

Trajectory Clustering with Mixtures of Regression Models

Scott Gaffney and Padhraic Smyth
Department of Information and Computer Science
University of California, Irvine
{sgaffney, smyth}@ics.uci.edu
<http://www.ics.uci.edu/~datalab>

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.

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

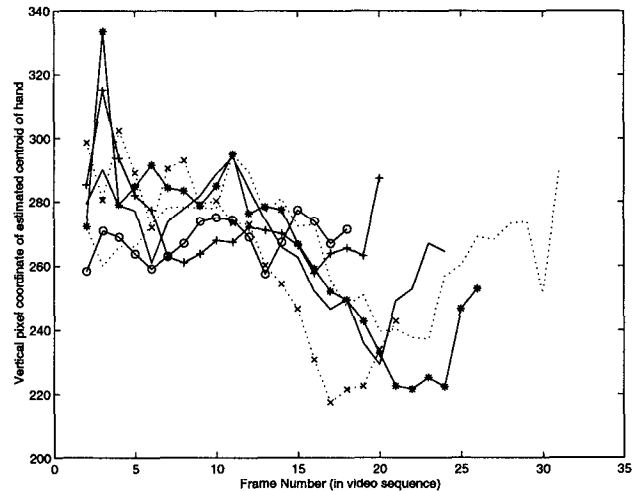


Figure 1: Trajectories of the estimated vertical position of a moving hand as a function of time, estimated from 6 different video sequences.

a particular individual performing a particular simple hand movement.

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. Trajectory 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. We are currently investigating the problem of clustering storm trajectories in the atmosphere over the Pacific and Atlantic oceans. Identification of such clusters is important to atmospheric scientists who model and predict storm behavior. Storms are currently clustered

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD-99 San Diego CA USA

Copyright ACM 1999 1-58113-143-7/99/08...\$5.00

using vector-based clustering methods such as K -means which (as we discuss below) require that all trajectories be of the same length.

An obvious way that one might go about clustering trajectory data is to take all of the n_j measurements for an object or individual and form a vector 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 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 y measurements may be multi-dimensional (e.g., 3d position estimates in tracking the dynamics of a moving object), in which case there is no natural vector representation. One could resort to various *ad hoc* techniques to try to force the data into a particular representation (e.g., concatenate each of the dimensions together in one long string), but a better approach would handle the natural dimensionality of the data directly.

By converting the data to a vector representation there is a fundamental loss of information. If we believe from the underlying physics of the data-generating process that the 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 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.

2 Related Work

What we describe in this paper as trajectory data is quite similar to what is known in statistics as *longitudinal* data or *repeated measures* data (e.g., [6]). In a slightly more general context the term *functional data* [10] 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. Although 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, thus requiring specialized approaches. Typically repeated measures and functional data analysis methods focus on problems where group (cluster) memberships of the trajectories are known a priori. In contrast we focus here on discovering the groups directly from the data, i.e., clustering.

Probabilistic clustering using vector representations is well established (e.g., [8], [4], and [12]). 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 [14] 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 [7], is also mathematically quite similar to our cluster models. However, in mixtures of experts, the emphasis is on prediction rather than clustering and there is no explicit focus on obtaining clusters. Furthermore, 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 the work presented here and the focus of mixtures of experts models are quite different.

3 A Generative Mixture Model for Clusters of Trajectories

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

1. An individual is drawn randomly from the population of interest.
2. 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.
3. 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: $P(y_j|\theta) = \sum_{k=1}^K f_k(y_j|\theta_k)w_k$.

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., [1]) is a general procedure for finding the maximum likelihood estimates of the parameters of a mixture model.

The probabilistic framework allows one to address issues such as finding the best number of clusters in a principled and relatively objective manner. For example, one can use penalized likelihood [4]) or cross-validated likelihood [13] to determine the most likely value for the number of components in the mixture model given the data.

We can straightforwardly generalize the multivariate mixture model above (Eq. (3)) 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.,

$$y = g_k(x) + e, \quad (1)$$

where e is zero-mean Gaussian with standard deviation σ_k , and $g_k(x)$ is a deterministic linear function of x (linear in the parameters). Thus, in the case of Gaussian noise we have that the conditional density $f_k(y|x, \theta_k)$, given that y belongs to the k th 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 .

