq. (2), we defined the regression equation for y such that the expected value of y was equal to $g(x)$, a linear function of x. We adapt our notation to fit our data $\mathcal{S}$ into this framework by defining the regression equation as follows.

$$\mathbf{Y}_j = \mathbf{X}_j\boldsymbol{\beta}_k + \mathbf{e}_k, \tag{11}$$

with $\mathbf{Y}_j = [1 \quad y_j(1) \quad \cdots \quad y_j(n_j)]'$, and

$$\mathbf{X}_j = \begin{bmatrix} 1 & x_j(1) & x_j(1)^2 & \cdots & x_j(1)^p \\ 1 & x_j(2) & x_j(2)^2 & \cdots & x_j(2)^p \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & x_j(n_j) & x_j(n_j)^2 & \cdots & x_j(n_j)^p \end{bmatrix}.$$

In other words, $\mathbf{Y}_j$ is a column vector formed from the measurements of the jth trajectory, and $\mathbf{X}_j$ is an n_j by $p+1$ matrix whose second column contains the *times* corresponding to the measurements in $\mathbf{Y}_j$ (p gives the order of the regression model). We assume that $\mathbf{e}_k$ is a vector of size n_j consisting of zero-mean Gaussians with variance σ_k^2, and that $\boldsymbol{\beta}_k = [\beta_{k0} \quad \beta_{k1} \quad \cdots \quad \beta_{kp}]'$ is a vector of regression coefficients. By specifying our mixture components as regression models defined by Eq. (11), we are setting $f_k(y_j|x_j, \theta_k)$ equal to a Gaussian with mean $\mathbf{X}_j\boldsymbol{\beta}_k$ and covariance matrix diag(σ_k^2).

The maximization of Eq. (9) with respect to the parameters $\theta_k = \{w_k, \boldsymbol{\beta}_k, \sigma_k^2\}$ is straightforward. In fact, the solutions for $\boldsymbol{\beta}_k$ and σ_k^2 are exactly those obtained from the well known weighted least squares problem (Draper & Smith, 1981).

The solutions for the regression coefficients $\hat{\boldsymbol{\beta}}_k$, the variance terms $\hat{\sigma}_k^2$, and the mixing weights $\hat{w}_k$ are given below.

$$\hat{\boldsymbol{\beta}}_k = (\mathbf{X}'\mathbf{H}_k\mathbf{X})^{-1}\mathbf{X}'\mathbf{H}_k\mathbf{Y} \tag{12}$$

$$\hat{\sigma}_k^2 = \frac{(\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}}_k)'\mathbf{H}_k(\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}}_k)}{\sum_j^M h_{jk}} \tag{13}$$

$$\hat{w}_k = \frac{1}{M}\sum_j^M h_{jk} \tag{14}$$

Above, we let $\mathbf{H}_k = \text{diag}([\mathbf{h}_{1k}^* \quad \mathbf{h}_{2k}^* \quad \cdots \quad \mathbf{h}_{Mk}^*])$, with $\mathbf{h}_{jk}^* = [h_{jk}^{(1)} \quad h_{jk}^{(2)} \quad \cdots \quad h_{jk}^{(n_j)}]$. That is, $\mathbf{h}_{jk}^*$ is a row vector consisting of n_j copies of the membership probability h_{jk}.

If we let $N = \sum_j^M n_j$, then $\mathbf{H}_k$ is an N by N diagonal matrix whose elements on its main diagonal represent the *weights* to be applied to $\mathbf{Y}$ and $\mathbf{X}$ during regression. The *weights*, in this case, are the membership probabilities for each of the trajectories. Thus, they determine how much of an impact the jth trajectory has on the kth regression. We also have $\mathbf{Y} = [\mathbf{Y}_1' \quad \mathbf{Y}_2' \quad \cdots \quad \mathbf{Y}_M']'$, and $\mathbf{X} = [\mathbf{X}_1' \quad \mathbf{X}_2' \quad \cdots \quad \mathbf{X}_M']'$. In other words, $\mathbf{Y}$ is an N by 1 matrix containing all the $y_j(i)$ measurements, one trajectory after another, and $\mathbf{X}$ is an N by $p+1$ matrix whose second column gives the *time points* where the corresponding $\mathbf{Y}$ values were measured.

Intuitively, the estimate $\hat{\sigma}_k^2$ is a kind of *weighted residual* resulting from the regression in the transformed weighted-space, and $\hat{w}_k$ is the average proportion of trajectories contributing to the kth regression. These equations yield the following EM algorithm for mixtures of linear regression models:

EM Algorithm for Mixtures of Linear Regression Models

1. Randomly initialize the membership probabilities h_{jk}.
2. Calculate new estimates for $\hat{\boldsymbol{\beta}}_k$, $\hat{\sigma_k^2}$, and $\hat{w_k}$ from the weighted least squares solutions, using the current membership probabilities as weights.
3. Compute the new membership probabilities using Eq. (10) and the newly computed parameter estimates from the previous step.
4. Loop to step 2 until the log-likelihood stabilizes.

4.1 Extensions to Non-Parametric Regression Models and Multivariate Y Measurements

A interesting extension to the framework developed in Section 4 is to model the density functions $f_k(y_j(i)|x_j(i), \theta)$ as non-parametric regression models. These types of models can be used to relax the assumptions placed on the form of the regression function. This approach is inherently more data-driven. Non-parametric function estimation has been studied in a number of different settings, for example, kernel smoothing (Wand and Jones, 1995), local polynomial modelling (Fan and Gijbels, 1996), and density estimation (Silverman, 1986).

