Sequence Learning via Bayesian Clustering by Dynamics

Paola Sebastiani[1] **Marco Ramoni**[2] **Paul Cohen**[3]

[1] Department of Mathematics, Imperial College, London, United Kingdom

[2] Knowledge Media Institute, The Open University, Milton Keynes, United Kingdom

[3] Department of Computer Science University of Massachusetts, Amherst, MA

KMi-TR-89

April 3, 2000

Sequence Learning via Bayesian Clustering by Dynamics

Paola Sebastiani[1] **Marco Ramoni**[2] **Paul Cohen**[3]

[1] Department of Mathematics, Imperial College, London, United Kingdom

[2] Knowledge Media Institute, The Open University, Milton Keynes, United Kingdom

[3] Department of Computer Science University of Massachusetts, Amherst, MA

Abstract

This chapter 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 algorithm uses an entropy-based heuristic search strategy. When applied to clustering sensor data from mobile robots, the algorithm produces clusters that are meaningful in the domains of application.

Keywords: Bayesian Learning, Clustering, Time Series, Markov Chains.

Reference: KMi Technical Report KMi-TR-89, Knowledge Media Institute, The Open University, Milton Keynes, United Kingdom, April 3, 2000. Also in *Sequence Learning: Paradigms, Algorithms, and Applications*, L. Giles and R. Sun, editors, Springer, New York, NY, 2000.

Address: Marco Ramoni, Knowledge Media Institute, The Open University, Milton Keynes, United Kingdom MK7 6AA. PHONE: +44 (1908) 655721, FAX: +44 (1908) 653169, EMAIL: `m.ramoni@open.ac.uk`, URL: `http://kmi.open.ac.uk/people/marco`.

1. Introduction

Suppose one has a set of univariate time series generated by one or more unknown processes. The problem we wish to solve is to discover the most probable set of processes generating the data by clustering time series into groups so that the elements of each group 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 [23]. Sound patterns can be clustered by their dynamics, also, and this is one way to discover patterns corresponding to words in the speech signal [16].

The method presented in this chapter transforms each time series into a Markov Chain (MC) and then clusters time series generated by the same MCs. A MC represents a dynamic process as a transition probability matrix. If we regard each time series as being generated by a stochastic variable, we can construct a transition probability matrix for each observed time series. Each row and column in the matrix represents a state of the stochastic variable, and the cell values are the probabilities of transition from one state to each other state of this variable in the next time step. A transition probability matrix is learned for each time series in a training batch of time series. Next, a Bayesian clustering algorithm groups time series that produce similar transition probability matrices. The task of the clustering algorithm is two-fold: to find the set of clusters that gives the best partition of time series according to some measure, and to assign each time series to one cluster.

Clustering can be simply a matter of grouping objects together so that the average similarity of a pair of objects is higher when they are in the same group and low when they are in different groups. Numerous clustering algorithms have been developed along this principle (see [8] for a survey). Our algorithm, called Bayesian Clustering by Dynamics (BCD), does not use a measure of similarity between MCs to decide if two time series need to belong to the same cluster but it uses a different principle: Both the decision of whether to group time series and the stopping criterion are based on the posterior probability of the clustering, that is, the probability of the clustering conditional on the data observed. In other words, two time series are assigned to the same cluster if this operation increases the posterior probability of the clustering, and the algorithm stops when the posterior probability of the clustering is maximum. Said in yet another way, BCD solves a Bayesian model selection problem, where the model it seeks is the most probable partition of time series given the data. To increase efficiency, the algorithm uses an entropy-based heuristic and performs a hill-climbing search through the space of clusterings, so it yields a local-maximum posterior probability clustering.

The algorithm produces a set of clusters, where each cluster is identified by a MC estimated from the time series grouped in the cluster. Given the model-based probabilistic nature of the algorithm, clusters induced from the batch of time series have a probability distribution which can be used for reasoning and prediction so that, for example, one can detect cluster membership of a new time series or forecast future values.

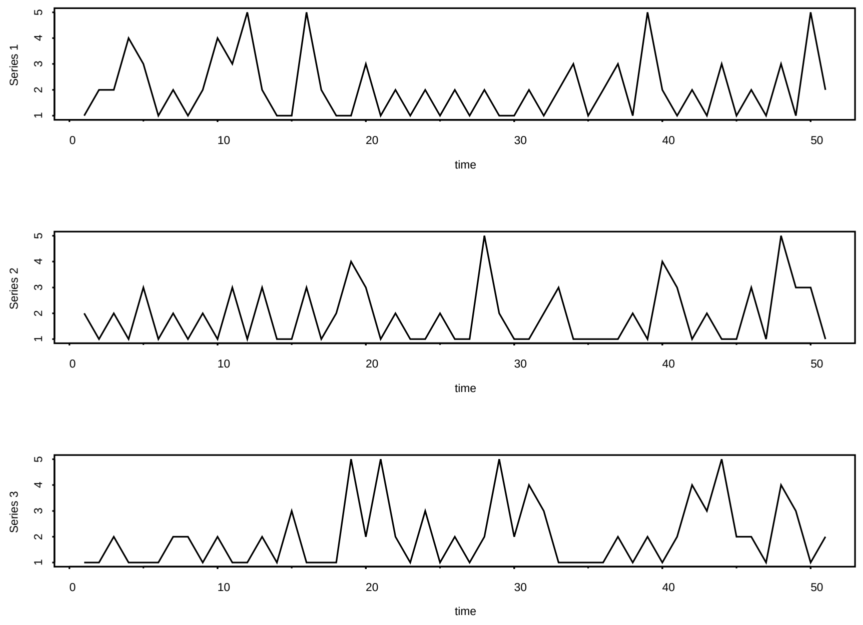


Figure 1: Plot of three time series

Although a MC is a very simple description of a dynamic process, BCD has been applied successfully to cluster robot experiences based on sensory inputs [29, 23], simulated war games [30], as well as the behavior of stocks in market and automated learning and generation of Bach's counterpoint. A conjecture of the success of the algorithm is that describing a dynamic process as a MC can be enough to capture the common dynamics of different time series without resorting to complex models as, for example, Hidden Markov models [21].

The reminder of this chapter is organized as follows. We describe BCD in Section 2. Section 3 shows how to make classification and prediction with clusters of dynamics. An application of the algorithm to cluster robot sensory inputs is in Section 4. Section 5 describes related and future work and conclusions are in the last section of this chapter.

2. Bayesian Clustering by Dynamics

Suppose we are given a batch of m time series that record the values $1, 2, ..., s$ of a variable X. Consider, for example, the plot of three time series in Figure 1. Each time series 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 to identify time series that exhibit similar dynamics. To cluster time series by their dynamics, we model time series as MCs. For each time series, we estimate a transition matrix from data and then we cluster similar transition matrices.