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, although this is not strictly necessary. Let the trajectory of measurements for the j th individual be denoted as y_j , with the i th 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 .

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 density of a complete trajectory, given a particular component model k as

$$\begin{aligned} P(y_j|x_j, \theta_k) &= P(y_j(1), \dots, y_j(n_j)|x_j(1), \dots, x_j(n_j), \theta_k) \\ &= \prod_{i=1}^{n_j} f_k(y_j(i)|x_j(i), \theta_k). \end{aligned}$$

Here we make the standard regression assumption that, conditioned on the model and the x values, 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, the conditional density of the observed data $P(y_j|x_j)$ is a mixture density:

$$P(y_j|x_j, \theta) = \sum_k^K f_k(y_j|x_j, \theta_k) w_k, \quad (2)$$

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:

$$P(Y|X, \theta) = \prod_j^M \sum_k^K w_k \prod_i^{n_j} f_k(y_j(i)|x_j(i), \theta_k). \quad (3)$$

The log-likelihood of the parameters θ given the data set \mathcal{S} can be defined directly from Eq. (3).

$$\mathcal{L}(\theta|\mathcal{S}) = \sum_j^M \log \sum_k^K w_k \prod_i^{n_j} f_k(y_j(i)|x_j(i), \theta_k). \quad (4)$$

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. A common approach for dealing with hidden data is to employ the EM algorithm [1], [9]. Realizing that if we knew the hidden data, the problem usually becomes a set of much simpler problems, it makes sense to estimate the hidden data, work out the answers to the simpler problems, and then re-estimate the hidden data using our newly found answers. 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. This process is then repeated, in theory, until we reach a *local* maximum in the log-likelihood. However, when used in practice, one normally applies some stopping criterion to halt the iterations (e.g., when the marginal change in log-likelihood falls below some threshold).

In Eq. (4), the hidden data corresponds to the unknown group membership for each of the M trajectories. Let \mathbf{Z} be a matrix of indicator vectors $\mathbf{z}_j = (z_{j1}, \dots, z_{jK})$, such that $z_{jk} = 1$ for some k , and $z_{jt} = 0, \forall t \neq k$. So, if $z_{jk} = 1$, then the j th trajectory is said to be *generated* from the k th mixture component. The joint density of Y and \mathbf{Z} given X can be defined as

follows.

$$\begin{aligned}
P(Y, \mathbf{Z}|X, \theta) &= P(Y|\mathbf{Z}, X, \theta)p(\mathbf{Z}|X, \theta) \\
&= \prod_j^M P(y_j|\mathbf{z}_j, x_j, \theta)p(\mathbf{z}_j) \\
&= \prod_j^M \prod_k^K [f_k(y_j|x_j, \theta_k)w_k]^{z_{jk}} \\
&= \prod_j^M \prod_k^K \left[w_k \prod_i^{n_j} f_k(y_j(i)|x_j(i), \theta_k) \right]^{z_{jk}} \quad (5)
\end{aligned}$$

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 independent. The augmented log-likelihood function (also referred to as the *complete-data likelihood*) follows directly from Eq. (5):

$$\begin{aligned}
\mathcal{L}(\theta|\mathcal{S}, \mathbf{Z}) &= \sum_j^M \sum_k^K z_{jk} \log w_k \\
&+ \sum_j^M \sum_k^K \sum_i^{n_j} z_{jk} \log f_k(y_j(i)|x_j(i), \theta_k). \quad (6)
\end{aligned}$$

The EM algorithm consists of two steps: (1) the expected value of Eq. (6) 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

$$\begin{aligned}
E[\mathcal{L}(\theta|\mathcal{S}, \mathbf{Z})] &= \sum_j^M \sum_k^K h_{jk} \log w_k \\
&+ \sum_j^M \sum_k^K \sum_i^{n_j} h_{jk} \log f_k(y_j(i)|x_j(i), \theta_k), \quad (7)
\end{aligned}$$

where

$$\begin{aligned}
h_{jk} &= E[z_{jk}] \\
&= p(z_{jk} = 1|y_j, x_j, \theta^{t-1}) \\
&\propto P(y_j|z_{jk} = 1, x_j, \theta^{t-1})p(z_{jk} = 1) \\
&\propto w_k \prod_i^{n_j} f_k(y_j(i)|x_j(i), \theta^{t-1}). \quad (8)
\end{aligned}$$

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. (1), 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.

$$\mathbf{Y}_j = \mathbf{X}_j \boldsymbol{\beta}_k + \mathbf{e}_k, \quad (9)$$

with $\mathbf{Y}_j = [1 \ y_j(1) \ \cdots \ y_j(n_j)]'$, 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 j th 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 $\boldsymbol{\beta}_k = [\beta_{k0} \ \beta_{k1} \ \cdots \ \beta_{kp}]'$ is a vector of regression coefficients. By specifying our mixture components as regression models defined by Eq. (9), 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 $\sigma_k^2 \mathbf{I}$.