In our context, by modelling our component densities as non-parametric regression models, we may be able to cluster many types of trajectory data for which the general relationship between y and x is uncertain, or for when we do not wish to make any such assumptions on our regression functions. In this paper, we experiment with the use of kernel regression model components for our densities $f_k(\cdot)$.

The basic idea behind kernel regression is that we can approximate any arbitrary function with a series of simple locally-weighted functions, such as linear regression functions. Therefore, we will approximate the unknown function at a point x_0, by running a locally-weighted linear regression (of order p) about the point x_0, and report the prediction $\hat{y}$ as the height of this fit. The weights are produced by a symmetric kernel (e.g., standard Gaussian density) centered about the point x_0, whose purpose is to *down-weight* points far away from x_0. When the random component for the locally fit regression model is Gaussian, the solution for the regression coefficients can be calculated using weighted least squares.

For our purposes, this means that if we include kernel regression model components into our mixtures of regressions framework, then all we need to modify in our previous algorithm specification is step 2. Instead of requiring that we calculate $\hat{\boldsymbol{\beta}}_k$ and $\hat{\sigma_k^2}$, we require the calculation of the mean $\hat{y_j(i)}$, or predicted value, and variance $\hat{\sigma_{ki}^2}$, at every point $x_j(i)$, by solving a locally-weighted least squares problem (the weights become the posterior probabilities multiplied by the kernel weights at each point). With these values (and, of course $\hat{w_k}$), we can proceed to step 3, to calculate our new membership probabilities.

The only other consideration, here, is the *bandwidth* for the kernels. The bandwidth is essentially a dispersion parameter. It determines how spread out the density for a kernel will be. Much has been written on the subject of learning this parameter from the data, for example (Fan and Gijbels, 1996). In this paper we will just assume a known fixed bandwidth (since we conjecture that the clustering will not be particularly sensitive to bandwidth), but clearly one could generalize our algorithms to include a "data-adaptive bandwidth" component.

One further extension to the above framework that we have developed so far, is to include the handling for multivariate $\mathbf{y}_j$ measurements (or outputs). For example, suppose we have trajectory data that measures the movement of a particle in two-dimensional space, over time. In this case, our regression equation will look like the following.

$$[y_j^{(1)}(i) \ \ y_j^{(2)}(i)] = [1 \ \ x_j(i)] \begin{bmatrix} \beta_{k0}^{(1)} & \beta_{k0}^{(2)} \\ \beta_{k1}^{(1)} & \beta_{k1}^{(2)} \end{bmatrix} + [e_k^{(1)} \ \ e_k^{(2)}], \tag{15}$$

where $\mathbf{e}_k$ is zero-mean multivariate Gaussian with covariance matrix Σ_k. In other words, now $\mathbf{y}_j$ is a multidimensional trajectory, and correspondingly, the density $f_k(\mathbf{y}_j(i)|x_j(i), \theta_k)$ will be multivariate Gaussian.

The steps in our EM algorithm do not change for the multidimensional measurements case, except that in Eq. (10), we replace the univariate density with the above multivariate density, and we calculate full covariance matrices during Step 2.

5 Experiments with Simulated Data

This section describes clustering experiments performed using mixtures of regression models, or more simply *regression mixtures*, using the EM framework described above. For comparison purposes, regression mixtures performance is compared with that of standard Gaussian mixtures, and K-means.

Because there are differences in the types of data that the three clustering methods can handle, a number of restrictions were adopted for proper comparison purposes. First, as pointed out in Section 2, in order for the use of Gaussian mixtures or K-means to be applicable, we must restrict our trajectories to be of the same lengths, and the trajectories must be measured at the same X values (or time points). Second, the multivariate density components for Gaussian mixture models can, in general, be described by a multi-dimensional mean vector $\boldsymbol{\mu}$ and a covariance matrix Σ. However, we previously focused our discussion on regression model components whose variance can be described by a single variance parameter σ^2, and so we will do so for the components in the Gaussian mixture models also. That is, the covariance matrices shall be restricted to be of the form $\sigma^2\mathrm{I}$, where I is the identity matrix. There is another reason why one may want to restrict the covariance matrices to be of this form. Some trajectory data sets contain M trajectories each of length n, where $M < n$. In this case, the data matrix D to be used with Gaussian mixtures is of size M by n, and the covariance matrices are of size n by n. However, $\Sigma = D'D/M$, and thus we have that $\mathrm{rank}(\Sigma) \leq \mathrm{rank}(D) \leq M$. Therefore, Σ cannot be of full rank, and thus will not have an inverse. We will see a similar type of trajectory data set in Section 6.

The data sets used for experiments described in this section were generated from K different polynomials by evaluating these polynomials at n different points (or *times*) in X, and adding some Gaussian noise to each of these values. For each polynomial, l different trajectories were *sampled* from it, giving a total of $M = lK$ trajectories in each generated data set. The task is to cluster the M trajectories into K groups that correspond to the different underlying polynomials (or clusters/classes).