2.1 Learning Markov Chains

Suppose we observe a time series $S = (x_0, x_1, x_2, ..., x_{t-1}, x_t, ..)$, where each x_t is one of the states $1, ..., s$ of a variable X. The process generating the sequence S is a first order MC if the conditional probability that the variable X visits state j at time t, given the sequence $(x_0, x_1, x_2, ..., x_{t-1})$, is only a function of the state visited at time $t-1$ [27]. Hence, we write

$$p(X_t = j|(x_0, x_1, x_2, ..., x_{t-1})) = p(X_t = j|x_{t-1}),$$

with X_t denoting the variable X at time t. In other words, the probability distribution of the variable X at time t is *conditionally independent* of the values $(x_0, x_1, x_2, ..., x_{t-2})$, once we know x_{t-1}. This conditional independence assumption allows us to represent a MC as a vector of probabilities $p_0 = (p_{01}, p_{02}, ..., p_{0s})$, denoting the distribution of X_0 (the initial state of the chain) and a matrix P of transition probabilities, where $p_{ij} = p(X_t = j|X_{t-1} = i)$, so that

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

Given a time series generated by a MC, we wish to estimate the probabilities p_{ij} of state transitions $(i \to j) \equiv (X_{t-1} = i \to X_t = j)$ from the data. This is a well known statistical estimation problem whose solution is to estimate p_{ij} as n_{ij}/n_i, where $n_i = \sum_j n_{ij}$ and n_{ij} is the frequency of the transitions $(i \to j)$ observed in the time series [2]. Briefly, the estimate is found in this way:

1. First, we need to identify the *sampling model*, that is, the probability distribution from which the observed data were generated. Typically, the sampling model is known up to a vector of unknown parameters θ, which we wish to estimate from the data. In our problem, the sampling model is the transition probability matrix P, which is a set of independent discrete distributions, one for each row of P. The vector of parameter θ is given by the set of conditional probabilities (p_{ij}).

2. The sample data S and the sampling model allow us to write down the *likelihood function* $p(S|\theta)$: The probability of the data given the sampling model and θ. Since data are observed, $p(S|\theta)$ is only a function of θ and we estimate θ by finding the value which maximizes the likelihood function. This procedure returns the *Maximum Likelihood estimate* of θ and, hence, the parameter value that makes the observed data most likely.

The assumption that the generating process is a MC implies that transitions from state i of the variable X are independent of transitions from any of the other states of the variables. Therefore, rows of the transition probability matrix P are independent distributions. Data relevant to the ith row distribution are the n_i transitions $(i \to j)$, for any j, observed in the time series and the probability of observing the transitions $(i \to 1)$, $\cdots$, $(i \to s)$ with frequencies $n_{i1}, \cdots, n_{is}$ is $\prod_j p_{ij}^{n_{ij}}$. A discrete random variable with probability mass function proportional to $\prod_j p_{ij}^{n_{ij}}$ has a multinomial distribution [2]. By independence, the likelihood function is the product:

$$p(S|\theta) = \prod_{i=1}^{s} \prod_{j=1}^{s} p_{ij}^{n_{ij}} \tag{1}$$

and depends on the data only via n_{ij}. Maximization of $p(S|\theta)$, with the constraint that $\sum_j p_{ij} = 1$, for all i, returns the estimate $\hat{p}_{ij} = n_{ij}/n_i$.

This estimate uses only the observed data while one may have some prior information about a MC that one wishes to take into account during the estimation of θ. A Bayesian approach provides a formal way to use both prior information and data to estimate θ. This is achieved by regarding θ as a random variable, whose density $p(\theta)$ encodes prior knowledge. Data are used to update the prior density of θ into the posterior density $p(\theta|S)$ by Bayes' Theorem:

$$p(\theta|S) = \frac{p(S|\theta)p(\theta)}{p(S)}$$

and the estimate of θ is the posterior expectation of θ [22].

To choose the prior, we suppose we have some background knowledge that can be represented in terms of a hypothetical time series of length $\alpha - s^2 + 1$ in which the $\alpha - s^2$

$$N = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 3 & 12 & 3 & 0 & 3 \\ 2 & 11 & 1 & 2 & 2 & 0 \\ 3 & 6 & 0 & 0 & 0 & 1 \\ 4 & 0 & 0 & 2 & 0 & 0 \\ 5 & 0 & 4 & 0 & 0 & 0 \end{array} \quad \Rightarrow \quad \hat{P} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0.15 & 0.55 & 0.15 & 0.01 & 0.15 \\ 2 & 0.66 & 0.07 & 0.13 & 0.13 & 0.01 \\ 3 & 0.78 & 0.03 & 0.03 & 0.03 & 0.15 \\ 4 & 0.07 & 0.07 & 0.73 & 0.07 & 0.07 \\ 5 & 0.04 & 0.84 & 0.04 & 0.04 & 0.04 \end{array}$$

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

transitions are divided into $\alpha_{ij} - 1$ 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 $\alpha_{ij} - 1$ that are used to formulate a conjugate prior[1] with density function $p(\theta) \propto \prod_{i=1}^{s} \prod_{j=1}^{s} p_{ij}^{\alpha_{ij}-1}$. This is the density function of s independent *Dirichlet* distributions, with hyper-parameters α_{ij}. Each Dirichlet distribution is a prior to the parameters $p_{i1}, \cdots, p_{is}$ associated with the ith row conditional distribution of the matrix P. Standard notation denotes one Dirichlet distribution associated with the conditional probabilities $(p_{i1}, ..., p_{is})$ by $D(\alpha_{i1}, ..., \alpha_{is})$. The distribution given by independent Dirichlet is called a Hyper-Dirichlet distributions by Dawid and Lauritzen [5] and it is commonly used to model prior knowledge in Bayesian networks [31]. We will denote such a Hyper-Dirichlet distribution by $HD(\alpha_{ij})_s$, where the index s denotes the number of independent Dirichlet distributions defining the Hyper-Dirichlet.

By letting α_i denote $\sum_j \alpha_{ij}$, this prior distribution assigns probability α_{ij}/α_i to the transition $(i \to j)$, with variance $(\alpha_{ij}/\alpha_i)(1-\alpha_{ij}/\alpha_i)/(\alpha_i+1)$. For fixed α_{ij}/α_i, the variance is a decreasing function of α_i and, since small variance implies a large precision about the estimate, α_i is called the *local precision* about the conditional distribution $(p_{i1}, \cdots, p_{is})$ and 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 defining the sampling model. We note that, when α_{ij} is constant, $\alpha_{ij}/\alpha_i = 1/s$ so that all transitions are supposed to be equally likely. Priors with this hyper-parameter specification are known as symmetric priors [10].

A Bayesian estimation of the probabilities p_{ij} is the posterior expectation of p_{ij}. By conjugate analysis [22], the posterior distribution of θ is still Hyper-Dirichlet with updated hyper-parameters $\alpha_{ij} + n_{ij}$ and the posterior expectation of p_{ij} is

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

Equation 2 can be rewritten as

[1]A prior distribution is said to be conjugate when it has the same functional form as the likelihood function

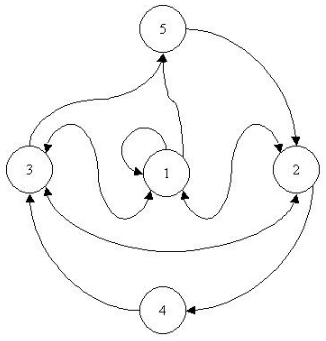


Figure 2: Markov Chain induced from data.

$$\hat{p}_{ij} = \frac{\alpha_{ij}}{\alpha_i}\frac{\alpha_i}{\alpha_i + n_i} + \frac{n_{ij}}{n_i}\frac{n_i}{\alpha_i + n_i} \tag{3}$$

which shows that $\hat{p}_{ij}$ is an average of the estimate n_{ij}/n_i and of the quantity α_{ij}/α_i, with weights depending on α_i and n_i. When $n_i \gg \alpha_i$, the estimate of p_{ij} is approximately n_{ij}/n_i, and the effect of the prior is overcome by data. However, when $n_i \ll \alpha_i$, the prior plays a role. In particular, the Bayesian estimate 2 is never 0 when $n_{ij} = 0$.

