Machine Learning, 47, 91–121, 2002

Bayesian Clustering by Dynamics

MARCO RAMONI marco_ramoni@harvard.edu
Children's Hospital Informatics Program, Harvard Medical School, Boston, MA 02115, USA

PAOLA SEBASTIANI sebas@math.umass.edu
Department of Mathematics and Statistics, University of Massachusetts, Amherst, MA 01003, USA

PAUL COHEN cohen@cs.umass.edu
Department of Computer Science, University of Massachusetts, Amherst, MA 01003, USA

Editor: Douglas H Fisher, Jr

Abstract. This paper introduces a Bayesian method for clustering dynamic processes. The method models dynamics as Markov chains and then applies an agglomerative clustering procedure to discover the most probable set of clusters capturing different dynamics. To increase efficiency, the method uses an entropy-based heuristic search strategy. A controlled experiment suggests that the method is very accurate when applied to artificial time series in a broad range of conditions and, when applied to clustering sensor data from mobile robots, it produces clusters that are meaningful in the domain of application.

Keywords: Bayesian learning, clustering, time series, Markov chains, heuristic search, entropy

1. Introduction

Suppose one has a set of univariate time series generated by one or more unknown processes that have characteristic dynamics. *Clustering by dynamics* is the problem of grouping time series into clusters so that the elements of each cluster have similar dynamics. For example, if a batch of time series represents sensory experiences of a mobile robot, clustering by dynamics might find clusters corresponding to abstractions of sensory inputs (Ramoni, Sebastiani, & Cohen, 2000a). If a batch contains a set of EKGs, clustering by dynamics might find clusters corresponding to the pathologies of the heart. Sound patterns can be clustered by dynamics, also, as a way to discover patterns corresponding to words in speech signals.

We regard observed data as realizations of a set of underlying stochastic processes ultimately responsible for what we observe. In our cardiac example, we expect to find a set of time series generated by healthy hearts and some sets of time series generated by different cardiac pathologies. As stochastic realizations of these processes, the observed data are burdened with variability and two EKG time series may appear to be different although they are generated by the same process. The task of a Bayesian method is to identify the most probable set of generating processes given the observed data.

This paper presents BCD: a Bayesian algorithm for clustering by dynamics. Given a batch of time series, BCD transforms each series into a Markov Chain (MC) and then clusters similar MCs to discover the most probable set of generating processes. A MCsummarizes a process dynamics by a transition probability matrix, each row in the matrix representing probabilities of transition from a state to each other state of the variable in the next time step. A transition matrix is learned for each time series in the training batch. Next, BCD groups time series generated by the same process. The task of the clustering algorithm is two-fold: to find the set of clusters that gives the best partition according to some measure, and to assign each MC to one cluster. BCD uses the posterior probability of a partition—i.e. the probability of a partition given the sample time series—as scoring metric and an entropy-based heuristic search strategy to increase search efficiency.

Bayesian clustering methods were pioneered by Cheeseman (1996) for static databases, i.e. under the assumption that the data are independent and identically distributed. More recently, Poulsen (1990), Ridgeway (1997, 1998) and Smyth (1999) extended the original method to temporal data using an approximate mixture-model approach to cluster discrete MCs within a pre-specified number of clusters. Here, we present the first exact Bayesian treatment of the task of clustering time series modelled as MCs, with no assumption on the number of clusters. A Bayesian approach is particularly well suited to clustering by dynamics because it provides a principled way to integrate prior and current evidence. Furthermore, because the posterior probability of a partition is our scoring metric, we avoid the problem of increasing the overall probability of errors that plagues classical statistical methods based on significance tests.

The remainder of this paper is organized as follows. We describe the Bayesian clustering approach in Section 2. As the search space of all possible models increases exponentially with the number of time series, Section 3 introduces a heuristic search algorithm and analyzes its computational complexity. Section 4 compares our approach to other current treatments of time series. Section 5 contains the evaluation of our method in a controlled experiment and presents an experimental comparison between BCD and an implementation of the mixture-model approach based on the EM algorithm. Section 6 describes the application of BCD to the discovery of prototype dynamics of sensory inputs in a mobile robot.

2. Clustering Markov chains

Suppose we have a batch of m time series that record the values $1, 2, \ldots, s$ of a variable X. The goal is to identify time series with similar dynamics. Consider, for example, the plot of three time series in figure 1. Each records the values of a variable with five states—labeled 1 to 5—in 50 time steps. It is not obvious that the three time series are observations of the same process. However, when we explore the underlying dynamics of the three series more closely, we find, for example, that state 2 is frequently followed by state 1, and state 3 is followed disproportionately often by state 1. We are interested in extracting these types of similarities among time series and, to do this, we model the dynamics of time series' as Markov chains (MCs). For each time series, we estimate a transition matrix from data and then we cluster transition matrices.

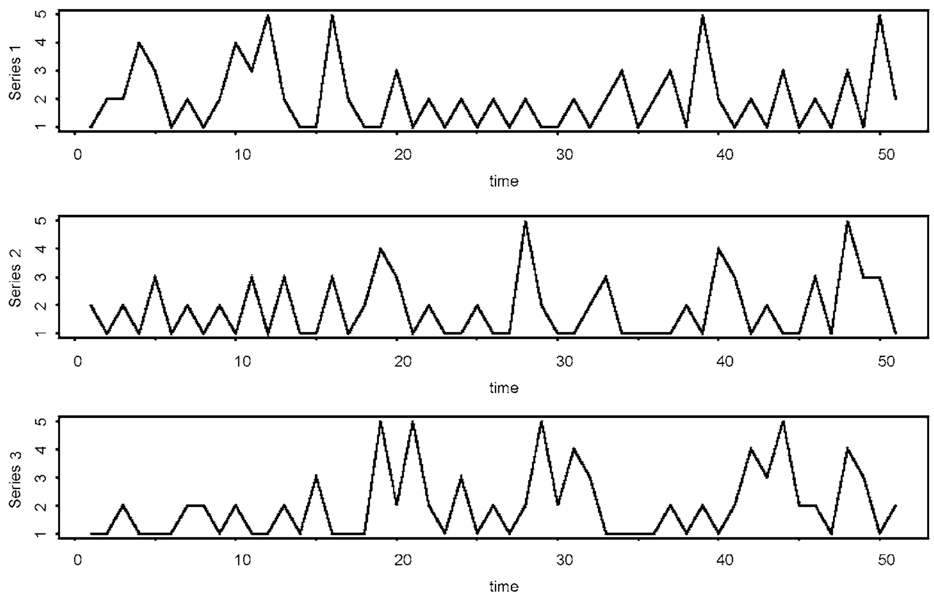

Figure 1. Plot of three time series.

2.1. *Learning Markov chains*

Suppose we observe a time series $x = (x_0, x_1, x_2, \ldots, x_{i-1}, x_i, \ldots)$, where each x_i is one of the states $1, \ldots, s$ of a variable X. The process generating the sequence x is a MC if the conditional probability that the variable visits state j at time t, given the sequence $(x_0, x_1, x_2, \ldots, x_{t-1})$, is only a function of the state visited at time $t - 1$ (Ross, 1996). Hence, we write $p(x_t = j \mid (x_0, x_1, x_2, \ldots, x_{t-1})) = p(x_t = j \mid x_{t-1})$ for any x_t in x. In other words, the probability distribution of the variable X at time t, say X_t, is *conditional independent* of the values $(x_0, x_1, x_2, \ldots, x_{t-2})$, once we know x_{t-1}. This conditional independence assumption allows us to represent a MC as a vector of probabilities $p_0 = (p_{01}, p_{02}, \ldots, p_{0s})$, denoting the distribution of X_0 (the initial state of the chain) and a matrix P of transition probabilities, where $p_{ij} = p(X_t = j \mid X_{t-1} = i)$.

$$P = (p_{ij}) = \begin{array}{c|cccc} & & & X_t & \\ X_{t-1} & 1 & 2 & \cdots & s \\ \hline 1 & p_{11} & p_{12} & \cdots & p_{1s} \\ 2 & p_{21} & p_{22} & \cdots & p_{2s} \\ \vdots & & & \cdots & \\ s & p_{s1} & p_{s2} & \cdots & p_{ss} \end{array}$$

Given a time series generated by a MC, we can estimate the probabilities p_{ij} from the data and store them in a matrix $\hat{P}$. The assumption that the generating process is a MC implies that only pairs of transitions $X_{t-1} = i \rightarrow X_t = j$ are sufficient, where a transition $X_{t-1} = i \rightarrow X_t = j$

occurs when we observe the pair $X_{t-1} = i$, $X_t = j$ in the time series. Hence, the time series can be summarized into an $s \times s$ contingency table containing the transition frequencies $n_{ij} = n(i \to j)$ where, for simplicity, we denote the transition $X_{t-1} = i \to X_t = j$ by $i \to j$. The frequencies n_{ij} are used to estimate the transition probabilities p_{ij} characterizing the dynamics of the process that generated the data.

However, the observed transition frequencies n_{ij} may not be the only source of information about the process dynamics. We may also have some background knowledge that can be represented as a hypothetical time series of length $\alpha + 1$ in which the α transitions are divided into α_{ij} transitions of type $i \to j$. This background knowledge gives rise to a $s \times s$ contingency table, homologous to the frequency table, containing these hypothetical transitions α_{ij} that we call *hyper-parameters*. A Bayesian estimation of the probabilities p_{ij} takes into account this prior information by augmenting the observed frequencies n_{ij} by the hyper-parameters α_{ij} so that the *Bayesian estimate* of p_{ij} is

$$\hat{p}_{ij} = \frac{\alpha_{ij} + n_{ij}}{\alpha_i + n_i} \tag{1}$$

where $\alpha_i = \sum_j \alpha_{ij}$ and $n_i = \sum_j n_{ij}$. Thus, α_i and n_i are the numbers of times the variable X visits state i in a process consisting of α and n transitions, respectively. By writing Eq. (1) as

$$\hat{p}_{ij} = \frac{\alpha_{ij}}{\alpha_i}\frac{\alpha_i}{\alpha_i + n_i} + \frac{n_{ij}}{n_i}\frac{n_i}{\alpha_i + n_i} \tag{2}$$

we see that $\hat{p}_{ij}$ is an average of the classical estimate n_{ij}/n_i and of the quantity α_{ij}/α_i, with weights depending on α_i and n_i. Rewriting of Eq. (1) as (2) shows that α_{ij}/α_i is the estimate of p_{ij} when the data set does not contain transitions from the state i—and hence $n_{ij} = 0$ for all j—and it is therefore called the *prior* estimate of p_{ij}, while $\hat{p}_{ij}$ is called the *posterior estimate*. The variance of the prior estimate α_{ij}/α_i is given by $(\alpha_{ij}/\alpha_i)(1 - \alpha_{ij}/\alpha_i)/(\alpha_i + 1)$ and, for fixed α_{ij}/α_i, the variance is a decreasing function of α_i. Since small variance implies a large precision about the estimate, α_i is called the *local precision* about the conditional distribution $X_t \mid X_{t-1} = i$ and it indicates the level of confidence about the prior specification. The quantity $\alpha = \sum_i \alpha_i$ is the *global* precision, as it accounts for the level of precision of all the s conditional distributions. Further details may be found in Ramoni and Sebastiani (1999).