The maximization of Eq. (7) 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 [2].

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.

$$\hat{\boldsymbol{\beta}}_k = (\mathbf{X}' \mathbf{H}_k \mathbf{X})^{-1} \mathbf{X}' \mathbf{H}_k \mathbf{Y} \quad (10)$$

$$\hat{\sigma}_k^2 = \frac{(\mathbf{Y} - \mathbf{X} \hat{\boldsymbol{\beta}}_k)' \mathbf{H}_k (\mathbf{Y} - \mathbf{X} \hat{\boldsymbol{\beta}}_k)}{\sum_j^M h_{jk}} \quad (11)$$

$$\hat{w}_k = \frac{1}{M} \sum_j^M h_{jk} \quad (12)$$

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

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 j th trajectory has on the k th regression. We also have $\mathbf{Y} = [\mathbf{Y}'_1 \ \mathbf{Y}'_2 \ \cdots \ \mathbf{Y}'_M]'$,

and $\mathbf{X} = [\mathbf{X}'_1 \ \mathbf{X}'_2 \ \dots \ \mathbf{X}'_M]'$. 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 form 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 k th 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 $\hat{\beta}_k$, $\hat{\sigma}_k^2$, 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. (8) and the newly computed parameter estimates from the previous step.
 4. Loop to step 2 until stopping criterion is met.
-

4.1 Extensions to Non-Parametric Regression Models and Multivariate \mathbf{Y} Measurements

A useful 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 [15], local polynomial modelling [3], and density estimation [11].

In our context, by modelling our component densities as non-parametric regression models, we can cluster 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 $\hat{\beta}_k$ and $\hat{\sigma}_k^2$, we require the calculation of the mean $y_j(i)$, 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 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 [3]. In this paper we will just assume a known fixed bandwidth but clearly one could generalize our algorithms to include a “data-adaptive bandwidth” component. The complexity of our EM algorithm is $O(NKI)$, where we have N data points in total, K clusters, and perform I iterations (I is often less than 10 or so). When one makes use of kernel regression models, however, the complexity can scale as $O(LNKI)$, where L is the number of unique x values in the data set, since we must perform a separate weighted-least squares regression at each of the L x points using (potentially) all of the N y points.

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.

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

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. (8), we replace the univariate density with the above multivariate density, and we calculate full covariance matrices during Step 2.