Table 1 reports the frequencies of transition n_{ij} $i, j = 1, ..., 5$ 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 frequent transitions between 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 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.

Original Data

Cluster	Time Series
C_1	S_1
C_2	S_2
C_3	S_3
C_4	S_4

Partitions

$M_1 =$

Cluster	Time Series
C_1	S_1
C_1	S_2
C_2	S_3
C_3	S_4

$M_2 =$

Cluster	Time Series
C_1	S_1
C_2	S_2
C_2	S_3
C_1	S_4

$M_3 =$

Cluster	Time Series
C_1	S_1
C_1	S_2
C_1	S_3
C_2	S_4

$$\begin{tikzcd} & M_1 \\ \text{Original Data} \arrow[ur] \arrow[r] \arrow[dr] & M_2 \\ & M_3 \end{tikzcd}$$

Figure 3: Three models corresponding to different re-labeling of a data set of four time series.

2.2 Clustering

The second step of the learning process is an unsupervised agglomerative clustering of time series on the basis of their dynamics. The available data is a set $S = \{S_i\}$ of m time series and the task of the clustering algorithm is two-fold: finding the set of clusters that gives the best partition of the data and assigning each time series S_i to one cluster, without fixing the number of clusters in advance.

Formally, clustering is done by regarding a partition as a discrete variable C with states $C_1, \ldots, C_c$ that are not observed. Each state C_k of the variable C labels, in the same way, time series generated by the same MC with transition probability matrix P_k and, hence, it represents a cluster of time series. The number c of states of the variable C is unknown but it is bounded above by the total number of time series in the data set S: The maximum number of clusters is obtained when each time series is generated by a different MC. Thus, initially, each time series has its own label. The clustering algorithm then tries to relabel those time series that are likely to have been generated by the same MC and thus merges the initial states $C_1, \ldots, C_m$ into a subset $C_1, \ldots, C_c$, with $c \leq m$. Figure 3 provides an example of three different relabelings of a data set S of four time series. Each relabeling determines a model so that, for example, model M_1 is characterized by the variable C having three states C_1, C_2 and C_3 with C_1 labeling time series S_1 and S_2, C_2 labeling S_3 and C_3 labeling S_4. In models M_2 and M_3, the variable C has only two states but they correspond to different labeling of the time series and hence different clusters.

The specification of the number c of states of the variable C and the assignment of one of its states to each time series S_i define a statistical model M_c. Thus, we can regard the clustering task as a Bayesian model selection problem, in which the model we are looking for is the most probable way of re-labeling time series, given the data. We denote by $p(M_c)$ the prior probability of each model M_c and then we use Bayes' Theorem to compute its posterior probability, and we select the model with maximum posterior probability. The posterior probability of M_c, given the sample S, is $p(M_c|S) \propto p(M_c)p(S|M_c)$ where $p(S|M_c)$ is the *marginal likelihood.* This marginal likelihood differs from the likelihood function because it is only a function of the model M_c, while the likelihood function depends on parameters θ quantifying the model M_c. Therefore, the marginal likelihood is computed by averaging out the parameters from the likelihood function. We show next that, under reasonable assumptions on the sample space, the adoption of a particular parameterization for the model M_c and the specification of a conjugate prior lead to a simple, closed-form expression for the marginal likelihood $p(S|M_c)$.

Given a model M_c, that is, a specification of the number of states of the variable C and of the labeling of the original time series (or, equivalently, conditional on the specification of c clusters of time series), we suppose that the marginal distribution of the variable C is multinomial, with cell probabilities $p_k = p(C = C_k|\theta)$. We also suppose that the MCs generating the time series assigned to different clusters C_k are independent, given C, and that time series generated by the same MC are independent. We denote by $P_k = (p_{kij})$ the transition probability matrix of the MC generating time series in cluster C_k and denote the cell probabilities by $p_{kij} = p(X_t|X_{t-1} = i, C_k, \theta)$. Therefore, the overall likelihood function is

$$p(S|\theta) = \prod_{k=1}^{c} p_k^{m_k} \prod_{ij=1}^{s} p_{kij}^{n_{kij}}$$

where n_{kij} denotes the observed frequency of transitions $(i \rightarrow j)$ observed in all time series assigned to cluster C_k, and m_k is the number of time series that are assigned to cluster C_k. Compared to Equation 1, the likelihood is a product of terms corresponding to each cluster and each factor is weighted by the cluster probability $p_k^{m_k}$. We now define a prior density for θ as a product of $c \times s + 1$ Dirichlet densities. One Dirichlet is the prior distribution assigned to (p_k), say $D(\beta_1, ..., \beta_c)$. The other $c \times s$ densities correspond to c independent Hyper-Dirichlet distribution $HD(\alpha_{kij})_s$, each distribution $HD(\alpha_{kij})_s$ being assigned to the parameters p_{kij} of the MC generating the time series in cluster C_k. The marginal likelihood is then given by

$$p(S|M_c) = \int p(S|\theta)p(\theta)d\theta$$

and it is easy to show (by using the same integration techniques in [3]) that

$$p(S|M_c) = \frac{\Gamma(\beta)}{\Gamma(\beta+m)} \prod_{k=1}^{c} \frac{\Gamma(\beta_k + m_k)}{\Gamma(\beta_k)} \prod_{i=1}^{s} \frac{\Gamma(\alpha_{ki})}{\Gamma(\alpha_{ki} + n_{ki})} \prod_{j=1}^{s} \frac{\Gamma(\alpha_{kij} + n_{kij})}{\Gamma(\alpha_{kij})}$$

where $\Gamma(\cdot)$ denotes the Gamma function, $n_{ki} = \sum_j n_{kij}$ is the number of transitions from state i observed in cluster C_k, $\sum_k m_k = m$ and $\beta = \sum_k \beta_k$.

Once the *a posteriori* most likely partition has been selected, the transition probability matrix P_k associated with the cluster C_k can be estimated as

$$\hat{p}_{kij} = \frac{\alpha_{kij} + n_{kij}}{\alpha_{ki} + n_{ki}}$$

and the probability of $C = C_k$ can be estimated as

$$\hat{p}_k = \frac{\beta_k + m_k}{\beta + m}.$$

We conclude this section by suggesting a choice of the hyper-parameters α_{kij} and β_k. We use symmetric prior distributions for all the transition probabilities 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 time series are assigned to the same cluster and the corresponding observed frequencies of transitions are summed up, their hyper-parameters are summed up. Thus, the hyper-parameters of a cluster corresponding to the merging of m_k time series will be $m_k\alpha/(ms^2)$. An alternative solution is to distribute the initial precision α uniformly, across clusters, so that the hyper-parameters of a model with c clusters are $\alpha/(cs^2)$. In both ways, the specification of the prior hyper-parameters requires only the prior global precision α, which measures the confidence in the prior model, and the marginal likelihood of different model is a function of the same α. An analogous procedure can be applied to the hyper-parameters β_k associated with the prior estimates of p_k. Empirical evaluations have shown that the magnitude of the α value has the effect of zooming out differences between dynamics of different time series, so that, increasing the value of α produces an increasing number of clusters.