When n_i is large relative to α_i, so that the ratio $n_i/(\alpha_i + n_i)$ is approximately 1, the Bayesian estimate reduces to the classical estimate n_{ij}/n_i. In this way, the Bayesian estimate of the transition probability p_{ij} is approximately 0 when $n_{ij} = 0$ and n_i is large. The variance of the posterior estimate p_{ij} is $\hat{p}_{ij}(1 - \hat{p}_{ij})/(\alpha_i + n_i + 1)$ and, for fixed $\hat{p}_{ij}$, it is a decreasing function of $\alpha_i + n_i$, the local precision augmented by the sample size n_i. Hence, the quantity $\alpha_i + n_i$ can be regarded as a measure of the *confidence* in the estimates: the larger the sample size, the stronger the confidence in the estimate.

Example. Table 1 reports the frequencies of transition n_{ij} $i, j = 1, \ldots, 5$ observed in the first time series in figure 1 and the learned transition matrix when the prior global precision is $\alpha = 5$ and $\alpha_{ij} = 1/5$. The matrix $\hat{P}$ describes a dynamic process characterized by transitions among states 1, 2 and 3 while states 4 and 5 are visited rarely. Note that although the observed frequency table is sparse, as 14 transitions are never observed, null frequencies of some

Table 1. Observed and learned transition matrices for the first time series in figure 1.

		1	2	3	4	5			1	2	3	4	5
	1	3	12	3	0	3		1	0.15	0.55	0.15	0.01	0.15
$N =$	2	11	1	2	2	0	$\Rightarrow \hat{P} =$	2	0.66	0.07	0.13	0.13	0.01
	3	6	0	0	0	1		3	0.78	0.03	0.03	0.03	0.15
	4	0	0	2	0	0		4	0.07	0.07	0.73	0.07	0.07
	5	0	4	0	0	0		5	0.04	0.84	0.04	0.04	0.04

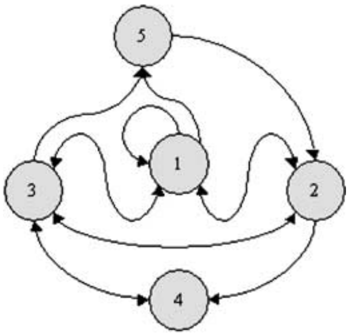

Figure 2. Markov Chain induced from data.

transitions do not induce null probabilities. The small number of transitions observed from state 3 ($n_3 = 7$), state 4 ($n_4 = 2$) and state 5 ($n_5 = 4$) do not rule out, for instance, the possibility of transitions from 3 to either 2, 3 or 4. A summary of the essential dynamics is in figure 2 in which double headed paths represent mutual transitions. Transitions with probability smaller than 0.05 are not represented.

2.2. Clustering

The second step of BCD is an unsupervised agglomerative clustering of the set of MCs, encoding the set $S = \{S_i\}$ of m time series, on the basis of their dynamics. The task of the clustering algorithm is two-fold: find the set of clusters that gives the best partition according to some measure, and assign each time series to one cluster. A partition is an assignment of MCs to clusters such that each time series belongs to exactly one cluster.

We regard the task of clustering MCs as a Bayesian model selection problem. In this framework, the model we seek is the most probable way of partitioning MCs according to their similarity, *given* the data. We use the probability of a partition given the data, i.e. the *posterior probability* of the partition, as a scoring metric and we select the model with maximum posterior probability. Formally, this is done by regarding a partition as a hidden discrete variable C. Each state C_k of C represents a cluster of time series, and hence determines a transition matrix. Graphically, the partition of the transition matrices, that we

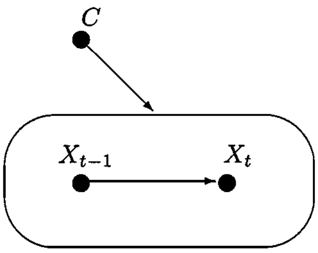

Figure 3. Graphical representation of a clustering model M_c.

can learn from S, can be represented as in figure 3. The directed link from the node C and the node containing the MC represents the dependence of the transition matrix $X_t \mid X_{t-1}$ on C. The number c of states of C is unknown, but the number m of available MCs imposes an upper bound, as $c \leq m$. Each partition identifies a model M_c, and we denote by $p(M_c)$ its prior probability. By Bayes' Theorem, the posterior probability of M_c, given the sample S, is

$$p(M_c \mid S) = \frac{p(M_c)p(S \mid M_c)}{p(S)}.$$

The quantity $p(S)$ is the marginal probability of the data. Since we are comparing all the models over the same data, $p(S)$ is constant and, for the purpose of maximizing $p(M_c \mid S)$, it is sufficient to consider $p(M_c)p(S \mid M_c)$. Furthermore, if all models are *a priori* equally likely, the comparison can be based on the *marginal likelihood* $p(S \mid M_c)$, which is a measure of how likely the data are if the model M_c is true.

The quantity $p(S \mid M_c)$ can be computed from the marginal distribution (p_k) of C and the conditional distribution (p_{kij}) of $X_t \mid X_{t-1} = i, C_k$, where C_k is the cluster membership, using a well-known Bayesian method (Cooper & Herskovitz, 1992). Let n_{kij} be the observed frequencies of transitions $i \rightarrow j$ in cluster C_k, and let $n_{ki} = \sum_j n_{kij}$ be the number of transitions observed from state i in cluster C_k. We define m_k to be the number of time series assigned to cluster C_k. The observed frequencies (n_{kij}) and (m_k) are the data required to learn the probabilities (p_{kij}) and (p_k) respectively and, together with the prior hyper-parameters α_{kij}, they are all that is needed to compute the probability $p(S \mid M_c)$, as

$$p(S \mid M_c) = f(S, C) f(S, X_{t-1}, X_t, C). \tag{3}$$

Intuitively, the first quantity is the likelihood of the data, if we assume that we can partition the m MCs into c clusters, and it is computed as

$$f(S, C) = \frac{\Gamma(\alpha)}{\Gamma(\alpha + m)} \prod_{k=1}^{c} \frac{\Gamma(\alpha_k + m_k)}{\Gamma(\alpha_k)}.$$

The second quantity measures the likelihood of the data when, conditional on having c clusters, we assign each time series to one and only one cluster. This quantity is given by

$$f(S, X_{t-1}, X_t, C) = \prod_{k=1}^{c} \prod_{i=1}^{s} \frac{\Gamma(\alpha_{ki})}{\Gamma(\alpha_{ki} + n_{ki})} \prod_{j=1}^{s} \frac{\Gamma(\alpha_{kij} + n_{kij})}{\Gamma(\alpha_{kij})}$$

where $\Gamma(\cdot)$ denotes the Gamma function. Equation (3) is derived directly from the results in (Cooper & Herskovitz, 1992) by treating a MC as *child* variable of C, using the popular terminology of Bayesian Networks. Once created, the transition probability matrix of a cluster C_k can be estimated as $\hat{p}_{kij} = (\alpha_{kij} + n_{kij})/(\alpha_{ki} + n_{ki})$.

We conclude this section by suggesting a choice of the hyper-parameters α_{kij}. We use uniform prior distributions for all the transition probability matrices considered at the beginning of the search process. The initial $m \times s \times s$ hyper-parameters α_{kij} are set equal to $\alpha/(ms^2)$ and, when two MCs are similar and the corresponding observed frequencies of transitions are merged, their hyper-parameters are summed up. Thus, the hyper-parameters of a cluster corresponding to the merging of m_k initial MCs will be $m_k\alpha/(ms^2)$. In this way, the specification of the prior hyper-parameters requires only the prior global precision α, which measures the confidence in the prior model. An analogous procedure can be applied to choose the hyper-parameters α_k associated with the prior estimates of p_k. Since $\Gamma(x)$ is defined only for values greater than zero, the hyper-parameters α_{kij} must be non-negative. Note further that a choice of uniform hyper-parameters biases the algorithm toward the hypothesis that the initial m time series are generated by the same MC.

3. A heuristic search method

To implement the clustering method described in the previous section, we might evaluate all possible partitions and return the one with the highest posterior probability. The number of possible partitions growing exponentially with the number of MCs, a heuristic method is required to make the search feasible. We use a measure of similarity between estimated transition probability matrices to guide the search process. The resulting algorithm is called *Bayesian Clustering by Dynamics* (BCD).

3.1. The algorithm

The algorithm performs a bottom-up search by recursively merging the closest MCs (representing either a cluster or a single time series) and evaluating whether the resulting model is more probable than the model where these MCs are kept distinct. The similarity measure that guides the process can be any distance between probability distributions. Let P_1 and P_2 be matrices of transition probabilities of two MCs. Because each is a collection of s probability distributions, and rows with the same index are probability distributions conditional on the same event, the measure of similarity that BCD uses is an average of the symmetrized Kullback-Liebler distance between corresponding rows. Let p_{1ij} and p_{2ij} be the probabilities of the transition $i \rightarrow j$ in P_1 and P_2. The Kullback-Liebler distance of these two probability distributions is

$$d(p_{1i}, p_{2i}) = \sum_{j=1}^{s} p_{1ij} \log \frac{p_{1ij}}{p_{2ij}} \tag{4}$$

The distance in Eq. (4) is not symmetric because $d(p_{1i}, p_{2i}) \neq d(p_{2i}, p_{1i})$. The symmetric version of it was introduced by Jeffreys (1946) and is defined as $D(p_{1i}, p_{2i}) =$

$[d(p_{1i}, p_{2i}) + d(p_{2i}, p_{1i})]/2$. The average distance between P_1 and P_2 is then $D(P_1, P_2) = \sum_i D(p_{1i}, p_{2i})/s$. Note that this distance becomes 0 when $P_1 = P_2$ and it is otherwise greater than zero. The current implementation of BCD is based on the symmetric Kullabck-Liebler distance but other measures of distance, such as the *mean square error*, could be used instead. The rationale behind the distance measure is that merging more similar MCs earlier should find sooner more probable models and increase the marginal likelihood in 3 used as a scoring metric by the algorithm.

1. The algorithm takes as input a set S of m time series and returns a partition B of times series S.

```
procedure BCD(S)
  $MCs \leftarrow \emptyset$; $Distances \leftarrow \emptyset$;
  while $i \leq |S|$ do $MCs \leftarrow MCs \bigcup \{\text{MC}(S_i)\}$;
  while $i \leq |MCs|$ do
    while $j \leq |MCs|$ do
      if $i \neq j$ then $Distances \leftarrow Distances \bigcup \{\text{DISTANCE}(MCs_i, MCs_j)\}$;
    end
  end
  $Distances \leftarrow \text{SORT}(Distances)$;
  $P_{new} \leftarrow \text{ML}(MCs)$; $Best \leftarrow MCs$;
  while $P_{new} > P_{old}$ do
    $P_{old} \leftarrow P_{new}$;
    while $i \leq |Distances|$ do
      $MC \leftarrow \text{MERGE}(Distances_i)$;
      $Current \leftarrow \{MCs \backslash \{Distances_i\}\} \cup \{MC\}$;
      when $P_{new} > \text{ML}(Current)$ do
        $Best \leftarrow Current$; $P_{new} \leftarrow \text{ML}(Current)$;
        while $j \leq |Distances|$ do
          if $Distances_i \bigcap Distances_j \neq \emptyset$ then $Distances \leftarrow Distances \backslash \{Distances_j\}$;

        end
        while $j \leq |Best|$ do
          $Distances \leftarrow Distances \bigcup \text{DISTANCE}(MC, Best_j)$;
        end
        $Distances \leftarrow \text{SORT}(Distances)$;
        return
      end
    end
  end
  return $Best$
end
```

Initially, the algorithm transforms each of the m time series in a set S into a MC. This is accomplished by the function MC($\cdot$), which follows the procedure described in Section 2.