Figure 2 shows one such generated data set and a few iterations of our developed EM algorithm for regression mixtures applied to this data. The data set shown in Figure 2 was sampled from the three underlying polynomials (three clusters): $y = 120 + 4x$,

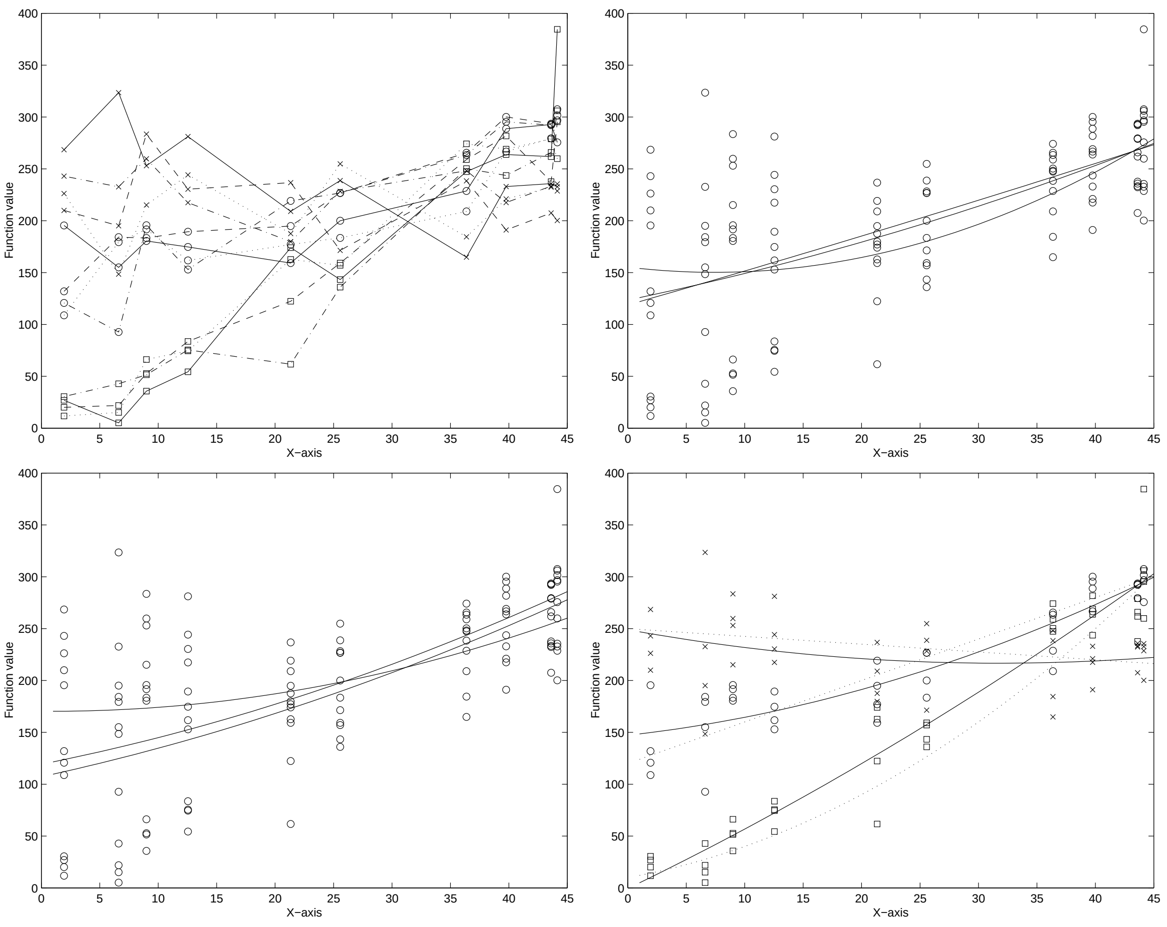


Figure 2: Trace of the EM algorithm as applied to a linear regression mixture model at various iterations. The upper left plot shows some of the original trajectories, the upper right shows the initial locations of the 3 cluster trajectories for EM, lower left shows the locations after 1 iteration of EM, and lower right shows the cluster locations (solid) after EM convergence, as well as the locations of the true data-generating trajectories (dotted).

$y = 10 + 2x + 0.1x^2$, and $y = 250 - 0.75x$. The number of trajectories sampled from each underlying *true* function is $l = 4$, and the length of each trajectory is $n = 10$. In the upper left graph of Figure 2 the plotting symbols (square, circle, and ex) for each trajectory represent its class label (i.e., which polynomial it was generated from), and the line styles are used to differentiate between different trajectories within a single class. The upper-right graph shows the same data (with the class labels and lines removed for clarity) along with the random starting points (solid lines) for each of the three *true* models. Initially, each sequence is randomly assigned to a cluster (more accurately, each sequence is assigned with uncertainty to each cluster through the use of membership probabilites, or weights on the sequences) and weighted least squares is run to get the three inital estimates. Each estimate, represented by a regression line, was obtained assuming a second-order polynomial regression model. The lower-left graph shows the regression lines obtained after iteration 1, and the lower-right graph shows the final regression lines as output by our algorithm. In addition, the lower-right graph shows the *true* models as dotted lines as well as the learned clustering for the trajectory data, shown by the plotting symbols. The cluster-