2.3 A Heuristic Search

To implement the clustering method described in the previous section, we should search all possible partitions and return the one with maximum posterior probability. Unfortunately, the number of possible partitions grows exponentially with the number of time series and a heuristic method is required to make the search feasible.

A good heuristic search could be to merge, or agglomerate, first pairs of time series producing similar transition probability tables. What makes two tables similar? Recall that each row of a transition probability table corresponds to a probability distribution over states at time t given a state at time $t-1$. Let P_1 and P_2 be tables of transition probabilities of two MCs. Because each table is a collection of s row conditional probability distributions, rows with the same index are probability distributions conditional on the same event. The measure of similarity that BCD uses is therefore an average of the Kulback-Liebler distances between row conditional distributions. Let p_{1ij} and p_{2ij} be the probabilities of the transition $X_t = j|X_{t-1} = i$ in P_1 and P_2. The Kulback-Liebler distance of the two probability distributions in row i is $D(p_{1i}, p_{2i}) = \sum_{j=1}^{s} p_{1ij} \log(p_{1ij}/p_{2ij})$. The average distance between P_1 and P_2 is then $D(P_1, P_2) = \sum_i D(p_{1i}, p_{2i})/s$.

Iteratively, BCD computes the set of pairwise distances between the transition probability tables, sorts the generated distances, merges the two time series with closest transition probability tables and evaluates the result. The evaluation asks whether the resulting model M_c, in which two time series are assigned to the same cluster is more probable than the model M_s in which these time series are generated by different MCs, given the data S. If the probability $p(M_c|S)$ is higher than $p(M_s|S)$, BCD updates the set of transition probability tables by replacing them with the table resulting from their merging. Then, BCD updates the set of ordered distances by removing all the ordered pairs involving the merged MCs, and by adding the distances between the new MC and the remaining MCs in the set. The procedure repeats on the new set of MCs. If the probability $p(M_c|S)$ is not higher than $p(M_s|S)$, BCD tries to merge the second best, the third best, and so on, until the set of pairs is empty and, in this case, returns the most probable partition found so far. The rationale of this search is that merging similar MCs first should result in better models and increase the posterior probability sooner thus improving the performance of the hill-climbing algorithm. Empirical evaluations in controlled experiments appear to support this intuition [23]. Note that the similarity measure is just used as a heuristic guide for the search process rather than a grouping criterion.

3. Reasoning and Prediction with Clusters of Dynamics

The BCD algorithm partitions a batch S of m time series into c clusters. Each cluster C_k groups time series generated by the MC with transition probability P_k, which is estimated as $(\hat{p}_{kij}) = (\alpha_{kij} + n_{kij})/(\alpha_{ki} + n_{ki})$. The grouping of time series into clusters provides estimates of the marginal probability that a future time series is generated from the MC with transition probability P_k. This probability is the quantity $\hat{p}_k = (\beta_k + m_k)/(\beta + m)$. The probabilities $\hat{p}_k$ and $\hat{p}_{kij}$ can be used to recognize the cluster from which a new time series is generated by using Bayesian predictive-sequential inference [4].

Suppose we observe a transition (x_0, x_1), with x_0 chosen, and we know that this transition can be generated only from one of the c MCs estimated from the c clusters, and we wish to decide which of the c MCs is more likely to be the generating model. Conditional on the transition (x_0, x_1), we can compute the probability that each of the c MCs is the generating process by using Bayes' Theorem:

$$p(C_k|x_0, x_1) = \frac{p(x_0, x_1|C_k)p(C_k)}{p(x_0, x_1)}.$$

The quantity $p(C_k|x_0, x_1)$ is the probability that the generating process is the MC in cluster C_k, given the transition (x_0, x_1), while $p(x_0, x_1|C_k)$ is the probability of observing the transition (x_0, x_1), given that the generating process is the MC in cluster C_k. Bayes' Theorem lets us update the prior probability $p(C_k)$ into the posterior probability $p(C_k|x_0, x_1)$ via the updating ratio $p(x_0, x_1|C_k)/p(x_0, x_1)$, and the posterior probability distribution over the clusters can be used to choose the MC which, most likely, generated the transition (x_0, x_1).

We only need to compute the updating ratio and, hence, the conditional probability $p(x_0, x_1|C_k)$, for any C_k, and the marginal probability $p(x_0, x_1)$. This marginal probability is computed as

$$p(x_0, x_1) = \sum_k p(x_0, x_1|C_k)p(C_k)$$

so that the crucial quantity to compute remains the conditional probability $p(x_0, x_1|C_k)$. If the value x_0 is chosen deterministically, the probability $p(x_0, x_1|C_k)$ is simply the transition probability $p(x_1|C_k, x_0)$, which, if $x_0 = i$ and $x_1 = j$, is $\hat{p}_{kij}$, and the posterior probability of cluster C_k is

$$p(C_k|x_0, x_1) = \frac{p(x_1|C_k, x_0)p(C_k)}{\sum_k p(x_1|C_k, x_0)p(C_k)}. \tag{4}$$

The updating becomes slightly more complex when more than one transition is observed. Suppose, for example, we observe the sequence $\tilde{S} = (x_0, x_1, x_2)$ and, hence, the pair of transitions (x_0, x_1) and (x_1, x_2). The posterior distribution over the clusters, conditional on $\tilde{S}$, is

$$p(C_k|x_0, x_1, x_2) = \frac{p(x_0, x_1, x_2|C_k)p(C_k)}{p(x_0, x_1, x_2)}$$

and the updating ratio is now $p(x_0, x_1, x_2|C_k)/p(x_0, x_1, x_2)$. As before, the marginal probability $p(x_0, x_1, x_2)$ is $\sum_k p(x_0, x_1, x_2|C_k)p(C_k)$ and the quantity to compute is $p(x_0, x_1, x_2|C_k)$. This probability can be factorized as

$$p(x_0, x_1, x_2|C_k) = p(x_0, x_1|C_k)p(x_2|C_k, x_0, x_1) = p(x_1|C_k, x_0)p(x_2|C_k, x_1). \tag{5}$$

The first simplification $p(x_0, x_1|C_k) = p(x_1|C_k, x_0)$ is the same used above. The second simplification $p(x_2|C_k, x_0, x_1) = p(x_2|C_k, x_1)$ is a consequence of the Markov assumption. Both probabilities are then given by the estimates $\hat{p}_{kij}$ and $\hat{p}_{kjl}$ if $x_0 = i$, $x_1 = j$, and $x_2 = l$.