Then, the algorithm sorts the set *Distances* of pairwise distances between the m generated MCs into an ascending order with the function DISTANCE(·,·). This function returns a pair of MCs indexed by a measure of their mutual distance as described above. The last step of the initialization phase is the computation of the marginal likelihood $p(S \mid M_s)$, where M_s represents the model in which each time series is generated by a different MC is taken as the current best model. The marginal likelihood (3) is computed by the function ML(·), taking as its argument a set of MCs.

The iterative core of the algorithm loops on the ordered set of MC pairs and tries to merge the two closest MCs into a single MC. The function MERGE(·) performs this merging as described in Section 2. This function also estimates the marginal likelihood $p(S \mid M_c)$, where M_c is the model in which the two merged MCs are replaced by the MC resulting from their merging. If the marginal likelihood of this model is higher than the marginal likelihood of the current *Best* model, the model M_c is taken as current *Best*, all the ordered pairs involving one of the merged MCs are removed from the set *Distances*, the distances between the new MC and the other MCs in the *Best* model are sorted into the set *Distances*, and the procedure is iterated on the new *Best* model. Otherwise, the procedure tries to merge the remaining pairs of MCs in the ordered set *Distances*. If no merging will result in a model with higher marginal likelihood than the current *Best* model, the procedure stops and returns the *Best* model found so far.

3.2. Computational complexity

We shall assume that the quantities required to compute Eq. (3) in the function ML(·) and Eq. (4) in the function DISTANCE(·,·) have been pre-computed and stored in some array. We do not consider the time required by the function SORT(·).

Recall that s is the number of states of the variable generating the m time series in the set S. We assume that the longest time series contains n data points. The execution of the function MC(·) requires at most ns^2 applications of Eq. (1). Therefore, the first *while* statement requires $O(mns^2)$ time. The following nested *while* statements call the DISTANCE(·,f)unction $(m\frac{m-1}{2})$ times, in case of symmetric distances, and $(m(m-1))$ times otherwise. As the function DISTANCE(·,·) applies Eq. (4) s^2 times, the cost of this second *while* statement is bounded by $O(s^2m^2)$. The third *while* statement is called, in the worse case, m times on the $(m(m-1))$ elements of *Distances*. As the function ML(·) sums over a table of ms^2 elements, the cost of the third *while* statement is bounded by $O(m^4s^2)$. When a new model is successfully created, we must also update the current set of *Distances* in order to sort the new generated MC into the other elements of the sorted list $Best$. As the number of new models is bounded by $(m-1)$, this updating requires, in the worse case, an additional $m^2 + s^2m^2(m-1)$ steps. Putting these results together, we obtain $O(nms^2) + O(m^2s^2) + O(m^4s^2) + O(m^2) + O(m^3s^2)$ and we can bound the cost of the algorithm by $O(m^4s^2)$.

4. Related works

BCD models time series as first order MCs. More complex models involve the use of k-order Markov chains (Saul & Jordan, 1999), in which the memory of the time series is extended to a window of k time steps, or Hidden Markov Models (MacDonald & Zucchini, 1997), in which hidden variables H are introduced to decompose the complex auto-regressive structure of the time series into smaller pieces. Hidden Markov Model were originally introduced in speech recognition (Rabiner, 1989) and are nowadays applied in many fields from DNA and protein sequencing (Lio & Goldman, 1998; Jaakkola, Diekhans, & Haussler, 2000) to robotics (Firoiu & Cohen, 1999) and language learning (Oates, 1999).

There are significant differences between BCD and Hidden Markov Models. BCD represents the process generating time series with MCs and clusters similar MCs by creating a variable C to encode cluster membership. Conditional on each state of the variable C, the model for the time series is a MC with transition probability $p(X_t = j \mid X_{t-1} = i, C_k)$. Graphically, this assumption is represented by the model in figure 3. The oval represents one MC and C separates different ovals, so that, conditional on $= C_k$, the transition probability from state i to state j is $p(X_t = j \mid X_{t-1} = i, C_k)$ and this is independent of other MCs. In a Hidden Markov Model, the hidden variable H (or variables) allows one to compute the transition probabilities among states as $p(X_t|X_{t-1}, H_k) = p(x_t|H_k)$, so that conditioning on the state of the hidden variable makes the dependence of X_t on X_{t-1} vanish. Figure 4 represents graphically this assumption. A detailed explanation of the difference between a Hidden Markov Model and the model used by BCD is in Smyth, Heckerman, and Jordan (1997).

The approach used in BCD is similar to the mixture-model approach proposed by Poulsen (1990) and further developed by Ridgeway (1997), Ridgeway and Altschuler (1998) and Smyth (1999). Ridgeway describes a probabilistic method for clustering discrete MCs with a pre-specified number of clusters. The algorithm models the partition of time series into a pre-specified number of clusters of MCs as a mixture model. The EM-algorithm is used to compute the mixture weights and the transition probabilities maximizing a scoring function, and then each time series is assigned to the cluster from which it is generated with maximum posterior probability. This algorithm, that we will refer to as EMC (EM-clustering), assumes a known number of clusters and, conditional on this number, finds the partition of time series into clusters. The scoring function is the likelihood function, that is, the probability of the data given some parameters representing the mixture weights and the transition probabilities of each cluster of MC. Compared to EMC, BCD finds both the best number of clusters and the optimal assignment of time series to clusters by searching for the partition that maximizes the marginal likelihood: the expected likelihood function in which the

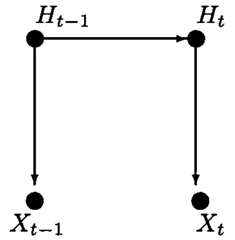

Figure 4. Graphical representation of a Hidden Markov Model.

expectation is taken with respect to the prior distribution of the parameters. In this way, rather than using maximum likelihood estimates of the parameters to compute a maximized score, BCD blends data and prior information to compute an average score. Intuitively, this approach should gain robustness and, indeed, in the next section we will show an empirical evaluation on simulated data which supports this intuition.

5. Experimental evaluations

This section includes two controlled experimental evaluations of BCD. The first experiment aims at assessing the accuracy of BCD. The second experiment compares BCD and an EM-based implementation of the mixture-model clustering approach.

5.1. Experimental evaluation of BCD

Here we describe the results of a controlled experiment to evaluate the accuracy of the algorithm when it is applied to a batch of time series generated from different MCs. The results show an overall accuracy that appears to suffer only when the time series in the batch are very short and have very similar dynamics.

5.1.1. Procedure. To assess the accuracy of the algorithm, we varied four factors:

Factor 1 The length of each time series: 25, 50, 125 and 250 time steps.
Factor 2 The number of different MCs generating the batch of data: either 4 or 8.
Factor 3 The number of time series generated from each MC: either equal or different.
Factor 4 Global prior precision: $\alpha = 4, 8, 80$.

The first three factors yield sixteen experimental conditions. For each condition, we generated eighty time series. Each time series included at most five distinct values, i.e. it was generated by a MC with five states. We then tested the algorithm with these time series for three values of the global prior precision α.

The choice of values of Factor 1 follows from the fact that our time series are generated by MCs with five states. For a given number of states, we require a minimum sample size to ensure that states are visited often enough to yield good estimates of transition probabilities. A time series of $n + 1$ steps generated from a MC on s states with uniform probabilities yields n/s^2 expected transitions $i \rightarrow j$, for any i and j. When the time series is not generated from a MC with uniform transition probabilities, some observed transition frequencies will be smaller than n/s^2, others will be larger. Hence, if n is not sufficiently larger than s, we expect to have several null transition frequencies. We decided to fix the number of states of each MC to five and to generate, from each MC, time series of length 25 = "very short", 50 = "short", 125 = "medium" and 250 = "long". Very short and short time series give an expected number of one and two transitions in the uniform case, so that we expect to have several zeros when the generating MCs are not uniform. The expected number of transitions are five and ten in the other two cases.

The time series in each experimental batch are generated by either four or eight MCs. We constructed the MCs as follows: Each of five rows of a transition matrix is a probability distribution with masses on five states. When we think of probability distributions, we imagine symmetric, uniform or skew-symmetric distributions. Uniform distributions aside, the other patterns are characterized by changing the concentration of probability masses—without changing the shape of the distributions—to have different degrees of variability. We might construct an infinite number of transition matrices whose rows mix these different distributions. To avoid selection bias, we generated a set of sixteen probability distributions given by a uniform distribution and symmetric and skew-symmetric distributions. We then generated eight transition probability matrices $P_1, \ldots, P_8$ by sampling, in each case, five distributions from this set. For the cases in which we generated the time series from four MCs, we sampled the four matrices P_1, P_4, P_5 and P_7.

The symmetric Kullback-Liebler distance between pairs of transition probability matrices so generated is displayed is Table 2 and ranges between 0.75, in which the two transition probability matrices have three identical rows, and 1.86. In the latter case, the two transition probability matrices represent very different dynamics, as in one case the chain is expected to visit all states 1–5 while, in the other case, transitions are mainly limited to states 1 and 4.

We suspected that having generating processes equally represented in the batch of time series can help the algorithm to cluster MCs correctly. Hence, for each of the eight combinations of length and number of generating MCs, we created two batches of time series in which the number of time series generated from each MC is either constant or variable. In the constant conditions, each of four (or eight) MCs generated 20 (or 10) time series. In the unequal conditions, each of four MCs generates 18, 10, 33 and 19 time series, respectively; and each of eight MCs generated 12, 23, 2, 4, 5, 12, 16, 6, series. These unequal distributions were generated randomly. We also decided to start each simulated time series from the same initial point. However, since some initial point can be more advantageous for some time series than others, we choose the common initial point at random. Thus we generated eighty time series for each of the sixteen combinations of levels of the factors. We first generated four sets of eighty time series of length 250 time steps, and then we extracted four sets given by the first 25 steps, four sets given by the first 50 time steps, and four sets given by the first 125 time steps. In this way, the comparisons of the results for time series of varying lengths

Table 2. Kullback Liebler distance between pairs of transition probability matrices used to generate the test sequences.

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
P_1	0.00	1.85	1.44	1.57	1.16	1.79	1.56	1.62
P_2	–	0.00	1.35	1.61	1.54	1.79	1.52	1.27
P_3	–	–	0.00	1.81	1.19	1.54	1.21	0.75
P_4	–	–	–	0.00	1.86	1.78	1.40	1.85
P_5	–	–	–	–	0.00	1.86	1.68	0.90
P_6	–	–	–	–	–	0.00	1.50	1.60
P_7	–	–	–	–	–	–	0.00	0.83

are not confounded with the fact that we observe different processes, although generated by the same MC.

We then applied the BCD algorithm to the sixteen data sets for three different values of the global prior precision. Values of $\alpha = 4$ or 8 give a small adjustment to the observed frequency, as the initial values of the prior hyper-parameters are $\alpha_{kij} = 0.002$ and $\alpha_{kij} = 0.004$ and they become at most $\alpha_{kij} = 0.16$ in the worst case in which the algorithm merges the eighty time series into one cluster. We note that, since several transition frequencies are zeros, small values of α_{kij} are supposed to affect the precision of the posterior probability. A choice of $\alpha = 80$ corresponds to setting $\alpha_{kij} = 0.20$ initially and, when the algorithm begins to assign time series to clusters, these values increase.