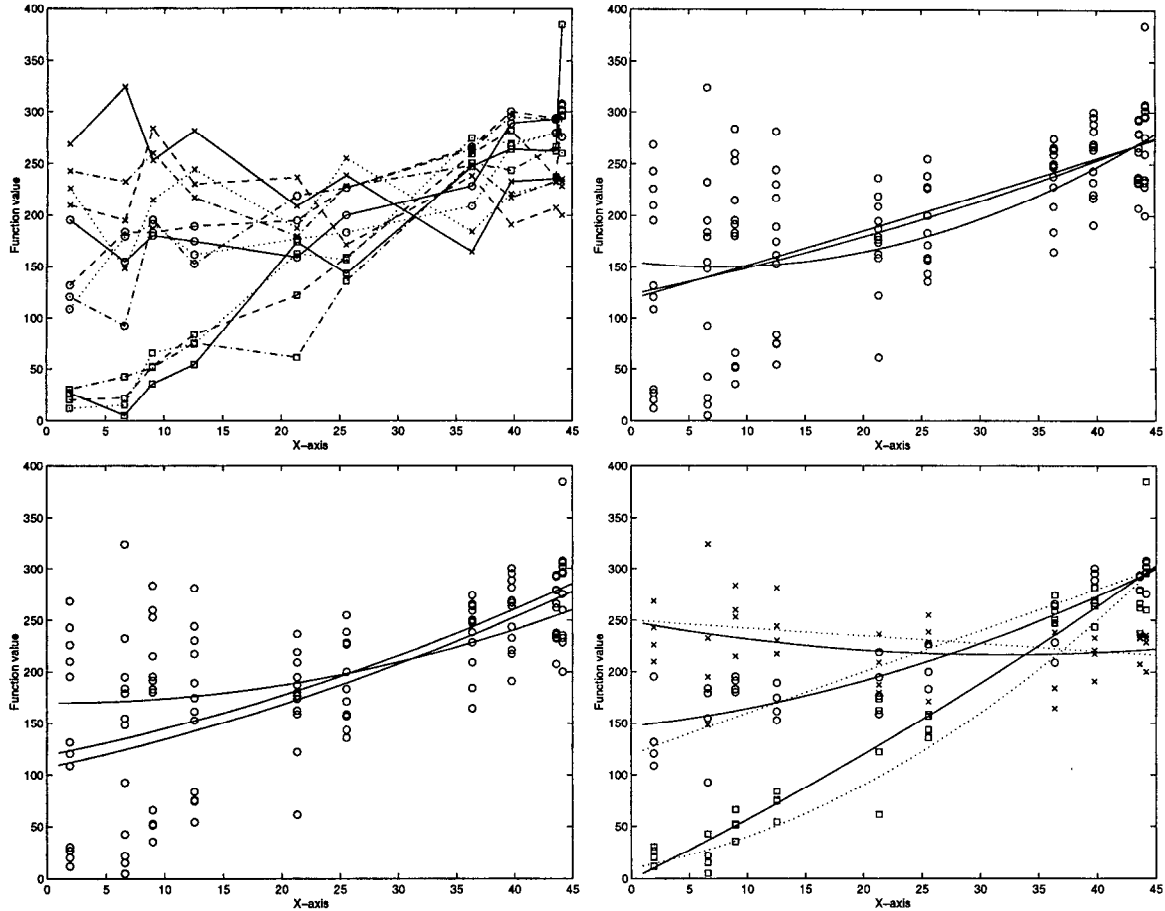


Figure 2: Trace of the EM algorithm as applied to a linear regression mixture model at various iterations. The upper left plot shows all of the original trajectories, the upper right shows the initial locations of the 3 cluster trajectories for EM, lower left shows the locations after 1 iteration of EM, and lower right shows the cluster locations (solid) after EM convergence (iteration 4), as well as the locations of the true data-generating trajectories (dotted).

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. Due to space limitations we only present a small portion of the tests that were conducted.

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 μ 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 $\sigma^2 I$, where I is the identity matrix.

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

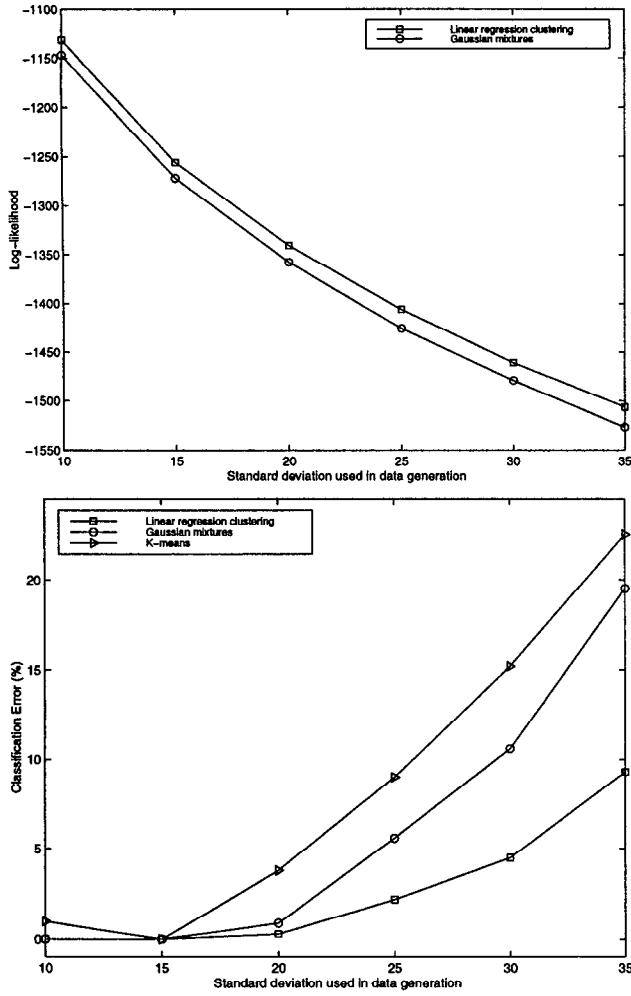


Figure 3: Mean log-likelihood and classification error rate performance on test data as the noise level increases from $\sigma = 10$ to $\sigma = 35$. ($l = 10, n = 15$).

clusters/classes).