Simplification of Equation 5 determines this expression for the marginal probability $p(x_0, x_1, x_2)$:

$$p(x_0, x_1, x_2) = \sum_k p(x_0, x_1, x_2|C_k)p(C_k) = \sum_k [p(x_1|C_k, x_0)p(C_k)]p(x_2|C_k, x_1)$$

so that the posterior probability $p(C_k|x_0, x_1, x_2)$ is given by

$$p(C_k|x_0, x_1, x_2) = \frac{[p(x_1|C_k, x_0)p(C_k)]p(x_2|C_k, x_1)}{\sum_k [p(x_1|C_k, x_0)p(C_k)]p(x_2|C_k, x_1)}.$$

Now, from Equation 4, we have that $p(x_1|C_k, x_0)p(C_k)$ is proportional to $p(C_k|x_0, x_1)$, with a proportionality constant independent of k. Therefore, we can rewrite $p(C_k|x_0, x_1, x_2)$ above as

$$p(C_k|x_0, x_1, x_2) = \frac{p(x_2|C_k, x_1)p(C_k|x_0, x_1)}{\sum_k p(x_2|C_k, x_1)p(C_k|x_0, x_1)}$$

which is identical to formula 4 with $p(C_k|x_0, x_1)$ playing the role of $p(C_k)$. This property is the core of the predictive-sequential approach: For any sequence $\tilde{S} = (x_0, x_1, x_2, \cdots)$, the posterior distribution over the clusters can be computed sequentially, by using each transition (x_{t-1}, x_t) in turn to update the current prior distribution into a posterior distribution using formula (4). The posterior distribution will become the prior for the next updating. This result is used in the next section.

4. Bayesian Clustering of Sensory Inputs

This section begins with a description of the robot used in the experiments. It then proceeds by analyzing the application of BCD to the unsupervised generation of a representation of the robot's experiences, and finally shows how to use this abstract representation to enable the robot to classify its current situation and predict its evolution.

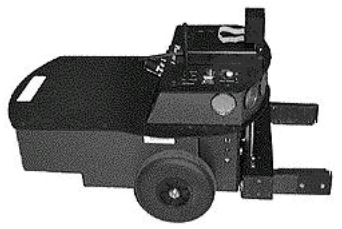


Figure 4: The Pioneer 1 robot.

Sensor name	Interpretation	Range of values
r.vel	velocity of right wheel	-600–600
l.vel	velocity of left wheel	-600–600
grip.f	infra-red sensor at the front of the gripper	0=off; 1=on
grip.r	infra-red sensor at the rear of the gripper	0=off; 1=on
grip.b	bumper sensor	0=off; 1=on
vis.a	number of pixels of object in the visual field	0–40,000
vis.x	horizontal location of object in the visual field	-140 = nearest 140= furthest
vis.y	position of object in the visual field	0 = most left, 256= most right

Table 2: Robot's sensors and their range of values.

4.1 The Robot and Its Sensors

Our robot is the Pioneer 1 depicted in Figure 4. It is a small platform with two drive wheels and a trailing caster, and a two degree of freedom paddle gripper (the two metal arms coming out of the platform). For sensors, the Pioneer 1 has shaft encoders, stall sensors, five forward pointing and two side pointing sonars, bump sensors, and a pair of infra-red sensors at the front and back of its gripper. The bump sensors signal when the robot has touched an object so, for example, they go on when the robot pushes or bumps into something. The infra-red sensors at the front and back of the gripper signal when an object is within the grippers. The robot has also a simple vision system that reports the location and size of colored objects. Our configuration of the Pioneer 1 has roughly forty streams of sensor data, though the values returned by some are derived from others.

We will focus attention on 8 sensors, described in Table 2. Figure 5 shows an example of sensor values recorded during 30 seconds of activity of the robot. The velocity-related sensors r.vel, l.vel, as well as the sensors of the vision system vis.a, vis.x and vis.y take continuous values and were discretized into 5 bins of equal length, labeled between .2 and 1. The vis.a sensor has a highly skewed distribution so that the square root of the original values were discretized. Hence, the category .2 for both sensors r.vel l.vel represents values between -600 and -340, while .4 represents values between -360 and -120, .6 represents values between -120 and 120 and so on. Negative values of the velocities of both wheels represent the robot moving backward, while positive velocities of both wheels represent forward movements. Both negative and positive velocities of the wheels result in the robot turning. The first two plots show sensory values of the left and right wheel velocity from which we can deduce that the robot is probably not moving during the first 5 seconds (steps 1 to 50) or moving slowly (the bin labeled 0.6 represents range of velocity between -120 and 120). After the 5th second, the robot turns (the values of the velocity of the two wheels are discordant) stops and then moves forward, first at low velocity then at increasing velocity until stops, begins moving backward (as the velocity of both wheels is negative) and then forward again. Note that the sensors grip.f and grip.b go on and stay on in the same time interval. Furthermore, the dynamic of the sensor vis.a shows the presence of an object of increasing and decreasing size in the visual field, with maximum size corresponding to the time in which the sensors of the wheel velocity record a change of trend. The trend of the other two sensors of the vision system, vis.x and vis.y, both support the idea that the robot is moving toward an object, bumps into it, and then moves away.

The robot is programmed to engage in several different activities, moving toward an object, loosing sight of an object, bumping into something, and all these activities will have different sensory signatures. If we regard the sequence of sensory inputs of the robot as a time series, different sensory signatures can be identified with different dynamic processes generating the series. It is important to the goals of our project that the robot's learning should be *unsupervised*, which means 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 — and we define an *episode* as the time series between two consecutive event markers. The available data is then a set of episodes for each sensor and the statistical problem is to cluster episodes having the same dynamics. The next section will apply the BCD algorithm to solve this problem.

4.2 Clustering the Robot Sensory Inputs

In this section we describe the results obtained with the BCD algorithm on a data set of 11,118 values recorded, for each of the 8 sensors in Table 2, during an experimental trial that lasted about 30 minutes. The event marker led us to split the original time series into 36 episodes, of average length 316 time steps. The shortest episode was 6 time steps long and the longest episode was 2917 time steps.

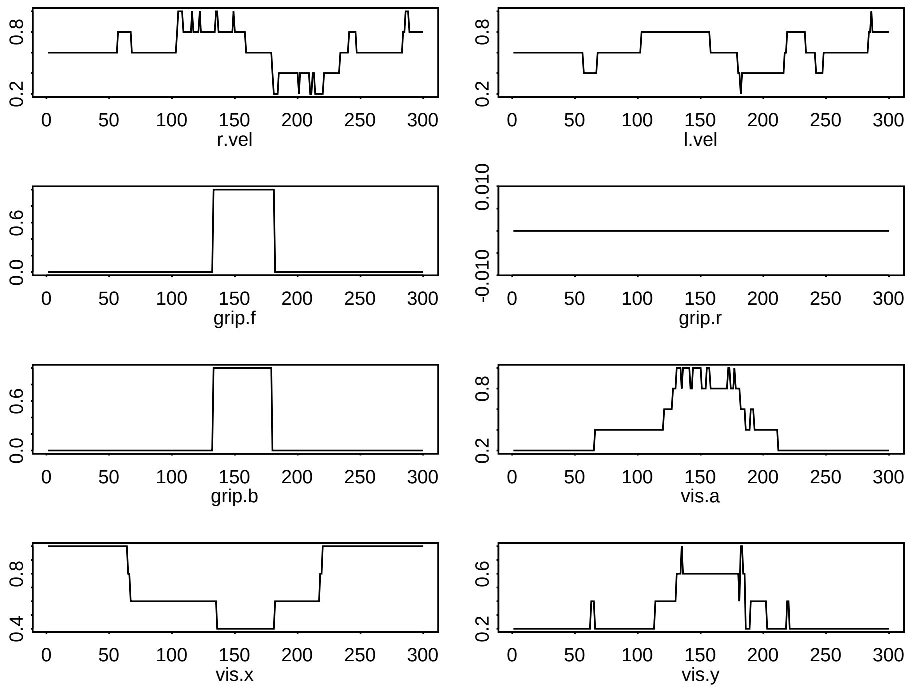


Figure 5: Sensory inputs recorded by the robot during 30 seconds. The x-axes report the time, measured every 1/10 of a second.