5.1.2. Evaluation measures. Recall that the task of the clustering algorithm is two-fold: Find the set of clusters that gives the best partition of time series, and assign the time series to clusters. Therefore, we evaluate the algorithm using three performance measures: the number of clusters found in each data set; the number of time series correctly assigned to clusters; and a measure of the loss of data information induced by clustering. For each cluster C_j found by the algorithm, we find the generating MC with transition matrix P_i that has minimum Kullback-Liebler distance from the transition probability matrix of the cluster. We then compute the number of time series generated by the MC with transition matrix P_i that are assigned to the cluster C_j. The cumulative sum of time series assigned, correctly, to all clusters is the value of the second performance measure. The first performance measure ranges between 1, when the algorithm creates only one cluster, and 80, if no time series are clustered. The correct value is 4 when the time series are generated by 4 MCs and 8 in the other case. The second performance measure ranges between 0—when the algorithm does not assign correctly any time series—and 80—when all time series are assigned correctly to their generating MCs.

The rationale behind the third performance measure is that clustering by dynamics involves two levels of abstraction. First, the transitions in a time series are summarized in a transition matrix, or MC; second, time series are grouped into clusters, and the cluster MCs are averages of the constituent MCs. Both operations lose information. The *log-score* of a transition helps us evaluate these losses (Hand, 1997). For each time series S_k, we estimate the transition probability matrix of the MC generating the series S_k by using the estimation method described in Section 2. Let $(\hat{p}_{o,kij})$ be the transition probability matrix estimated from series S_k. We use the estimated transition probabilities to compute the score $s_{o,kij} = -\log \hat{p}_{o,kij}$ of the transition $i \to j$, and this is done for all transitions *observed* in the kth time series. This score penalizes a MC by assigning large values to observed transitions when the MC assigns them small probabilities. We sum this score over all transitions in the time series to get a cumulative score $s_{o,k}$ for the loss incurred by summarizing the observed time series S_k into a MC. Now, instead of computing losses for the time series S_k based on the MC with transition probabilities $\hat{p}_{o,kij}$, we compute them based on the estimated transition probability matrix of MC for the *cluster* to which the time series belongs. We let $(\hat{p}_{c,kij})$ be the transition probability matrix of the cluster C to which S_k belongs, and we compute the score $s_{c,kij} = -\log \hat{p}_{c,kij}$ assigned to the transition $i \to j$ observed in the time series. As before, we sum $s_{c,kij}$ over the transitions within the time series S_k to get a cumulative score

$s_{c,k}$ for the loss incurred by summarizing the observed time series into a cluster of MCs. With a batch of m time series, this procedure determines m pairs of cumulative scores $(s_{o,k}, s_{c,k})$. For each time series, the data-score $s_{o,k}$ will be inferior to the cluster-score $s_{c,k}$, since the latter is computed after two summaries of the data, while the former is computed after just one summary of the data. A scatter plot of the scores $s_{c,k}$ versus $s_{o,k}$ provides a simple description of the loss of information due to clustering time series. A numerical summary of this loss is then computed by fitting a regression line $s_c = a + bs_o$. The magnitude of the intercept term a is the average loss of information due to summarizing the data into clusters of MCs, compared to summarizing the data into MCs. The slope b is the rate at which information is lost, consequent to clustering. The coefficient of determination R^2, which measures the ratio between the variability explained by the regression line and the overall variability of the scores, is used to decide if the loss of information due to clustering can be sufficiently described by a linear function. The ratio $(s_c - s_o)/s_o$ measures the loss of information due to clustering, relative to the data-score, and its average $(a + (b - 1)s_o)/s_o$ can be used to quantify the relative loss of data information incurred by clustering.

5.1.3. Results and discussion. The number of clusters created by the BCD algorithm in each of the sixteen experimental conditions is shown in Table 3 (global prior precision is $\alpha = 8$. The results were not sensitive to changes of the global prior precision). Table 4 represents the accuracy with which the algorithm assigns time series to clusters. For conditions in which the batch of 80 time series was generated by four MCs, exactly four clusters were always found, and in all but one condition, all time series were correctly assigned to their generating MCs. In one condition, when the time series are only 25 steps long, the algorithm wrongly assigns a single time series generated from matrix P_7 to a cluster of series generated from matrix P_5. Thus, in the conditions in which we used four generating MCs neither the lengths of the time series nor the fact that the generating MCs are represented unevenly in the data set has any effect. Figure 5 displays scatter plots of the data and cluster-scores computed in the 4 experimental conditions in which equal numbers of time series were generated by four different MCs. The plots show a large correlation between the data and cluster-scores. The correlation becomes stronger as the length of the originated time series increases. When the time series are 25 time steps long, the fitted regression line explains 67% of the overall variability of the cumulative scores, and the coefficient of determination R^2 increases to 94%, when the time series are 250 time steps long. The slopes of the four regression lines are 1.23, 1.29, 1.07, and 1.01, thus showing that the loss of information decreases with the

Table 3. Number of clusters created in the sixteen data sets.

	Number of generated time series per MC							
	Equal proportions				Different proportions			
Length	25	50	125	250	25	50	125	250
4 MCs	4	4	4	4	4	4	4	4
8 MCs	8	8	8	8	4	7	7	8

Table 4. Number of time series correctly assigned to clusters in the sixteen data sets.

	Number of generated time series per MC							
	Equal proportions				Different proportions			
Length	25	50	125	250	25	50	125	250
4 MCs	79	80	80	80	80	80	80	80
8 MCs	77	80	80	80	63	76	77	80

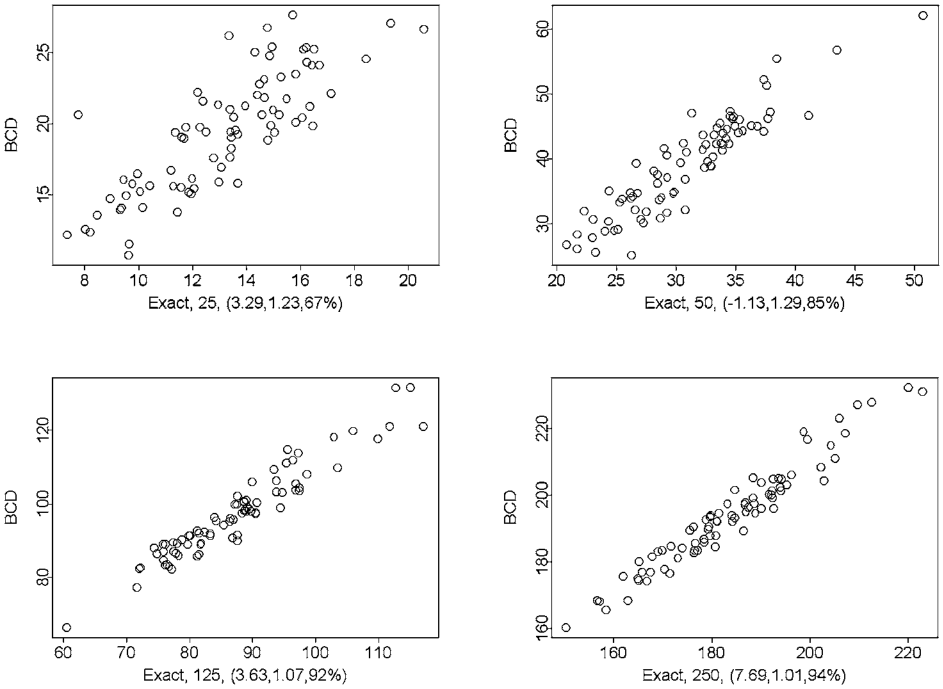

Figure 5. Scatter plot of the data (x-axis) and cluster (y-axis) scores in the group of experiments, in which the time series were generated by four different MCs, and each MC is equally represented. Values within brackets are respectively the regression line fitted to each group of scores.

length of the time series. The intercept terms are all small, ranging from 3.29 to 7.69, so that the overall loss of information is never large. The average losses of data information due to clustering, relative to the data-scores, are $(3.29 + .23s_o)/s_o$; $(-1.13 + .29s_o)/s_o$; $(3.63 + .07s_o)/s_o$; $(7.69 + 0.01s_o)/s_o$. The ratios $(3.29 + .23s_o)/s_o$, $(3.63 + .07s_o)/s_o$, and $(7.69 + 0.01s_o)/s_o$ are decreasing functions of s_o, and are bounded below by 0.23, 0.07 and 0.01, so that, for example, when s_o is large and the time series are long 250 steps, clustering determines a loss of data information of 1%. The ratio $(-1.13 + .29s_o)/s_o$ is increasing and bounded above by .29. The plots in figure 6 show similar results for the data

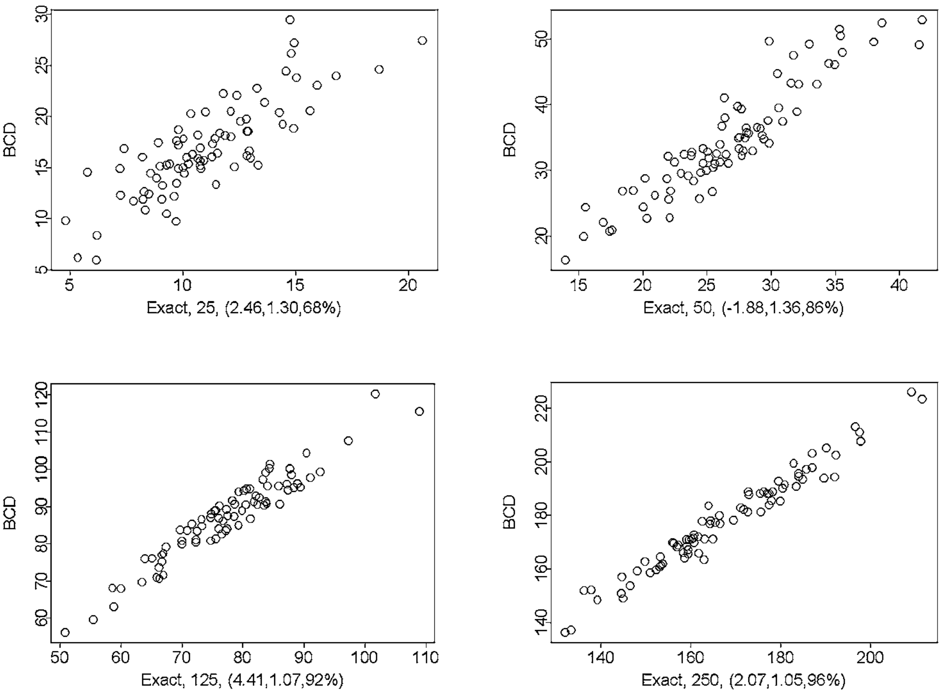

Figure 6. Scatter plot of the data (x-axis) and cluster (y-axis) scores in the group of experiments, in which the time series were generated by four different MCs, and the MCs are unevenly represented. Values within brackets are respectively the intercept term, the slope and the coefficient of determination of the regression line fitted to each group of scores.

and cluster cumulative scores in the 4 experimental conditions in which different numbers of time series were generated by four different MCs. When eight MCs are used to generate the 80 time series, we observe, as expected, a slight decrease in accuracy, because fewer time series are available to represent each MC. Consider first the four conditions in which each MC generates exactly ten time series. Table 3 shows that, in these cases, the algorithm generated the correct number of clusters, one for each generating MC. Table 4 shows that in three of these four conditions, the assignment of time series to clusters was perfect. Only when the time series were very short problems arise: One cluster in this condition contains just eight time series, albeit generated by the same MC; two clusters contain ten time series generated by the same MC and one series wrongly assigned. One cluster contains nine time series generated from the same MC and one wrongly assigned.