Figure 2 shows one such generated data set and a few iterations of our developed EM algorithm applied to this data. The data set shown in Figure 2 was sampled from the three underlying polynomials (three clusters): $y = 120 + 4x$, $y = 10 + 2x + 0.1x^2$, 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 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 representative random starting points (solid lines) for each of the three *true* models. Initially, a proportion of each sequence is randomly

assigned to each cluster and weighted least squares is run to get the three initial 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 (iteration 4). 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 clustering/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 top 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 bottom 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 (or smoothness) 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-based clustering approaches. In [5] we describe further experiments in which the number of trajectories and noise-level per class was varied: once again, the mixtures of regression models universally outperformed the vector-based approaches.

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, and show that the extension to multidimensional trajectories is useful. 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 5 displays some sample images from these streams.

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. Since each of the trajectories are of different lengths (each video may contain a different number of frames) and are multidimensional, vector-based clustering such as K-means or Gaussian mixtures is difficult to apply here without some further (non-obvious) processing of the data.

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 4 displays two of the four example trajectories for each of the *Down*, *Up*, and *Right-Left* movements with the solid, dotted, and dashed lines, respectively. These lines depict the movement of the hand in the horizontal direction. The

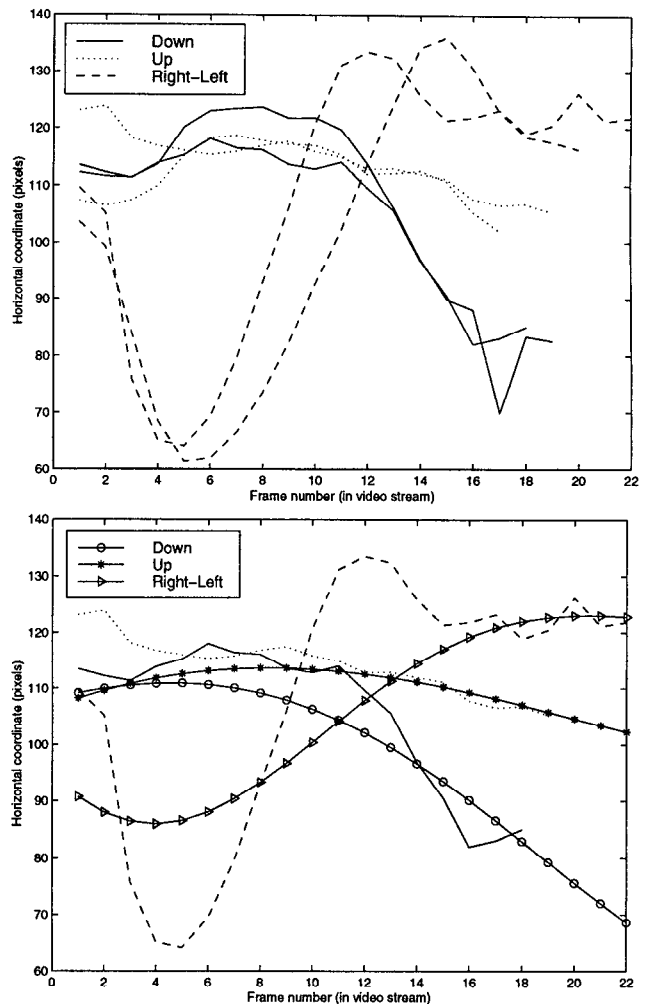


Figure 4: The top graph shows example trajectories for three movements. The bottom graph superimposes cluster lines returned from kernel regression mixtures onto some trajectories from above.

bottom graph of the figure shows the kernel regression lines returned for these three movements, superimposed on a single example trajectory (for each movement) from the above graph. Here, we see that our algorithm is able to find and *describe* the different movements in an unsupervised manner.