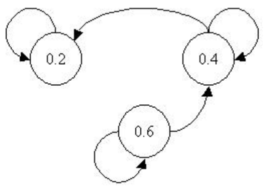
			α		
Sensor	1	5	10	20	40
					11
				18	7
vis.a	34	34	18	13	10
			16	3	3
					3
	2	2	2	2	2

Table 3: Size of clusters found by the BCD algorithm for the sensor vis.a. Brackets before a pair of numbers indicate that the clusters were produced by splitting the episodes belonging to one cluster only for a smaller value of α.

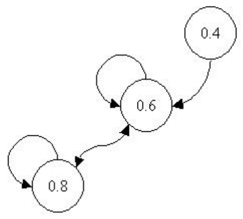
We ran our implementation of the BCD algorithm on the set of 36 episodes for each sensor, using different values for the precision α, while β was set equal to 1. Table 3 shows the number of clusters created by the BCD algorithm for the sensor vis.a, for some of the values of α used in this experiment. A small value of the precision α leads BCD to identify two clusters, one merging 34 episodes, the other merging two episodes. Increasing values of α make the BCD algorithm create an increasing number of clusters by monotonically splitting the cluster merging 34 episodes. For example, this cluster is split into two clusters of 18 and 16 episodes, when $\alpha = 10$. The cluster collecting 16 episodes is then split into two of 13 and 3 episodes, for $\alpha = 20$, and then, when $\alpha = 40$, the cluster of 18 episodes is split into two clusters of 11 and 7 episodes, while the cluster of 13 episodes is split into two clusters of 10 and 3 episodes. Larger values of α make BCD create an even larger number of clusters, some of which represent MC learned from very sparse frequency matrices, so we decide to stop the algorithm for $\alpha = 40$, to avoid overfitting.

The pictures in Table 4 represent the *essential* dynamics of the MCs induced from the BCD algorithm with the 36 episodes of the sensor vis.a. Thus, neither transitions with probability inferior to 0.01 are represented in the chains nor are the uniform transition probabilities of visiting all states. We stress here that the interpretation of the dynamics represented by the MCs is our own one — we looked at the transition probability matrices and labeled the dynamics according to our knowledge of the robot's perception system — and it is by no means knowledge acquired by the robot. So, for example, the first chain represents a dynamic process concentrated on the first three states of the sensor vis.a. State 0.2 represents the presence, in the robot's visual field, of an object of size varying between 0 and 1600 pixels, state 0.4 represents the presence of an object of size between 1600 and 6400 pixels, while state 0.6 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. Now the dynamics between these three states can be that either the sensor value is constant or decreases because it visits a state preceding itself, so that the overall dynamics is that of an object of decreasing size in the visual field that eventually disappears. The interpretation of the other dynamics was deduced in a similar way. Interestingly, the last chain represents essentially a deterministic process, in which the sensor value is constant in state 0.2 showing that there is no object in the robot's visual field.

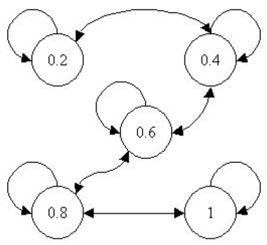
Similar results were found for the other sensors related to the robot's vision system. For example, the values of the sensors vis.x and vis.y produce two clusters for $\alpha = 1$ and eight clusters for $\alpha = 40$. The values of the other sensors tend to produce a smaller number of clusters and to need a much higher prior precision to induce clusters: the minimum value of the precision to obtain at least 2 clusters was 20 for the sensor l.vel; 40 for the sensors r.vel, grip.f and grip.r; and it was 45 for the sensor grip.b. We used the same approach described above to choose a value of α: we run the BCD algorithm for increasing values of α and stopped the algorithm when it began producing MCs with very sparse frequency tables.



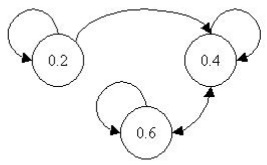
Cluster 1: Object disappearing



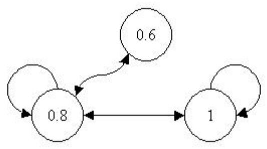
Cluster 2: Object approaching



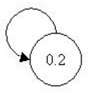
Cluster 3: Object moving back and forth



Cluster 4: Object appearing



Cluster 5: Object approaching



Cluster 6: no object

Table 4: MCs learned with the BCD algorithm from the 36 episodes of the sensor vis.a.

We found three clusters of MCs for both the sensors l.vel and r.vel ($\alpha = 45$ and 50), representing 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 dynamics of sensor grip.f ($\alpha = 40$), 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 were partitioned in three clusters ($\alpha = 40$), 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 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 is, it pushes it for some time. The second cluster is the sensor dynamics when the robot is pushing an object. Finally, the three clusters found for the sensor vis.x ($\alpha = 5$) discriminate the sensor dynamics when an object is far from the robot, or near the robot or a mixture of the two. The four clusters of dynamics for the sensor vis.y ($\alpha = 1$) distinguish among an object moving from the left of the robot's visual field to the center, from the left to the right, from the right to the left, or in front of the robot.

The interpretation of the dynamic processes, represented by the clusters that the BCD algorithms found for each sensor, is our own and not the robot's one. An interesting and still open question is what the robot learned from these clusters. The clusters found by the BCD algorithm 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 8 sensor clusters. Now, episodes labeled with the same combination of sensor clusters represent the same "activity" characterized by the same sensor 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 36 episodes. Using our interpretation of the dynamics represented by the clusters, we can deduce that this activity 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) and hence we have a confirmation that this activity is meaningful. However, as far as the robot's world in concerned, this activity is nothing more than a combination of sensory dynamics.

This process of labeling the episodes in activities by replacing each sensor episode by the cluster membership reduces the initial 36 episodes into 22 different activities, some of which are experienced more than once. Thus, the robot learned 22 different activities characterized by different dynamic signatures.