Globally, three time series are mis-assigned so that the second performance measure is 77. The scatter plots of the cumulative cluster-scores against the cumulative data-scores in figure 7 show again a large linear association, which becomes stronger with the increasing length of the time series. The average losses of data information due to clustering, relative to the data-scores, are $(2.27 + .26s_o)/s_o$; $(4.08 + .12s_o)/s_o$; $(3.77 + .07s_o)/s_o$; $(-1.13 + 0.06s_o)/s_o$. The ratios $(2.27 + .26s_o)/s_o$; $(4.08 + .12s_o)/s_o$; $(3.77 + .07s_o)/s_o$ are

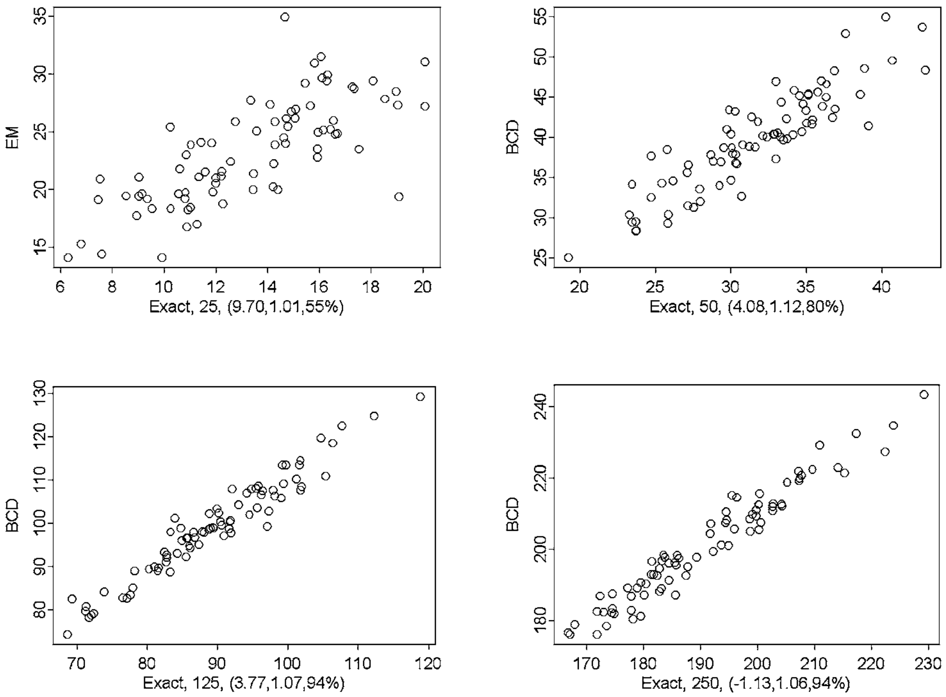

Figure 7. Scatter plot of the data (x-axis) and cluster (y-axis) scores in the group of experiments, in which the time series were generated by eight different MCs, and each MC is equally represented. Values within brackets are respectively the intercept term, the slope and the coefficient of determination of the regression line fitted to each group of scores.

increasing functions of s_o and, therefore, the largest loss of data information is incurred when the data-score s_o is small. For example, with series 25 steps long, the relative loss of data information due to clustering compared to the loss of data information due to using MC is 64%, when the data-score is 6, and the cluster-score is 9.83. The same fraction reduces to 32% when the time series are long 50 steps, and to 12% when the series are long 125 steps. With longer series, the relative change of data information is $(-1.13 + 0.06s_o)/s_o$ which is increasing in s_o. In this case, the largest relative change of information is 6%. The results in the last group of conditions, in which we generated an uneven number of time series from eight MCs is indicative of a possible weaknesses of the BCD algorithm. This is the only situation in which the algorithm fails to create the correct number of clusters until the time series are 250 steps long. The assignment of time series to clusters also fails to be optimal until the time series are long enough. The data set contains eighty time series generated in numbers 12, 23, 2, 4, 5, 12, 16, 6 by eight MC transition matrices $P_1, \ldots, P_8$. The algorithm creates 4 clusters when the time series are 25 steps long, 7 clusters when the time series are either 50 or 125 steps long and identifies the correct number of clusters when the time series are 250 steps long.

Table 5 shows the details of the clusters and of the assignment of time series to clusters. When the sequences are very short, the algorithm merges time series generated from matrices

Table 5. Cluster identification and assignments in the experiments. A dot represents a correct cluster identification.

		Time series length			
MC	Time series	25	50	125	250
P_6	47–58	•	•	•	•
P_1	1–12	}	•+ 46	•	•
P_5	42–46		• −46	•	•
P_2	13–35	}	•	•	•
P_4	38–41		•	•	•
P_3	36–37	}	}+ 67	}+ 67	•
P_8	75–80				•
P_7	59–74		•−67	•−67	•
Total time series:		80	80	80	80
Classification errors:		17	4	3	0

Curly brackets put together time series generated from different MCs and assigned to one single cluster by the algorithm. A $+n$ denotes that the time series n is assigned to the wrong cluster, a $-n$ denotes that the times series n is not included in the right cluster.

P_3, P_7 and P_8 in one cluster. Time series generated from P_1 and P_5 are also assigned to one cluster, as are P_2 and P_4. Only the series generated from P_6 are correctly identified. Although the amount of information about the generating process is not sufficient to let the algorithm identify the correct generating MCs, the clusters generated in these conditions have meaningful features. For instance, the matrices P_3 and P_8 have the smallest Kullback-Liebler distance of any pair of matrices (0.75), while the matrices P_7 and P_8 have the second smallest distance (0.83). The matrix P_7 is frequently represented in the data set, sixteen time series generated from it, whereas just two and six time series were generated from P_3 and P_8, respectively. Apparently, the cluster containing the sixteen series generated from P_7 absorbs the series generated from P_3 and P_8. Similarly, the cluster containing the twelve time series generated from P_1 absorbs the five time series generated from P_5, just as the cluster containing the twenty-three time series generated from P_2 absorbs the four series generated from P_4. We note that the Kullback-Liebler distance $D(P_1, P_4)$ is greater than $D(P_1, P_5)$, so that the assignment of time series to clusters reflects features of the generating processes. The only cluster that is correctly identified contains twelve time series generated from P_6. This matrix is maximally different from the seven other matrices. As the time series become longer, the algorithm accuracy increases. With time series of 50 steps, the algorithm is able to distinguish series generated from P_2 and P_4, as well as P_1 and P_5—although series 46 is still assigned to the wrong cluster. Similarly, the time series generated from P_3, P_7 and P_8 are assigned to two clusters. One of these contains all but one series generated from P_7, and this series is assigned to the cluster merging the series generated from P_3 and P_8. This merging is the worst error incurred by the algorithm when the time series are 50 and 125 steps long. For series of length 250, the accuracy of the algorithm is 100%. The scatter plots of the cluster-scores against the data-scores in figure 8 show again

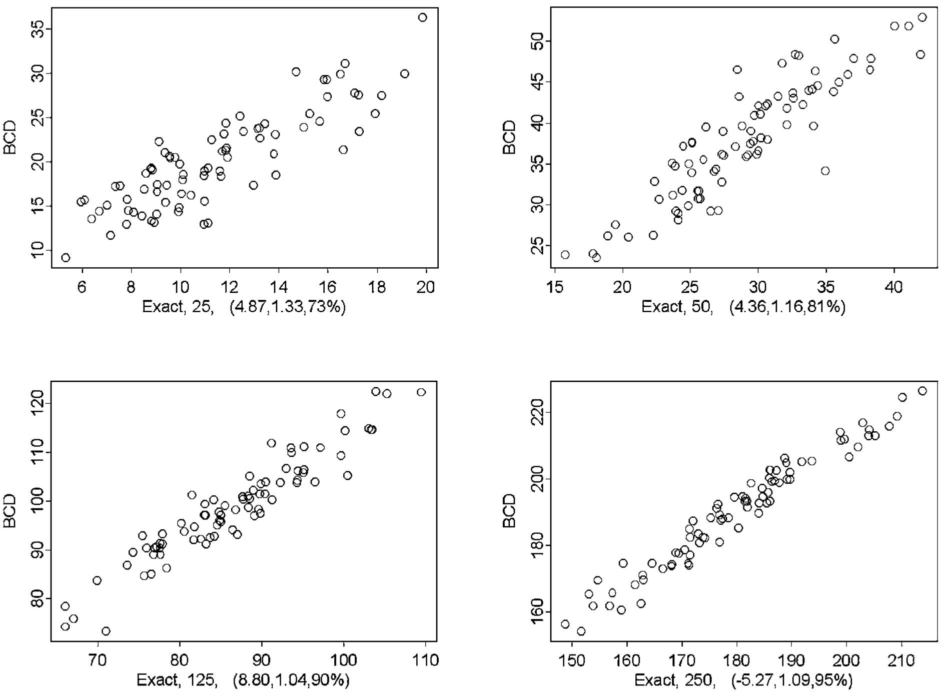

Figure 8. Scatter plot of the data (x-axis) and cluster (y-axis) scores in the group of experiments, in which the time series were generated by eight different MCs, and the MCs are unevenly represented. Values within brackets are respectively the intercept term, the slope and the coefficient of determination of the regression line fitted to each group of scores.

a large linear association, which becomes stronger with the increasing length of the time series. Compared to the previous groups, the cluster-scores appear to be slightly worse than the data-scores when the time series are not enough long. With time series of 25 steps, the regression line of the cluster-scores versus the cumulative data-score has slope 1.33 and intercept term 4.87. Therefore, a data-score of 10 is expected to degrade to 18.17 when the score is computed using the cluster MC. For comparison, in the second experimental group with 80 time series generated by four MCs in different number, a data-score of 10 is expected to deteriorate to 15.46.

Clearly the algorithm can be extremely accurate when the batch of data contain roughly equal numbers of time series generated by different MCs. The fact that series are short does not seem to jeopardize accuracy when only four generating MCs are involved. However, when time series are short and their generating MCs are quite similar, and some of these MCs contribute many more series to a batch of data than the others, then one cluster may be formed from time series from the heavily-represented MC as well as a few series from similar MCs. Our algorithm may not distinguish similar MCs when one contributes many more time series to a batch of data than another, very similar one, and the series are short. This potential problem may be exacerbated by the fact that uniform priors bias the clustering algorithm toward the hypothesis that the time series in a data set are generated from a common MC. However, BCD shows an interesting property, which we will term *monotonic discrimination ability*: when the time series are too short to allow proper discrimination, it groups entire clusters together rather than mixing time series from different clusters. When the time series are long enough to allow discrimination, BCD will properly break down the clusters. This *monotonic discrimination ability* supports the idea that, indeed, short sequences do not really convey enough information to be discriminated and ensures that time series too short will typically result only in a loss of granularity of the partitioning.