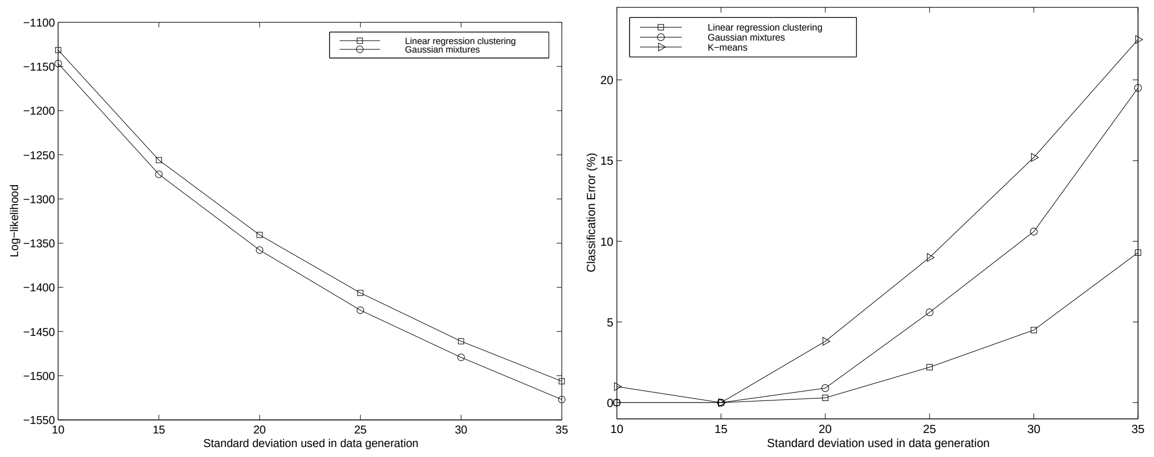


Figure 3: Mean log-likelihood and classification error rate performance on test data as the noise level increases from $\sigma = 10$ to $\sigma = 35$. ($l = 10, n = 15$).

ing/classification is perfect in this case, i.e., all trajectories are assigned to the true clusters which generated them.

In order to demonstrate the effectiveness of the linear regression mixture model, several different comparison tests are presented here. The experimental results described below were collected as follows. Two polynomials were selected to represent the mean behavior for two different clusters of Gaussian perturbed trajectories: $y = 200 + 1.7x$, and $y = 200 + 0.7x$. Fifty different randomly generated training sets and test sets were created by randomly choosing some X-values and adding gaussian noise to the function values evaluated at these points. Using these data sets, each of the three clustering techniques (K-means, Gaussian mixtures, and linear regression mixtures) were applied to the sets in order to assess performance based on log-likelihood scores and classification error rate tests (note that K-means is not a probabilistic model, and thus will not have log-likelihood scores). Finally, these tests were repeated over eight different noise levels, and the mean values for the attained log-likelihood scores, and the classification error rates on the test data are reported.

In the left graph of Figure 3, we see the log-likelihood scores for both linear regression mixtures, and Gaussian mixtures on test data. The training data in these tests each contained $l = 10$ trajectory samples from each class (i.e., from each polynomial), with each trajectory having length of $n = 15$. The noise level was increased from $\sigma = 10$ to $\sigma = 35$ along the x-axis. In the graph, we see that the linear regression mixture model attains a higher likelihood score, on the test sets, at every noise level.

The right-hand graph of Figure 3 compares the clustering (or classification) effectiveness for each of the three clustering algorithms on the same data as above. In the graph, we see that the linear regression mixture model classifies (clusters) the test data with less error, at every noise level, than the other approaches.

In most tests, the Gaussian mixture model returns a higher log-likelihood on the training data, than the regression mixture model does. This appears to be an overfitting effect. The Gaussian mixtures model exhibits this behavior on this type of data because it treats the trajectory data as if it were simple vector data, and thus cannot use the trajectory information to guide it in the learning process. The natural incorporation of this trajectory information into the regression mixture model is one of its distinct advantages over vector-

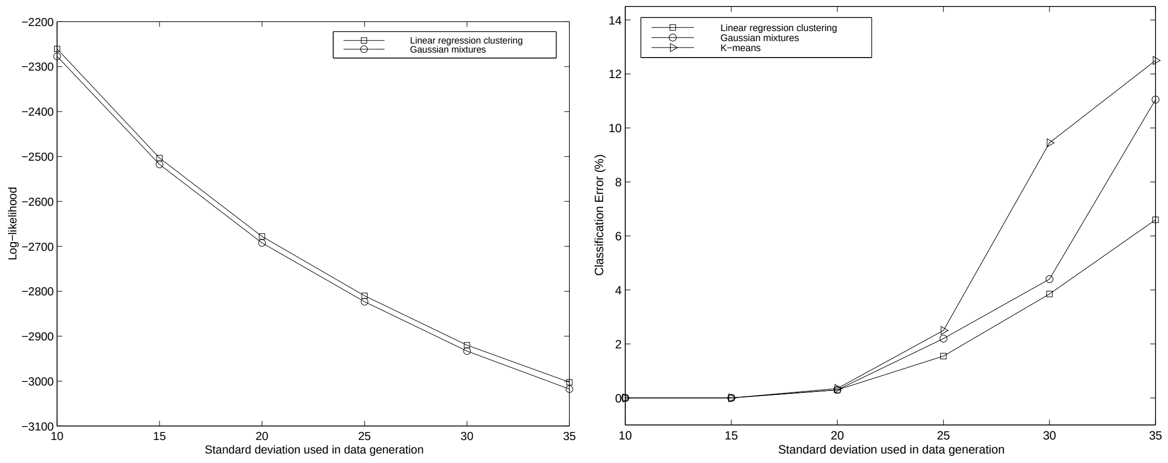


Figure 4: Mean log-likelihood and classification error rate performance on test data as the noise level increases from $\sigma = 10$ to $\sigma = 35$. ($l = 20$ and $n = 15$).