The picture in Figure 6 gives the resulting clustering from running linear regression mixtures on the video data. The line marked *True*, in the picture, 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,

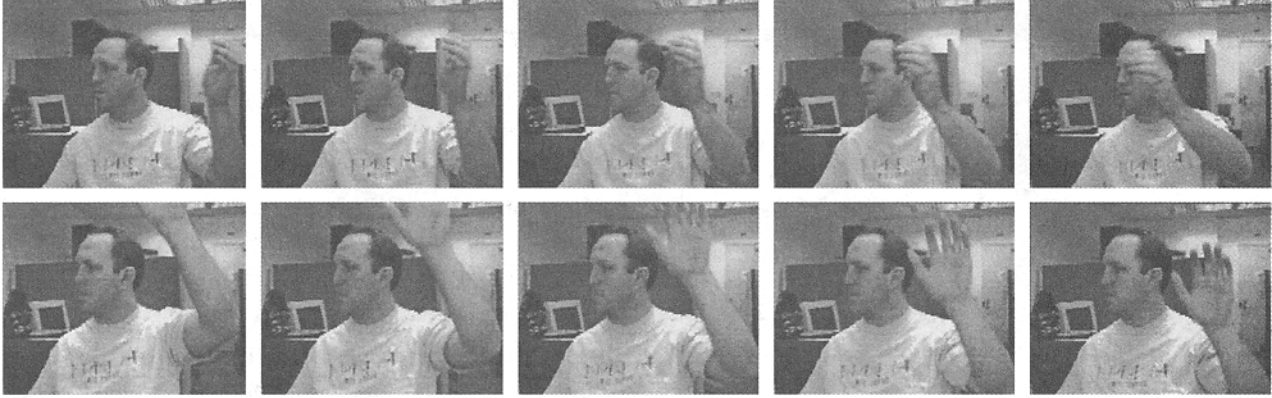


Figure 5: Sample frames from our video stream data, depicting hand movements made by an actor. The top row of images show a portion of a *left-to-right* movement, while the bottom row of images show a portion of a *down* movement.

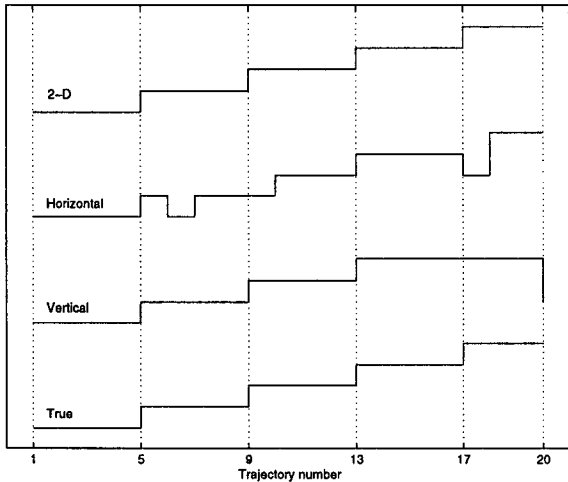


Figure 6: The graph shows clustering/classification comparisons based on regressing only on the *vertical* trajectories, the *horizontal* trajectories, or on both trajectories (*2D*). The line labeled *True* gives the *pattern* of the true clustering. Only the *2D* regression achieves the correct clustering.

the 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 graphs in Figure 7 show the horizontal regression curves returned for each of the clusters using both kernel (top) and linear (bottom) regression mixtures in 2D.

The curves describe the mean-movement behavior in the horizontal direction. Note (for example) that the *Left-Right* and *Right-Left* curves appear as inverses in the horizontal direction (as they should, since they are inverse movements in this direction), but they appear similar in the vertical direction (not shown). We can also see that the *bLeft-tRight* curve appears inverse to the *Right-Left* curve in the horizontal direction. This nicely fits with our understanding of these movements in the real world. In terms of the difference between the kernel (top) and linear (bottom) models, one can clearly see that the linear means in the bottom plot are simple summaries of the kernel means on the top.

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. 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.