5.2. *Experimental comparison of* BCD *and* EMC

The similarity of BCD and EMC discussed in Section 4 calls for a comparison between the two clustering algorithms. Both BCD and EMC model the dynamic of a time series via a MC and then seek the partition of a batch of time series into clusters that maximizes a scoring function. EMC assumes the number of clusters known, while BCD seeks the optimal number of clusters and the optimal partition. Given the methodological similarities, it may seem that EMC should determine the same clusters of MCs as BCD when provided with the number of clusters found by BCD. Furthermore, if some prior knowledge about the number of clusters is available, then EMC would be a faster clustering algorithm than BCD. However, as discussed in Section 4, the scoring function used by BCD—marginal likelihood—seems to be more robust than the one used by EMC—maximum likelihood. The goal of the experimental comparison in this section is to investigate further this claim.

For the comparison, we use the sixteen groups of time series generated for the experimental evaluation in Section 5. On each group, we use the program implementing EMC (Ridgeway & Altschuler, 1998), to find clusters of MCs. Given that EMC finds a pre-specified number of clusters of MCs, in each experimental group we use the algorithm by assuming a number of clusters equal to the number of MCs used to generate the data. With the exception of the last experimental group—eighty time series generated unevenly by eight MCs—this number equals the number of clusters found by BCD. For a fair comparison, in this group we also run EMC with the number of pre-specified clusters equal to the number of clusters found by BCD. We evaluate the results of EMC with two of the three performance measures used in the experimental evaluation: the number of time series correctly assigned to clusters, and the measure of the loss of data information induced by clustering. The first performance measure is computed as in the previous section. To determine the second performance measure, we compute cluster-scores for each time series using the transition probability matrices of the cluster to which the series belongs, in the clustering produced by EMC. The number of time series correctly assigned to clusters compares the accuracy of EMC with that of BCD. Furthermore, the comparison of the cluster-scores shows whether EMC is more or less lossy than BCD.

Table 6 reports the number of time series that EMC correctly assigns to clusters in the sixteen groups, when given the number of clusters found by BCD. In those conditions in which four MCs are used to generate the time series, BCD always finds the correct number of clusters, so that EMC finds the best partition knowing the correct number of generating processes. When the generating MCs are represented equally in the data set, the accuracy of

Table 6. Number of time series correctly assigned to clusters by EMC in the sixteen data sets, when EMC is given the number of clusters found by BCD.

	Number of generated time series per MC							
	Equal proportions				Different proportions			
Length	25	50	125	250	25	50	125	250
4 MCs	79 (79)	79 (80)	80 (80)	80 (80)	74 (80)	71 (80)	76 (80)	62(80)
8 MCs	56 (77)	77 (80)	58 (80)	70 (80)	64 (63)	69 (76)	70 (77)	52(80)

Number in brackets are the time series correctly assigned to clusters by BCD.

EMC and BCD are essentially the same. Only in one case—time series of length 125 generated from four MCs—EMC assigns one time series to the wrong cluster. In the same case, BCD assigns correctly all time series to clusters. When the time series are represented unevenly, however, the accuracy of EMC is inferior to that of BCD. BCD finds the correct number of clusters and assigns correctly time series to the four clusters. Despite the advantage given to EMC, which is provided with the correct number of clusters, some time series are assigned to the wrong cluster. Recall that, in this group, we generated eighty time series from the four MCs with transition probability matrices P_1, P_4, P_5, and P_7. Each of the four MCs generated 18, 10, 33, and 19 time series. The clusters found by EMC, when the length of time series is 25 steps, group the originated time series into $C_1 = \{17 \text{ from } P_1, 2 \text{ from } P_7\}$; $C_2 = \{1 \text{ from } P_1,\ 10 \text{ from } P_4\}$; $C_3 = \{30 \text{ from } P_5\}$; $C_4 = \{3 \text{ from } P_5,\ 17 \text{ from } P_7\}$, for an overall number of six time series assigned to the wrong cluster. These clusters are similar to those found with time series of length 125. In this case, the clusters found by EMC are $C_1 = \{14 \text{ from } P_1\}$; $C_2 = \{4 \text{ from } P_1,\ 10 \text{ from } P_4\}$; $C_3 = \{33 \text{ from } P_5\}$; $C_4 = \{19 \text{ from } P_7\}$, for an overall number of four time series assigned to the wrong cluster. However, the number of errors is larger with time series of length 50, and even larger with time series of length 250. In the first case, the clusters found by EMC are $C_1 = \{17 \text{ from } P_1,\ 8 \text{ from } P_7\}$; $C_2 = \{1 \text{ from } P_1,\ 10 \text{ from } P_4\}$; $C_3 = \{33 \text{ from } P_5\}$; $C_4 = \{19 \text{ from } P_7\}$, for an overall number of nine time series assigned to the wrong cluster. In the second case, the clusters found by EMC are $C_1 = \{18 \text{ from } P_1,\ 10 \text{ from } P_4\}$; $C_2 = \{8 \text{ from } P_7\}$; $C_3 = \{33 \text{ from } P_5\}$; $C_4 = \{11 \text{ from } P_7\}$, for an overall number of eighteen time series assigned to the wrong cluster. It is worth noting here that the first cluster merges the time series generated by two distinct MCs. However, the fact that EMC knows there are 4 clusters to generate, forces the algorithm to split the time series generated by the MC with transition probability matrix P_7 into two groups. These large number of errors results in a loss of data information which is higher than that of BCD. Figure 9 plots the cluster-scores, computed with the cluster transition probability matrices found by EMC, versus the data-scores. Compared to the plots in figure 6, the cluster-scores based on EMC exhibit a larger variability than those based on BCD, smaller coefficients of determination, and result in a larger average loss of data information. The regression line fitted with the group of time series of length 50 in figure 9, for example, has intercept term 5.10 and slope 1.20. Thus, the expected cluster-score is $5.10 + 1.20 s_o$. In the same conditions, the expected cluster-score computed with the cluster

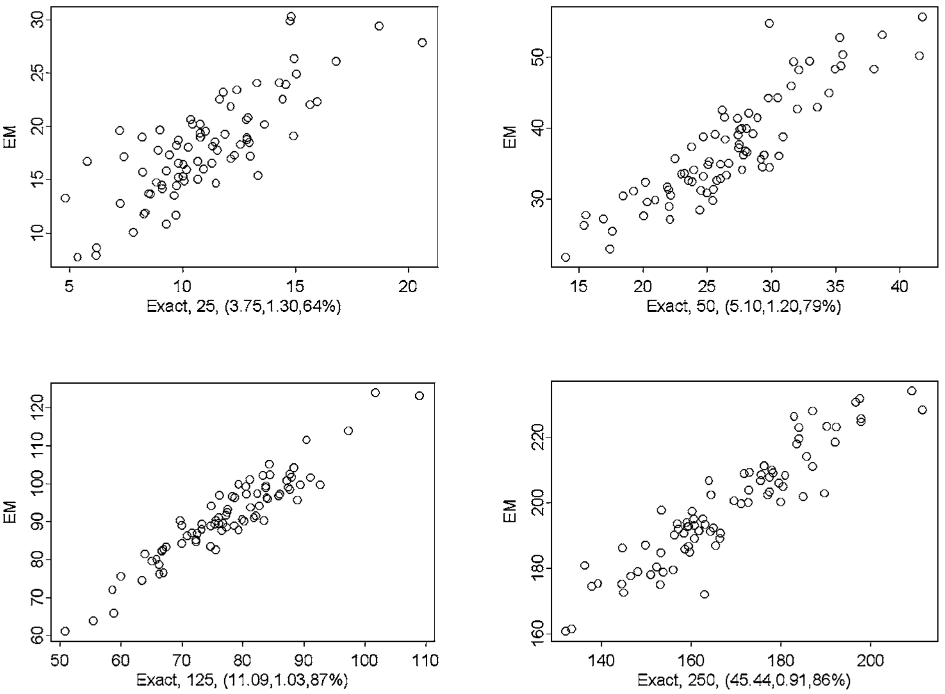

Figure 9. Scatter plot of the data (x-axis) and cluster (y-axis) scores in the group of experiments, in which the time series were generated by four different MCs, and the MCs are unevenly represented. Values within brackets are respectively the intercept term, the slope and the coefficient of determination of the regression line fitted to each group of scores.

transition probability matrices found by BCD is $-1.88 + 1.36s_o$ (from figure 6), and the difference is $5.10 + 1.20s_o + 1.88 - 1.36s_o = 6.98 - 0.16s_o$. This line is positive in the range $0 < s_o < 43.6$, which is the range of observed data-scores, thus showing that the loss of data information induced by EMC is uniformly larger than that induced by BCD. Similar is the result for the group of time series of length 250. The two regression lines for the cluster-scores based on EMC and the cluster-scores based on BCD are $45.44 + 0.91s_o$ and $2.07 + 1.05s_o$, with a difference in expected scores of $45.44 + 0.91s_o - 2.07 + 1.05s_o = 43.37 - 0.14s_o$. Again, this line is positive in the range $0 < s_o < 309.79$ which contains the range of observed data-scores, and hence shows that the loss of data information induced by EMC is uniformly larger than that induced by BCD.

When eight MCs are used to generate the time series, and each MC is equally represented in the batch of eighty, BCD finds the correct number of eight clusters so that, again, EMC finds the best partition knowing the correct number of generating processes. The fact that each MC generates only ten time series in this group has a negative effect on the accuracy of EMC as shown in Table 6. The number of time series assigned to the wrong cluster ranges between 56 and 77, and in none of the four groups EMC finds the correct partition. Recall that, in this group, we generated eighty time series from the eight MCs with transition probability

Table 7. Assignment of time series to cluster by EMC, when the 80 time series are generated by 8 groups of 10 MCs.

	Generating MCs							
EMC	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
C_1	0	4	0	0	0	0	1	0
C_2	0	5	0	0	5	0	0	0
C_3	3	0	7	0	0	0	0	0
C_4	0	1	0	9	0	0	0	0
C_5	0	0	0	0	5	2	0	0
C_6	7	0	0	1	0	8	0	0
C_7	0	0	0	0	0	0	8	0
C_8	0	0	3	0	0	0	1	10
Total	10	10	10	10	10	10	10	10

matrices P_1—P_8, and each of the eight MCs generated ten time series. The clusters found by EMC, when the length of time series is 25 steps, group the originated time series into the eight clusters described in Table 7. Except for cluster C_7, which only contains time series generated by P_7, all the other clusters mix time series generated by two or three different MCs. Results are similar when the time series are long 125 steps, and better when the time series are long 50 steps or 250 steps. Figure 10 plots the cluster-scores, computed with the cluster transition probability matrices found by EMC, versus the data-scores. The scatter plots show that the cluster-scores produced by EMC compare unfavorably to BCD, as their range is larger than the range of cluster-scores induced by BCD. A comparison of the regression lines also shows a systematically larger loss of data information when EMC is used. For example, with eighty time series of length 250, the expected cluster-score of BCD is $-1.13 + 1.06s_o$ and the expected cluster-score of EMC is $35.89 + 0.97s_o$. Their difference is $35.89 + 0.97s_o + 1.13 + 1.06s_o = 37.02 - 0.09$ which is always positive for $0 < s_o < 411.33$. As this range includes the observed range of data-score, the loss of data information of EMC is always larger than the loss of BCD.