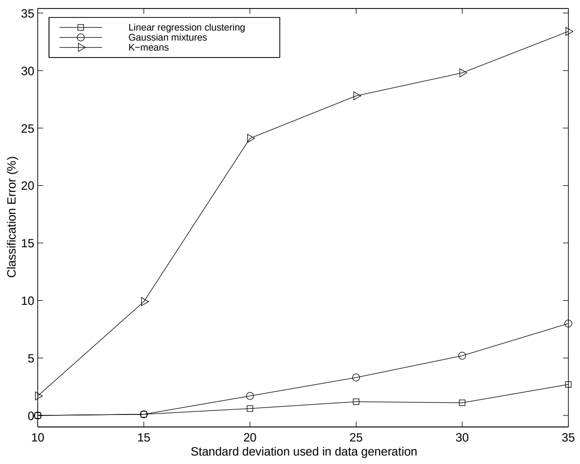
based clustering approaches.

If the number of trajectories sampled for each underlying *true* model is increased, then we would expect that the loss of the trajectory information incurred by Gaussian mixtures would be somewhat offset by the increased data information. The results shown in Figure 4 demonstrate this type of behavior for our data. In these graphs, the number of trajectories sampled from each model was increased from $l = 10$ to $l = 20$ (i.e., the total number of trajectories in the data was increased from 20 to 40), while all else remained as for earlier tests. The graphs, indeed, show that the performance of Gaussian mixtures more closely resembles that of regression mixtures when the data information is increased.

Another way in which we can increase the data information useful for clustering is to sample the trajectories with different levels of noise for each class. In the left graph of Figure 5, the noise level for trajectories sampled from the first model was initially set to $\sigma_1 = 10$, while the noise level for trajectories sampled from the second model was initially set to $\sigma_2 = 20$ (again, $l = 10$ and $n = 10$). While it is clear from the graph that the performance for both mixture modelling techniques has increased, it still seems that the regression mixture model outperforms the Gaussian mixture model proportionately so. In the right-hand graph of Figure 5, we initially set $\sigma_1 = 10$ and $\sigma_2 = 30$. Even though the difference in variance between classes is so large that the classification/clustering is almost perfect, there is still a slight advantage that the linear regression mixture model enjoys.

6 Clustering Trajectories in Video Streams

In this section, we apply both the linear regression mixture model, and the kernel regression mixture model, to the problem of clustering sequences of images. Our data set consists of 20 video streams depicting 5 different hand movements made by an actor: (1) an upward movement (Up), (2) a downward movement (Down), (3) a left-to-right movement (Left-Right), (4) a right-to-left movement (Right-Left), and (5) a diagonal movement acting from the bottom-left to the top-right of the frame (bLeft-tRight). All movement directions are from the perspective of the actor. There are 4 samples (video streams) for each movement. Figure 6 displays some sample images from these streams.



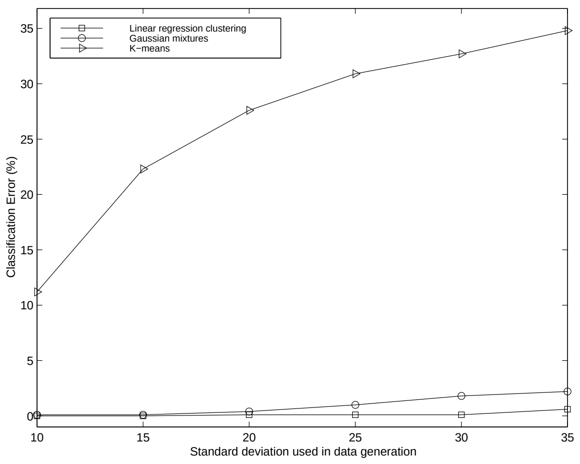


Figure 5: Mean classification error rate performance measured on test data with different noise levels per class. In the left graph, σ_1 increases from 10 to 35, and σ_2 increases from 20 to 45. In the right-hand graph. σ_1 increases from 10 to 35, and σ_2 increases from 30 to 55. Note that the x-axis tracks σ_1; however, both σ_1 and σ_2 are increasing by the same amount at each point. ($l = 10$ and $n = 15$).

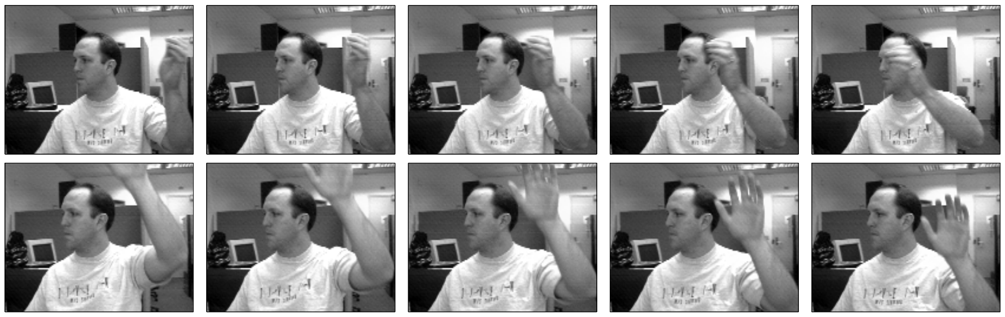


Figure 6: Sample frames from our video stream data, depicting hand movements made by an actor. The top row of images show a portion of a *left-to-right* movement, while the bottom row of images show a portion of a *down* movement.