4.3 Reasoning with Clusters of Dynamics

Results of Section 3 can be used to provide the robot with tools to recognize the cluster it is in, given sensor data. We consider here the six clusters of dynamics learned for the vis.a sensor. From the data in Table 3, one can see that the probability distribution over the 6 clusters in Figure 4 is

C_k	C_1	C_2	C_3	C_4	C_5	C_6
$\hat{p}_k$	0.19	0.30	0.06	0.09	0.09	0.27

This probability distribution is estimated using $\hat{p}_k = (\beta_k + m_k)/(\beta + m)$ with $\beta = 1$ and $\beta_k = 1/6$. Each cluster, in turns, represent a MC with transition probability learned from the time series merged in the cluster. For example, the transition probability of the MC in cluster C_1 is the table

$$\hat{P}_1 = \begin{array}{c|ccccc} & 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ \hline 0.2 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.4 & 0.52 & 0.12 & 0.12 & 0.12 & 0.12 \\ 0.6 & 0.02 & 0.10 & 0.82 & 0.02 & 0.02 \\ 0.8 & 0.20 & 0.20 & 0.20 & 0.20 & 0.20 \\ 1 & 0.20 & 0.20 & 0.20 & 0.20 & 0.20 \end{array}$$

Suppose now we observe the sequence $S = (0.2, 0.2, 0.4, 0.6)$. The posterior distribution over the clusters C_k, conditional on the first transition $(0.2, 0.2)$ is computed using formula 4. Since $\hat{p}_{k11}$ is 1.00 for $k = 1, 2, 6$, 0.20 for $k = 3$, and 0.43 for $k = 4, 5$, there follows that $p(0.2, 0.2) = 0.83$ and $p(0.2, 0.2|C_1)p(C_1) = 0.194$; $p(0.2, 0.2|C_2)p(C_2) = 0.301$; $p(0.2, 0.2|C_3)p(C_3) = 0.012$; $p(0.2, 0.2|C_4)p(C_4) = 0.036$; $p(0.2, 0.2|C_5)p(C_5) = 0.017$; and $p(0.2, 0.2|C_6)p(C_6) = 0.271$. Formula 4 gives the posterior distribution over the clusters

C_k	C_1	C_2	C_3	C_4	C_5	C_6
$p(C_k\|0.2, 0.2)$	0.24	0.36	0.01	0.04	0.02	0.33

This distribution, compared to those learned from the data, assigns more weight to clusters C_1, C_2 and C_6 and the most likely cluster generating transition $(0.2, 0.2)$ appears to be C_2. When we use this updated distribution over the clusters as prior for the next updating, conditional on transition $(0.2, 0.4)$, the distribution turns out to be

C_k	C_1	C_2	C_3	C_4	C_5	C_6
$p(C_k\|0.2, 0.2, 0.4)$	0.00	0.04	0.10	0.67	0.15	0.04

and the most likely cluster is C_4. Indeed, from Figure 4, we see that cluster C_4 assigns high probability to the transition $(0.2, 0.4)$. The next updating produces

C_k	C_1	C_2	C_3	C_4	C_5	C_6
$p(C_k\|0.2, 0.2, 0.4, 0.6)$	0.00	0.00	0.61	0.08	0.24	0.06

so that likely generating clusters are C_3 and C_5. Note that cluster C_3 is the most likely and indeed it is the only cluster that gives high probability to the sequence $(0.2, 0.2, 0.4, 0.6)$.

This procedure can be used to reason with longer sequences so that the robot can recognize the cluster it is in and use this to make decisions.

5. Future and Related Work

We are currently implementing one extension to BCD. First, the BCD algorithm clusters *univariate* time series, but a multivariate version is in the works. Suppose one has a multivariate time series of k variable each of which takes, say, v values. It's trivial to recode this series as one in which a single variable takes v^k values. The difficulty is computational: the transition probability table for the univariate case will have $(v^k)^2$ probabilities to estimate, and will tend to be very sparse unless the time series are quite long. In contrast, the transition probability tables for the k variables in the multivariate case could be quite dense, yielding good probability estimates, with shorter series. The current multivariate extension of BCD makes some assumptions of conditional independence among the variable dynamics so that each cluster becomes a set of MCs with independent transition probability tables.

BCD models time series as first order MCs. More complex models involve the use of k-order Markov chains [28], in which the memory of the time series is extended to a window of k time steps, or Hidden Markov Models [15], 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 [21] and are nowadays applied in many fields ranging from DNA and protein sequencing [13, 14] to robotics [6] and language learning [17]. Despite their popularity, Hidden Markov Models make assumptions that are questionable on the basis of recent results about the identifiability of hidden variables [32] because identifiability of the hidden variables may impose strong constraints on the auto-regressive structure of the series. At first glance it may appear that BCD and Hidden Markov Models are similar technologies, and indeed we have used Hidden Markov Models for some robot learning tasks [7], but they are quite different. BCD approximates the process generating time series with MCs and clusters similar MCs by creating a variable C which represents cluster membership. Conditional on each state of the variable C, the model for the time series is a MC with transition probability $p(X_t = j|X_{t-1} = i, C_k)$. Graphically, this assumption is represented by the model in Figure 6. The oval represents one MC and C separates different ovals, so that, conditional on $C = C_k$ we have $p(X_t = j|X_{t-1} = i, C_k)$ 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

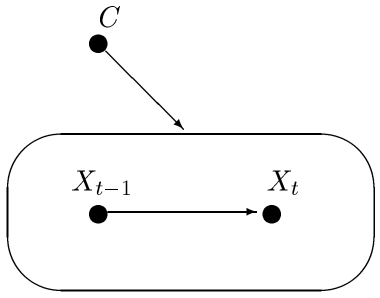


Figure 6: Graphical representation of the model used by BCD.

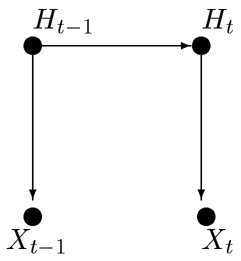


Figure 7: Graphical representation of a Hidden Markov Model.

on the state of the hidden variable makes the dependence of x_t on x_{t-1} vanish. Figure 7 represents graphically this assumption.

A detailed explanation of the difference between a Hidden Markov Model and the model used by BCD is in [35]. Smyth [33] presents an algorithm for clustering Hidden Markov Model and we will compare it with BCD.

The Bayesian modeling-based approach used in BCD is similar to that used, for example, by Raftery [1, 9] to cluster static data. Recent work [24, 34] attempted to extend the idea to dynamic processes without, however, succeeding in finding a closed form solution as the one we have identified. For example, Ridgway [24] proposed using Markov Chain Monte Carlo methods, notorious for their slowness. Smyth [34] proposed a model-based clustering algorithm to cluster time series. Compared to BCD, that algorithm supposes the number of clusters known in advance, while BCD finds the number of clusters. Furthermore, an important novelty of our method is its heuristic search that makes the algorithm feasible.

Methodology aside, BCD is similar in some respects to some other algorithms for clustering time series. To assess the dissimilarity of a pair of multivariate, real-valued time series, Oates applies dynamic time warping to force one series to fit the other as well as possible; the residual lack of fit is a measure of dissimilarity, and with this, Oates can cluster episodes [19]. Rosenstein solves the problem by first detecting events in time series, then measuring the root mean squared difference between values in two series in a window around an event [25, 26]. It is worth noting that these methods and BCD handle time very differently. In Rosenstein's method, two time series are compared moment by moment for a fixed interval. In Oates's approach, one series is stretched and compressed within intervals to make it fit the other as well as possible. The former method keeps time rigid, the latter makes time elastic. If the duration of a sequence within a series is important to the identity of the series — if clustering should respect durations — the former method is probably preferable to the latter. BCD is even more extreme because it transforms a time series, in which durations might be important, into a table of state transitions, which is inherently atemporal. For instance, one cannot tell by looking at a transition table whether a transition $X_t = j|X_{t-1} = i$ occurred before or after a transition $X_{t'} = j|X_{t'-1} = i$.