In the last experimental set, with eighty time series generated unevenly from the eight MCs with transition probability matrices P_1-P_8, BCD fails to identify the correct number of clusters when the time series are not long enough. Table 3 shows that BCD finds four clusters when the time series are long 25 steps, seven clusters when the time series are long 50 or 125 steps, and identifies the correct number of eight clusters only when the time series are long 250 steps. In this case, we used EMC in two ways. For a direct comparison with the accuracy of BCD, we run the algorithm with the number of clusters found by BCD. The numbers of correct time series assigned to clusters found by EMC in this condition are reported in Table 6, and it is directly comparable to the number of correct time series assigned to clusters by BCD. With time series long 25 steps, EMC clusters correctly 64 time series, while BCD clusters correctly 63 time series. With longer time series, the accuracy of BCD is better than that of EMC and increases with the length of the time series, while EMC

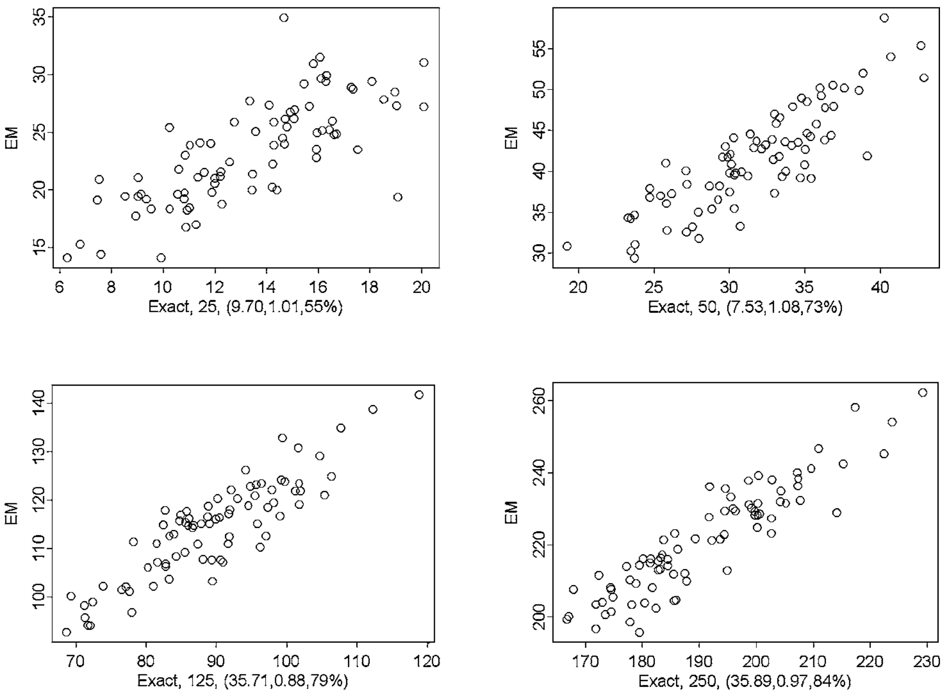

Figure 10. Scatter plot of the data (x-axis) and cluster (y-axis) scores in the group of experiments, in which the time series were generated by eight different MCs, and the MCs are equally represented. Values within brackets are respectively the intercept term, the slope and the coefficient of determination of the regression line fitted to each group of scores.

seems to be incapable of using the larger information provided by the data to gain accuracy. Furthermore, while BCD maintains the correct grouping of time series, except for one or two—time series number 46 and 67 in Table 5—which are assigned to the wrong cluster, EMC mixes a larger number of time series generated by different MCs. For example, when seeking seven clusters of time series in the batch of time series of length 50, EMC merges in one cluster the 23 time series generated by P_2 with the four time series generated by P_4, and assigns to different clusters the time series generated by P_1, P_7 and P_8. Similar is the result with time series of length 125, but the clustering becomes worse with time series of length 250, in which EMC merges into the same cluster time series generated from P_2 and P_7, and merges into the same cluster time series generated from P_1 and P_6. Note that in this case, BCD identifies the correct number of clusters and therefore EMC seeks the correct number of clusters. The inaccuracy of EMC results again in a larger loss of information and the cluster-scores of EMC—figure 11—are worse than the cluster-scores of BCD.

We also run the algorithm by assuming the correct number of eight clusters with the three batches of time series of length 25, 50 and 125. The numbers of correct time series assigned to clusters found by EMC in this condition are 60, when the series are long 25 steps, 57 with series long 50 steps, and 63 with series long 125 steps. Compared to the

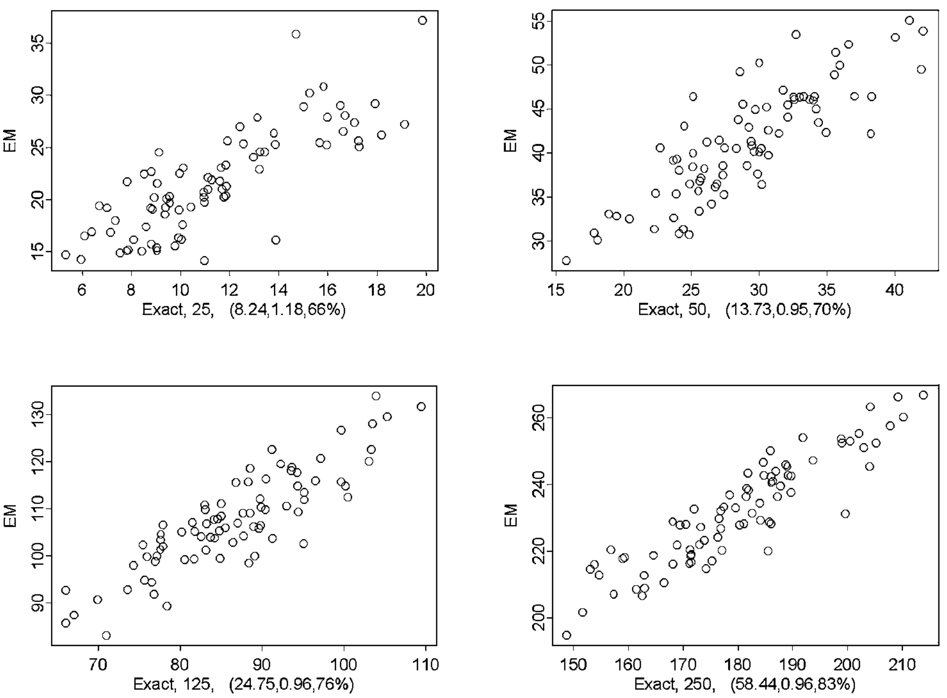

Figure 11. Scatter plot of the data (x-axis) and cluster (y-axis) scores in the group of experiments, in which the time series were unevenly generated by eight different MCs, and EMC was given the number of clusters found by BCD. Values within brackets are respectively the intercept term, the slope and the coefficient of determination of the regression line fitted to each group of scores.

case in which EMC is given the number of clusters found by BCD, accuracy is sensibly less. This surprising result is explained by the fact that, despite EMC knows the correct number of generating MCs, it fails to recover them correctly from the data, and assigns time series generated by the same MC to different clusters. For example, with time series of length 50, three of the eight clusters found by EMC contain only time series generated by the same MC. These are C_1 with five series generated by P_1; C_2 with six time series generated by P_2 and C_7, with five time series generated by P_7. All the remaining clusters contain a mixture of time series generated by two or three different MCs. When the time series are long 125 steps, the number of "pure" clusters rises up to four, and the remaining four clusters mix time series generated by at most two different MCs.

However, all time series generated by P_2 and P_7 are now merged into the same cluster, so that the number of time series mistakenly classified is larger. One possible explanation of this loss of accuracy may be that, indeed, data do not provide enough information to identify the correct number of clusters thus explaining the fact that BCD finds a number of clusters which is inferior to the number of generating MCs. With exception of the first group of 25 steps time series, cluster-scores are more lossy than in the other case. The regression lines are $6.37+1.14s_o$ when the series are 25 steps long, $14.23 + 1.03s_o$,and $30.57 + 0.99s_o$ when the length of the series are 50 and 125 respectively. Direct comparisons of the regression lines shows that, with time series of 25 steps, the reduction in loss of information when EMC assumes the correct number of clusters is $6.37+1.14s_o-8.24-1.18s_o- = -1.87-0.04s_o$. So, compared to the results found by EMC when the number of clusters sought is four, the loss of information is contained, and it is also smaller than that induced by BCD. Compared to BCD, the difference in cluster-scores is $6.37 + 1.14s_o - 4.87 - 1.33s_o = 1.5 - 0.19s_o$, which is negative for $s_o > 7.89$, thus showing the BCD is expected to produce more lossy clusters than EMC. However, with time series long 50 or 125 steps, the loss of information incurred in using EMC with the correct number of clusters is larger than that induced by BCD.

A further difference highlighted by these experiments is that EMC does not enjoy the *monotonic discrimination ability* of BCD. When the sequences are too short to discriminate, they are mixed across the clustering partition with no apparent criterion. Hence, a loss in discrimination ability will not result in a loss of granularity but rather in a set of clusters unrelated to the generating ones.

6. Clusters of sensory inputs

The original motivation behind BCD was the development of a method to enable mobile robots to learn classes of activities without supervision. Our robot—a Pioneer 1—is a small platform with two drive wheels and a trailing caster, and a two degree of freedom paddle gripper. Our configuration of the Pioneer 1 has roughly forty sensors including sonars, gripper and velocity sensors, and a primitive vision system, although the values returned by some sensors are derived from others. During its interaction with the world, the robot records sensors' values every 1/10 of a second, and in an extended period of wandering around the laboratory, it will engage in several different activities—moving toward an object, losing sight of an object, bumping into something—and these activities will have different sensory signatures to be clustered. The goal is to enable the robot to recognize similar activities by the similarity of the dynamics of the sensors' inputs. It is important, to the goals of our project, that the robot's learning should be unsupervised so that we do not tell the robot when it has switched from one activity to another. Instead, we define a simple *event marker*—a simultaneous change of at least three sensors—to segment a time series of sensor values into episodes.

In this section we describe the results obtained with BCD on a data set of 42 episodes for each of the eight sensors comprising left and right wheel velocity, front and rear grippers, bumper, and x, y coordinates and area of an object in the robot visual field. The sensors recording the left and right wheel velocity, as well as the sensors of the vision system take continuous values and were discretized into 5 equal bins of equal length, labeled $1, \ldots, 5$. The data were collected during an experimental trial that lasted about 30 minutes. After discretizing the time series of continuous values, the set of eight time series was segmented into 42 episodes by the event marker. The length of an episode ranges between 6 and 2917 time steps, with an average length of 316 time steps. Our prior hyper-parameters are computed by uniformly distributing the global prior precision. We used $\alpha' = 42$ and $\alpha = 40$.