From each video stream is produced a *two dimensional* trajectory, in pixel coordinates, measured over time (or frames). The trajectory crudely follows the movement of the hand by tracking the centroid of the *image difference* between frames.Because each of the trajectories are of different lengths (different videos of the movements have different numbers of frames), vector-based clustering such as K-means or Gaussian mixtures cannot be applied directly here.

We attempt to cluster the 20 video streams into 5 groups, based on our estimated trajectory data. The data is clustered using both a linear regression mixture model, and a kernel regression mixture model (fitting locally-weighted linear regressions of order 1). The trajectory data is input to our algorithms without scaling or registering them in any way. Since the trajectory data is two dimensional, we will regress two dimensional output vectors $\mathbf{y}_j(i) = [y_j(i)^{(1)}\ y_j(i)^{(2)}]$ on a univariate $x_j(i)$ representing time (frame number), and our density on $\mathbf{y}_j$ will be the multivariate gaussian with mean $g_k(x_j)$ and covariance matrix Σ_k (note that, in this case, Σ_k is a 2 by 2 matrix).

The top graph in Figure 7 gives the resulting clustering from running linear regression

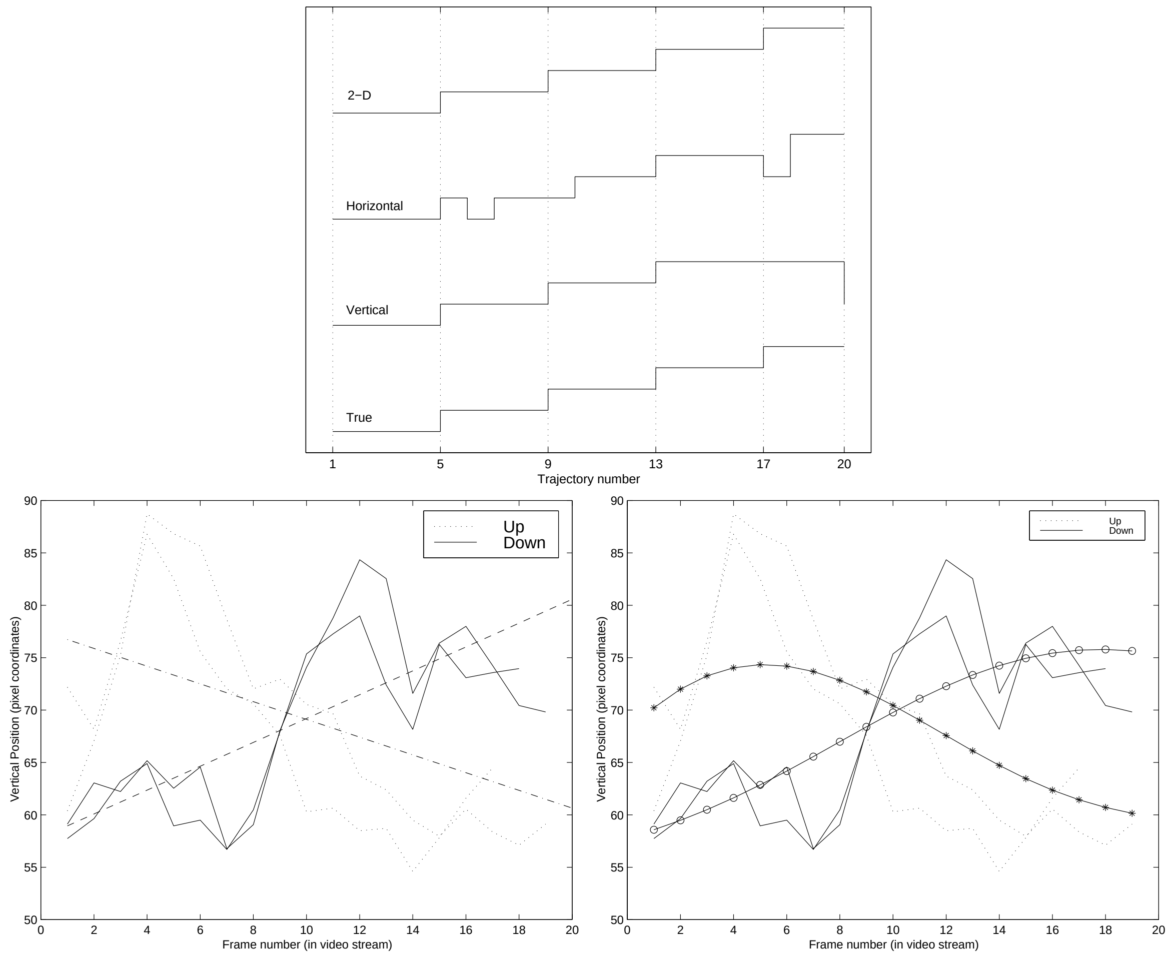


Figure 7: The top graph shows clustering/classification comparisons based on regressing only on the *vertical* trajectories, the *horizontal* trajectories, or on both trajectories ($2D$). The line labeled *True* gives the *pattern* of the true clustering. Only the $2D$ regression achieves the correct clustering. The bottom graphs show mean-behavior lines as estimated by linear regression mixtures and kernel regression mixtures.