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

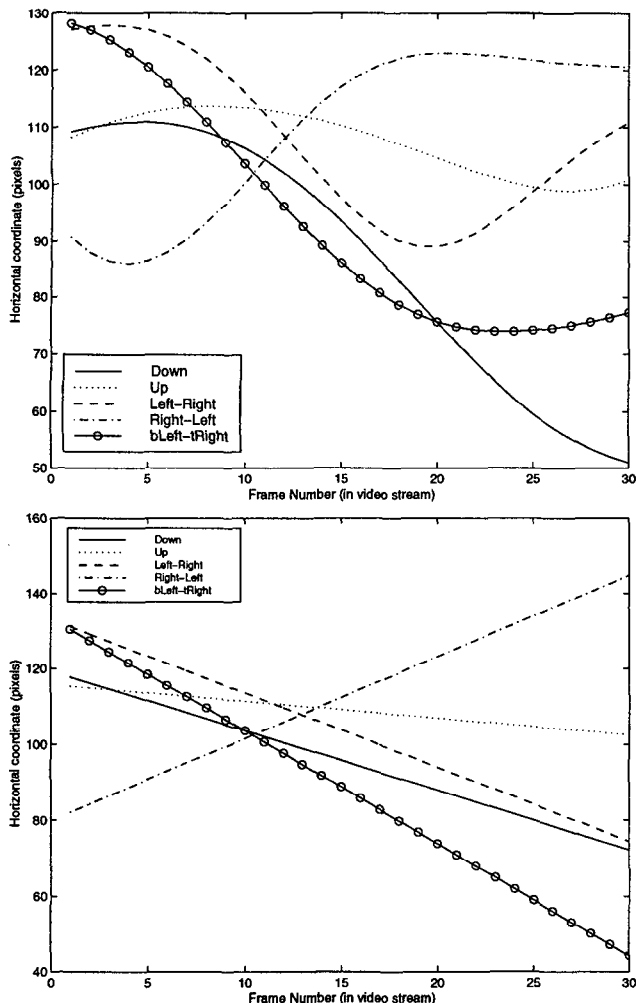


Figure 7: Estimated component model lines as returned from our EM algorithm for mixtures of kernel regression models (top) and linear regression models (below), horizontal component.

for non-parametric regression components and the technique can also be easily extended to handle multivariate trajectories. We demonstrated that the proposed approach outperforms the more traditional vector-based approaches in simulation experiments and illustrated the utility of the method on video data sequences.

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

- [1] Dempster, A.P., Laird, N.M., & Rubin, D.B. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc. B*, 39(1):1–38, 1977.
- [2] Draper, N.R., and Smith, H. *Applied Regression Analysis*. New York: John Wiley and Sons, 2nd ed., 1981.
- [3] Fan, J., and Gijbels, I. *Local Polynomial Modelling and Its Applications*. New York: Chapman & Hall, 1996.
- [4] Fraley, C., and Raftery, A.E. How many clusters? Which clustering method? Answers via Model-based cluster analysis. *Computer Journal*, 41, 578–588, 1998.
- [5] Gaffney, S., and Smyth, P. Trajectory clustering with mixtures of regression models. UCI-ICS-TR-99-15, Information and Computer Science, University of California, Irvine, 1999.
- [6] Jones, R.H. *Longitudinal Data with Serial Correlation: A State Space Approach*. New York: Chapman & Hall, 1993.
- [7] Jordan, M.I., and Jacobs, R.A. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6, 181–214, 1994.
- [8] McLachlan, G.J., and Basford, K.E. *Mixture Models: Inference and Applications to Clustering*. New York: Marcel Dekker, 1988.
- [9] McLachlan, G.J., and Krishnan, T. *The EM Algorithm and Extensions*. New York: John Wiley and Sons, 1997.
- [10] Ramsay, J.O., and Silverman, B.W. *Functional Data Analysis*. New York: Springer-Verlag, 1997.
- [11] Silverman, B.W. *Density Estimation*. New York: Chapman & Hall, 1986.
- [12] Smyth, P., Ghil, M., & Ide, K. Multiple regimes in Northern hemisphere height fields via mixture model clustering. *Journal of Atmospheric Science*, in press.
- [13] Smyth, P. Model selection for probabilistic clustering using cross-validated likelihood. ICS Technical Report 98-09 (also to appear in *Statistics and Computing*), 1998.
- [14] Stanford, D., and Raftery, A.E. Principal curve clustering with noise. Technical Report No. 317, Department of Statistics, University of Washington, February 1997.
- [15] Wand, M.P., and Jones, M.C. *Kernel Smoothing*. New York: Chapman & Hall, 1995.