Figure 12 depicts the MCs representing the two clusters C_1 and C_2 learned for the sensor `vis-a-x`, the horizontal location of an object in the visual field. The first cluster captures

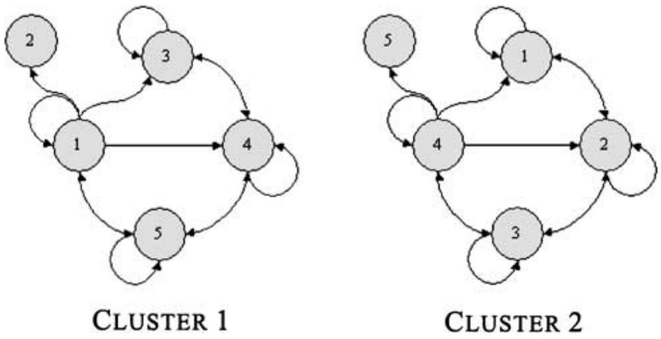

Figure 12. MC representing the first and the second cluster of sequences of sensory inputs.

the sensor dynamics when the robot is not close to an object. The transitions are limited to states 3, 4 and 5 that correspond to the range -28, 140. The initial state 1 can be reached from state 5, which represents the fact that the object appears and disappears from the visual field. However, since the estimate of the probability of transitions 5 → 1 and 1 → 5 are derived from only two cases observed in all the episodes merged into cluster C_1, the confidence in these estimate is very low. The second cluster, on the other hand, represents the sensor dynamics for an object not far from the robot, since transitions are essentially limited among the first 4 states. The prior specification does not rule out the possibility that either state 1 or 5 be reached from state 4. However, in the 12 episodes merged to create cluster C_2, the transitions 4 → 5 and 4 → 1 were never observed, while state 4 was reached only once, from state 3. Figure 13 plots cluster-scores against data-scores in the 42 episodes. Circles represents the scores for the episodes assigned to cluster C_1, and crosses are scores for the episodes assigned to cluster C_2. Both cluster and data-scores are small, thus showing that the abstraction of data into MCs induces a relative small loss of data

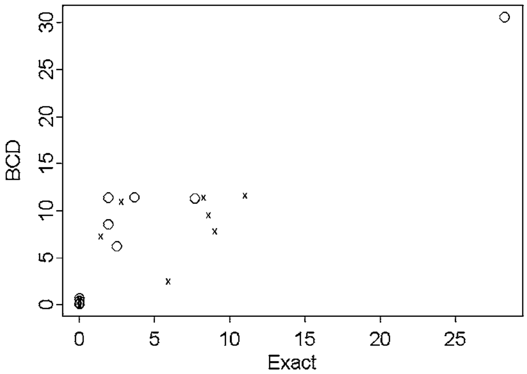

Figure 13. Cluster-scores (y-axis) and data-scores (x-axis) cumulated in the 42 episodes of the robot sensory experience. Circles represents the scores for the episodes assigned to cluster C_1, and crosses are scores for the episodes assigned to cluster C_2.

information, which is then contained when the 42 episodes are merged into two clusters. This second claim is confirmed by the regression line of the cluster-scores versus data-scores which is $s_c = 1.068 + 1.085 s_o$, with a coefficient of determination $R^2 = 82\%$. The individual analysis of the two clusters shows that there is no significant difference between the cluster and data-scores in the two clusters.

Similarly, the 42 episodes for the sensor `vis-a-y`, the vertical location of an object in the visual field, were merged into two clusters both inducing a contained loss of data information. The 42 episodes of the sensor `vis-a-area` recording the area—in number of pixels—of an object in the visual field were merged into six clusters. For example, the MC learned from the episodes assigned to the first cluster represents a dynamic process concentrated on the first three states of the sensor. The first state represents the presence, in the robot's visual field, of an object of size varying between 0 and 1600 pixels, state 2 represents the presence of an object of size between 1600 and 6400 pixels, while state 3 represents the presence of an object of size between 6400 and 14,400 pixels. The maximum size is given by 40,000 pixels so that values between 0 and 14,400 represent an object that, at most, takes 1/4 of the visual field. As the dynamics between these three states is that either the sensor value is constant or decreases because it visits a state preceding itself, the overall dynamics is that of an object of decreasing size in the visual field that eventually disappears. Cluster 2, on the other hand, represents the dynamics of an object of increasing size in the robot visual field, but without reaching the maximum. Episodes in which an object of increasing size is in the robot visual field and terminate with the encounter of the robot with the object are assigned by BCD to a different cluster. Again, merging MCs into clusters induce a small loss of information.

We found three clusters of MCs for both the sensors `l.vel` and `r.vel`—left and right wheel velocity. The three clusters represent dynamics concentrated on null or negative values of the velocity, null or positive values of the velocity and a mixture of those. We found two clusters for the sensor `grip.f`—front gripper—the first one representing a process in which the gripper front beam stays off with high probability and with small probability goes on and stays on, while, in the second one, there is a larger probability of changing from the off to the on state. Hence, the second cluster represents more frequent encounters with an object. The episodes for the sensor `grip.r`—rear gripper—were partitioned in three clusters, one representing rapid changes from the on to the off state, followed by a large probability of staying off; one representing rare changes from the off to the on state, or the other way round, followed by a large probability of staying in that state; the last one representing the sensor in the on state. The episodes for the sensor `grip.b`—grip bumper—were partitioned in two clusters, one representing rare changes from the off to the on state, or the other way round, followed by a large probability of staying in that state; the last one representing the sensor in the on state. So, for example, the first cluster represents the sensor dynamics when the robot is not near an object but, when it does, it pushes it for some time. The second cluster is the sensor dynamics when the robot is pushing an object. Again, in all these cases, merging MCs into clusters induces a small loss of information, although short episodes exhibit large scores.

The clusters found by BCD assign a label to each episode so that, after this initial cluster analysis, the robot can replace each episode with a label representing a combination of the eight sensors' dynamics. Now, episodes labeled with the same combination of sensor clusters represent activities characterized by the same dynamic signature. For example, one such activity is characterized by the combination cluster 1 for `r.vel`, cluster 3 for `l.vel`, cluster 1 for `grip.f, grip.r` and `grip.b`, cluster 2 for `vis.a` and `vis.x` and cluster 1 for `vis.y`. This activity is repeated in 7 of the 42 episodes and represents the robot that rotates and moves far from an object (the velocity of the wheels are discordant, and the size of the object in the visual field decreases and becomes null). Similarly, our robot has learned activities that correspond to passing an object or moving toward an object.

In fact, such activities are learned from the raw sensors' values, assuming that the sensors are independent. We have also used BCD for clustering episodes described by propositions produced by the robot perceptual system (Romani, Sebastiani, & Cohen, 2000a), and since these time series are not independent, we are developing a multivariate version of BCD for clustering multivariate set of time series. Preliminary results are in Ramoni, Sebastiani, and Cohen (2000b).

7. Conclusions

This paper presented an unsupervised Bayesian method to cluster time series and, in general, sequential data. The method recasts the task of clustering time series as a Bayesian model selection problem and searches for the most probable set of clusters given the observed time series. The method is based on an exact—closed-form—scoring function and a heuristic search algorithm, based on the mutual distance between time series, to explore in polynomial time feasible an exponential space of possible partitions. We are currently using this method for the temporal profiling of genomic data and we have found it able to handle thousands of time series on relatively inexpensive equipment. We have reported the results of a controlled experiment showing that high reliability of BCD and the graceful degradation of its performance when the available data do not convey enough information to discriminate among different clusters. This graceful degradation is mainly due to the monotonic discrimination ability of the algorithm which, in absence of sufficient information, tends to group together entire clusters rather than randomly mixing time series. We have compared this algorithm to the approximate, mixture-model method EMC and we have found that EMC is systematically outperformed by BCD and that it does not enjoy a monotonic discrimination ability.

We have also shown the application of BCD to the task of clustering sensory inputs of a mobile robot in order to generate compact representation of the robot's experiences. Many natural processes and engineered artifacts similarly generate masses of time series data. BCD reduces data to a few prototypical time series. Explaining half a dozen clusters is much easier than explaining the original time series. So whether one's task is to cluster robot experiences into an ontology of activities, or to reduce data preparatory to explanation, BCD has a role.

Current limitations of BCD are that it is univariate, and can cluster discrete time series. A multivariate generalization of BCD is outlined in Ramoni, Sebastiani, and Cohen (2000b) while we are currently developing a generalization of the algorithm to cluster time series of continuous values. The intuition behind this development is to model time series of continuous values as autoregressive models and preliminary experiments suggest that the algorithm can be as accurate as BCD.

Acknowledgments

We wish to thank Greg Ridgeway for his generous contribution and for making us available the source code of his program. We are also grateful to Doug Fisher and the anonymous referees for dramatically improving the quality of this paper. This research is supported by DARPA/AFOSR under contract(s) No(s) F49620-97-1-0485. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA/AFOSR or the U.S. Government.

References

Cheeseman, P., & Stutz, J. (1996). Bayesian classification (AutoClass): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (Eds.), *Advances in knowledge discovery and data mining* (pp. 153–180) Cambridge, MA: MIT Press.

Cooper, G. F., & Herskovitz, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning, 9*, 309–347.

Firoiu, L., & Cohen, P. R. (1999). Experiments in abstracting from robot sensor data using hidden markov models. In D. J. Hand, J. N. Kok, & M. Berthold (Eds.), *Advances in intelligent data analysis. Third International Symposium, IDA-99* (pp. 99–110) Springer, NY: New York.

Hand, D. J. (1997). *Construction and assessment of classification rules*. New York, NY: Wiley.

Jaakkola, T. S., Diekhans, M., & Haussler, D. (2000). A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology, 7*, 95–114.

Jeffreys, H. (1946). An invariant form for the prior probability in estimation procedures. *Proceedings of the Royal Society, London, A*, *186*, 453–461.

Lio, P., & Goldman, N. (1998). Models of molecular evolution and phylogeny. *Genome Research, 8*, 1233–1244.

MacDonald, I. L., & Zucchini, W. (1997). *Hidden Markov and other models for discrete-values time series*. London, UK: Chapman and Hall.

Oates, T. (1999). Identifying distinctive subsequences in multivariate time series by clustering. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 322–326) New York: ACM.

Poulsen, C. (1990). Mixed Markov and latent Markov modelling applied to brand choice behavior. *International Journal of Research in Marketing, 7*, 5–19.

Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE*, (Vol. 77, pp. 257–285).

Ramoni, M., & Sebastiani, P. (1999). Bayesian methods. In M. Berthold, & D. J. Hand (Eds.), *Intelligent data analysis. An introduction* (pp. 129–166). New York, NY: Springer.

Ramoni, M., Sebastiani, P., & Cohen, P. (2000a). Unsupervised clustering of robot activities: A Bayesian approach. In *Proceedings of the Fourth International Conference on Autonomous Agents (Agents 2000)* (pp. 134–135). New York, NY: ACM Press.

Ramoni, M., Sebastiani, P., & Cohen, P. R. (2000b). Multivariate clustering by dynamics. In *Proceedings of the 2000 National Conference on Artificial Intelligence (AAAI-2000)* (pp. 633–638). San Francisco, CA: Morgan Kaufmann.

Ridgeway, G. (1997). Finite discrete markov process clustering. Technical Report MSR-TR-97-24, Microsoft Research. Redmond, WA.

Ridgeway, G., & Altschuler, S. (1998). Clustering finite discrete markov chains. In *Proceedings of the Joint Statistical Meetings, Section on Physical and Engineering Sciences* (pp. 228–229). ASA Press.

Ross, S. M. (1996). *Stochastic Processes*. New York, NY: Wiley.

Saul, L. K., & Jordan, M. I. (1999). Mixed memory markov models: Decomposing complex stochastic processes as mixture of simpler ones. *Machine Learning, 37*, 75–87.

Smyth, P. (1999). Probabilistic model-based clustering of multivariate and sequential data. In *Uncertainty 99: Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics* (pp. 299–304). San Francisco, CA: Morgan Kaufmann.

Smyth, P., Heckerman, D., & Jordan, M. (1997). Probabilistic independence networks for hidden Markov probability models. *Neural Computation, 9*, 227–269.

Received February 15, 1999
Revised May 19, 2001
Accepted May 21, 2001
Final manuscript September 23, 2001