mixtures on the video data. The line marked *True*, in the graph, represents the true "pattern" of clustering for the video data. It shows that trajectories 1-4 are in group one, 5-8 are in group two, 9-12 are in group three, 13-16 are in group four, and 17-20 are in group five. The lines marked *Vertical* and *Horizontal* show the patterns of clustering when our algorithm is only allowed to look at one of the dimensions of the data at a time. It is clear that these patterns do not match the *True* clustering. However, the line marked $2D$ shows the pattern of clustering if our algorithm is allowed to regress the full 2-dimensional *output* trajectories $\mathbf{y}_j$ on the univariate x_j. In this case, the true pattern of clustering is found, and we find value in the multi-dimensional extension to the problem. The same sort of caricature can be seen when kernel regression components are inserted into the EM algorithm.

The lower-left graph in Figure 7 displays two of the four example *Up* trajectories with dotted lines, and two of the four example *Down* trajectories with solid lines. The dashed-dotted line represents the weighted-regression line returned from the EM algorithm for the

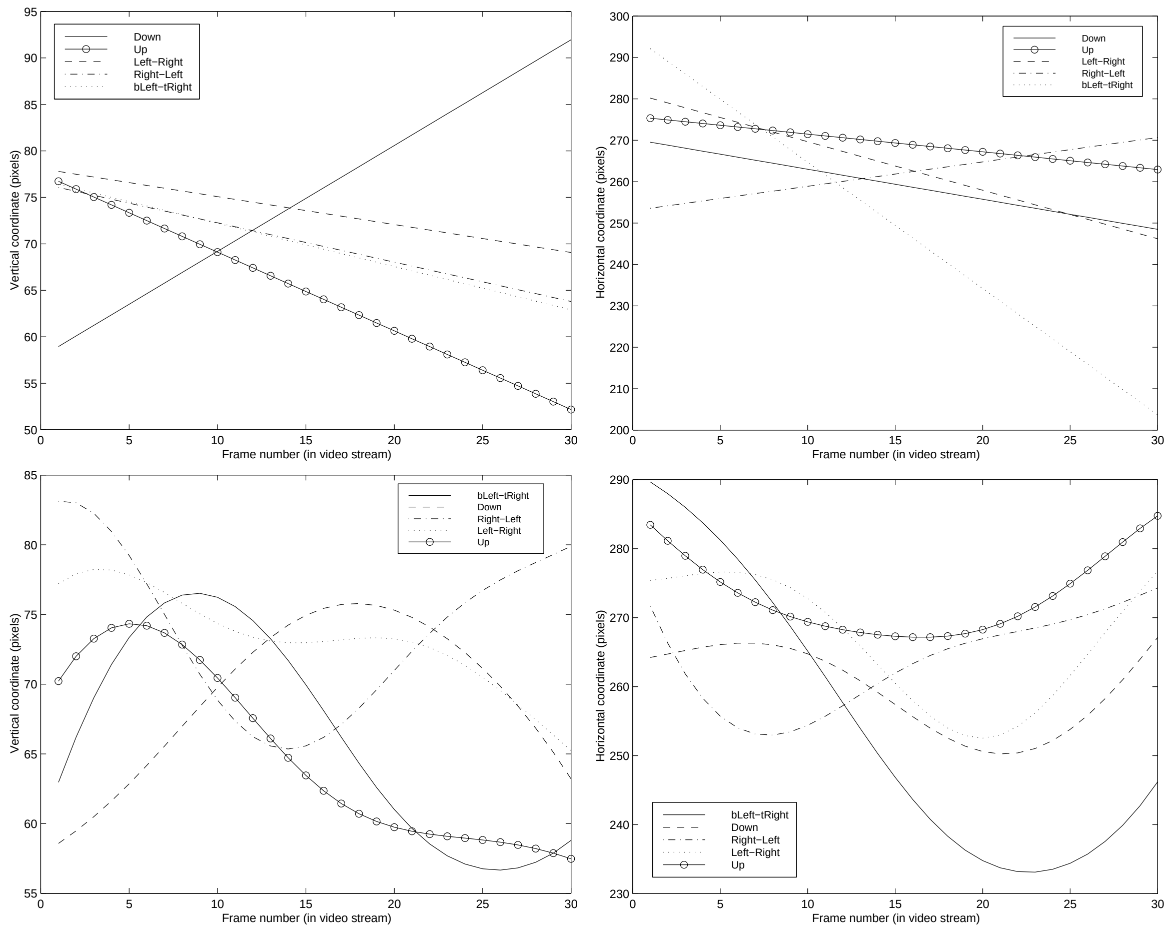


Figure 8: Estimated component model lines as returned from our EM algorithm for mixtures of kernel regression models.

Up class (movement), when (2-dimensional output) linear regression components are used, and the dashed line represents the same line for the *Down* class (movement). The *inverse* relationship that can be seen in the trajectory data is exactly what the EM algorithm found through the use of the weighted-linear fits. The lower-right graph depicts the same scene, except where (2-dimensional output) kernel regression components were used in the EM algorithm. We can see that the same features are picked up during the clustering, albeit with more degrees of freedom.