The problem of finding dependencies between states in time series has been studied by [20, 18, 11, 12]; the current work is unique in its approach to *clustering* time series by the dependency structure that holds among states, and in particular, its ability to differentiate time series that have different dependency structures.

6. Conclusions

This chapter presented BCD a model-based Bayesian algorithm to cluster time series generated by the same process. Although in this chapter we applied BCD to cluster time series of robot's sensory values, the algorithm has been applied successfully to identify prototypical dynamics in simulated war games, music composition and tracking of financial indexes. Our conjecture for this success is that, in those applications, approximating the true generating process by a first order MC was enough to capture the essential dynamics thus providing the algorithm with the information needed to partition the original set of time series into groups. However, approximating a dynamic process by a MC has limitations: for example the order with which transitions are observed in a time series is lost and this may be a serious loss of information. The results given in Section 3 can be used to assess the adequacy of the Markov assumption, by using clusters found by the algorithm to solve some classification or prediction task.

Acknowledgments

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

[1] J. D. Banfield and A. E. Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49:803–821, 1993.

[2] Y. M. M. Bishop, S. E. Fienberg, and P. W. Holland. *Discrete Multivariate Analysis: Theory and Practice.* MIT Press, Cambridge, MA, 1975.

[3] G. F. Cooper and E. Herskovitz. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[4] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Springer, New York, NY, 1999.

[5] A. P. Dawid and S. L. Lauritzen. Hyper Markov laws in the statistical analysis of decomposable graphical models. *Annals of Statistics*, 21:1272–1317, 1993. Correction ibidem, (1995), *23*, 1864.

[6] L. Firoiu and P. R. Cohen. Experiments in abstracting from robot sensor data using hidden markov models. In *The Third Symposium on Intelligent Data Analysis.* Springer, Berlin, 1999.

[7] L. Firoiu, T. Oates, and P. R. Cohen. Learning regular languages from positive evidence. In *Proceedings of the Twentieth Annual Meeting of the Cognitive Scien ce Society*, pages 350–355, 1998.

[8] D. Fisher. Conceptual clustering. In W. Klosgen and J. Zytkow, editors, *Handbook of Data Mining and Knowledge Discovery.* Oxford University Press, 1985.

[9] C. Fraley and A. E. Raftery. How many clusters? Which clustering methods? Answers via model-based cluster analysis. Technical Report 329, Department of Statistics, University of Washington, 1998.

[10] I. J. Good. *The Estimation of Probability: An Essay on Modern Bayesian Methods.* MIT Press, Cambridge, MA, 1968.

[11] A. E. Howe. Analyzing failure recovery to improve planner design. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 387–392. AAAI Press/The

MIT Press, 1992. Also available as University of Massachusetts Computer Science Department Technical Report 92-42.

[12] A. E. Howe and G. Somlo. Modeling discrete event sequences as state transition diagrams. In *Advances in Intelligent Data Analysis: Reasoning About Data. Proceedings of IDA-97.* SpringerVerlag, 1997.

[13] T. S. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 1999. To appear.

[14] P. Lio and N. Goldman. Models of molecular evolution and phylogeny. *Genome Research*, 8:1233–1244, 1998.

[15] I. L. MacDonald and W. Zucchini. *Hidden Markov and other Models for discrete-values Time Series.* Chapman and Hall, London, 1997.

[16] T. Oates. Identifying distinctive subsequences in multivariate time series by clustering. In *Proceedings of the Sixteenth International Conference on Machine Learning.* Morgan Kaufmann Publishers, Inc., 1999. To appear.

[17] T. Oates. Identifying qualitatively different experiences. In *Proceedings of KDD-99, International Conference on Knowledge Discovery and Data Mining*, 1999.

[18] T. Oates and P. R. Cohen. Searching for structure in multiple streams of data. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 346–354. Morgan Kaufmann Publishers, Inc., 1996.

[19] T. Oates, M. D. Schmill, and P. R. Cohen. Identifying qualitatively different experiences: Experiments with a mobile robot. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, San Mateo, CA, 1999. Morgan Kaufman.

[20] T. Oates, M. D. Schmill, D. Jensen, and P. R. Cohen. A family of algorithms for finding temporal structure in data. In *Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics*, pages 371–378, 1997.

[21] L. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[22] M. Ramoni and P. Sebastiani. Bayesian methods. In *Intelligent Data Analysis. An Introduction*, pages 129–166. Physica Verlag, Heidelberg, 1999.

[23] M. Ramoni, P. Sebastiani, P. R. Cohen, J. Warwick, and J. Davis. Bayesian clustering by dynamics. Technical report, Knowledge Media Institute, The Open University, 1999. Available at http://kmi.open.ac.uk/techreports/KMi-TR-??

[24] G. Ridgeway and S. Altschuler. Clustering finite discrete markov chains. In *Proceedings of the Joint Statistical Meetings, Section on Physical and Engineering Sciences.* ASA Press, 1998.

[25] M. Rosenstein and P. Cohen. Concepts from time series. In *Proceedings of the Fiftheen National Conference on Artificial Intelligence*, pages 1808–1814. AAAI Press, 1998.

[26] M. T. Rosenstein and P. R. Cohen. Continuous categories for a mobile robot. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.

[27] S. M. Ross. *Stochastic Processes.* Wiley, New York, NY, 1996.

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

[29] P. Sebastiani, M. Ramoni, and P. Cohen. Bayesian analysis of sensory inputs of a mobile robot. In *Case Studies in Bayesian Statistics.* Springer, New York, NY, 2000.

[30] P. Sebastiani, M. Ramoni, P. Cohen, J. Warwick, and J. Davis. Discovering dynamics using Bayesian clustering. In *Proceedings of the Third International Symposium on Intelligent Data Analysis (IDA-99)*, pages 199–209. Springer, New York, NY, 1999.

[31] P. Sebastiani, M. Ramoni, and A. Crea. Profiling customers from in-house data. In *Proceedings of the 5th Workshop on Case Studies in Bayesian Statistics.* Springer, New York, NY, 2000.

[32] R. Settimi and J. Q. Smith. On the geometry of bayesian graphical models with hidden variables. In *Proceedings of of the Fourteen Conference on Uncertainty in Artificial Intelligence*, pages 472–479, San Mateo, CA, 1998. Morgan Kaufman.

[33] P. Smyth. Clustering sequences with hidden Markov models. In M.C. Mozer, M.I.Jordan, and T.Petsche, editors, *Advances in Neural Information Precessing*, pages 72–93. MIT Press, Cambridge, MA, 1997.

[34] P. Smyth. Probabilistic model-based clustering of multivariate and sequential data. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics (Uncertainty 99)*, pages 299–304. Morgan Kaufman, San Mateo, CA, 1999.

[35] P. Smyth, D. Heckerman, and M. Jordan. Probabilistic independence networks for hidden Markov probability models. *Neural Computation*, 9:227–269, 1997.