The top half of Figure 8 shows each of the weighted-regression lines returned by our EM algorithm while employing (two-dimensional output) linear regression components. The lines in the left graph describe the mean-movement behavior, per class, with respect to vertical pixel position. Notice, that the *Up* and *Down* lines behave inversely (as they should, since up is the inverse movement of down), and that the *Left-Right* and *Right-Left* lines behave similar to each other (as they should, since they are similar movements as far as the vertical dimension is concerned). The *bLeft-tRight* line shows some similarity to all lines, except the *Down* line, which is what we expect.

The top-right graph depicts the same mean-movement behavior as above, except that here we see the mean behaviors with respect to the horizontal dimension. The same sort of relationships as noted above, can be seen in this graph also. For example, the *Left-Right* and *Right-Left* lines are now inverse, as they should be.

The bottom half of Figure 8 mimics the previous two graphs, but depicts the (two-dimensional output) kernel regression mixtures case. We can see the same types of inverse relationships in these graphs, as for the previous, though the behaviors seem more separated. For example, the differences between the *bLeft-tRight* and *Right-Left* mean-vertical behaviors can be more easily discerned in the kernel regression case, than for the linear regression case.

7 Discussion and Future Work

The probabilistic framework allows for a variety of extensions which were not discussed in this paper due to space limitations. In particular, the number of clusters and the functional form of the component models can in principle be determined automatically using penalized likelihood or cross-validated likelihood. There is of course a rather large search space and an interesting direction for future work is how to perform efficient heuristic search over the space of such models. Another direction for generalization is to allow linear shifts and scaling of the trajectories such as replacing $g(x)$ by $g(ax+b)$ where a and b are scaling and translation parameters specific to each trajectory which are estimated from the data.

Another generalization of this approach is to *couple* the parameters for each trajectory to a prior density on parameters in a hierarchical modeling framework. It can be shown that the method proposed in this paper (as well as other earlier methods for clustering sequences using mixtures of Markov models) is a special case of a more general hierarchical framework (Cadez and Smyth, 1999). In particular, constraining all of the individuals in a cluster to have the same parameters is equivalent to assuming a delta function prior in a hierarchical framework. By relaxing this assumption, one can still couple the regression parameters both within and between clusters, but allow for a more flexible clustering "language," e.g., the slopes of the lines in a particular cluster are constrained to be quite similar (a narrow prior) but the intercepts are allowed to vary quite substantially (a fairly broad prior). These priors are not conventional Bayesian priors which need not be specified ahead of time, but instead can be estimated from the data using EM (as in an empirical Bayesian framework).

8 Conclusions

In this paper we investigated the problem of clustering trajectory data. Traditional vector-based clustering algorithms are inadequate in many cases for these types of data sets. We introduced a probabilistic mixture regression model for such data and showed how the EM algorithm could be used to cluster trajectories. The model-based assumption can be relaxed to allow for non-parametric regression components and the technique can also be easily extended to handle multivariate trajectories. We demonstrated the utility of the approach on both simulated and real data sets; the method is seen to outperform the more naive k-means and Gaussian mixture models.

Acknowledgments

The work described in this paper was supported by the National Science Foundation under Grant IRI-9703120. The authors gratefully acknowledge the contributions of Steve Cody for providing the video data.

References

Banfield, J. D., and A. E. Raftery, Model-based Gaussian and non-Gaussian clustering, *Biometrics*, 49, 803–821, Sept. 1993.

Cadez, I. and Smyth, P., 'Probabilistic clustering using hierarchical models,' submitted for publication, 1999.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977) 'Maximum likelihood from incomplete data via the EM algorithm,' *J. Royal Stat. Soc. B*, 39(1), 1–38.

Draper, N. R., and Smith H. (1981). *Applied Regression Analysis*, New York: John Wiley and Sons, 2nd ed.

C. Fraley and A. E. Raftery, 'How many clusters? Which clustering method? Answers via Model-based cluster analysis,' *Computer Journal*, 41, 578–588, 1998.

Jones, R.H. (1993). *Longitudinal Data with Serial Correlation: A State Space Approach*, New York: Chapman & Hall.

M. I. Jordan and R. A. Jacobs, 'Hierarchical mixtures of experts and the EM algorithm,' *Neural Computation*, 6, 181-214, 1994.

Cheeseman, P. and Stutz. J., 'Bayesian classification (AutoClass): theory and results,' in *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (eds.), Cambridge, MA: AAAI/MIT Press, pp. 153–180, 1996.

McLachlan, G. J. and K. E. Basford, *Mixture Models: Inference and Applications to Clustering*, New York: Marcel Dekker, 1988.

McLachlan, G. J. and Krishnan, T., *The EM Algorithm and Extensions*, New York: John Wiley and Sons, 1997.

Ramsay, J. O. and Silverman, B. W. (1997). *Functional Data Analysis*, New York: Springer-Verlag.

Silverman, B.W. (1986). *Density Estimation*, New York: Chapman & Hall.

Smyth, P., M. Ghil, K. Ide, J. Roden, and A Fraser, 'Detecting atmospheric regimes using cross-validated clustering,' *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA: AAAI Press, 61–66, 1997.

Smyth, P., M. Ghil, and K. Ide, 'Multiple regimes in Northern hemisphere height fields via mixture model clustering,' *Journal of Atmospheric Science*, in press.

Stanford, D. and Raftery, A. E., 'Principal curve clustering with noise,' Technical Report No. 317, Department of Statistics, University of Washington, February 1997.

Wand, M.P. and Jones, M.C. (1995). *Kernel Smoothing*, New York: Chapman & Hall.