Discovering Dynamics using Bayesian Clustering

Paola Sebastiani[1] Marco Ramoni[2] Paul Cohen[3] John Warwick[3] James Davis[3]

[1] Statistics Department, The Open University, Milton Keynes, United Kingdom
[2] Knowledge Media Institute, The Open University, Milton Keynes, United Kingdom
[3] Department of Computer Science, University of Massachusetts, Amherst, MA, USA

Abstract. This paper introduces a Bayesian method for clustering dynamic processes and applies it to the characterization of the dynamics of a military scenario. The method models dynamics as Markov chains and then applies an agglomerative clustering procedure to discover the most probable set of clusters capturing the different dynamics. To increase efficiency, the method uses an entropy-based heuristic search strategy.

1 Introduction

An open problem in exploratory data analysis is to automatically construct explanations of data [5]. This paper takes a step toward automatic explanations of time series data. In particular, we show how to reduce a large batch of time series to a small number of clusters, where each cluster contains time series that have similar dynamics, thus simplifying the task of explaining the data. The method we propose in this paper is a Bayesian algorithm for *clustering by dynamics*.

Suppose one has a set of univariate time series generated by one or more unknown processes, and the processes 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 contains a time series of sistolic and diastolic phases, clustering by dynamics might find clusters corresponding to the pathologies of the heart. If the batch of time series represents sensory experiences of a mobile robot, clustering by dynamics might find clusters corresponding to abstractions of sensory inputs [4].

Our algorithm learns Markov chain (MC) representations of the dynamics in the time series and then clusters similar time series to learn prototype dynamics. A MC represents a dynamic process as a transition probability matrix. For each time series observed on a variable X, we construct one such matrix. Each row in the matrix represents a state of the variable X, and the columns represent the probabilities of transition from that state to each other state of the variable on the next time step. The result is a set of conditional probability distributions, one for each state of the variable X, that can be learned from a time series. A transition 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 reminder of this paper is organized as follows. We first describe the scenario on which we apply our clustering algorithm. The Bayesian clustering algorithm is described in Section 3. We apply the algorithm to a set of 81 time series generated in our application scenario and discuss the results in Section 4.

2 The Problem

The domain of our application is a simulated military scenario. For this work, we employ the *Abstract Force Simulator* (AFS) [1], which has been under development at the University of Massachusetts for several years. AFS uses a set of abstract agents called *blobs* which are described by a small set of physical features, including mass and velocity. A blob is an abstract unit; it could be an army, a soldier, a planet, or a political entity. Every blob has a small set of primitive actions that it can perform, primarily *move* and *apply-force*, to which more advanced actions, such as tactics in the military domain, can be added. AFS operates by iterating over all the units in a simulation at each clock tick and updates their properties and locations based on the forces acting on them. The physics of the world specifies probabilistically the outcomes of unit interactions. By changing the physics of the simulator, a military domain was created for this work.

	Blob	*Task*
Primary Effort	Red 2	**retain** objective **Red Flag**
	Blue 2	**attack** objective **Red Flag**
Supporting Effort	Red 1	**attack** blob Blue 1
	Blue 1	**escort** blob Blue 1

Table 1. The tasks given to each blob in the scenario.

The time series that we want to analyze come from a simple 2-on-2 *Capture the Flag* scenario. In this scenario, the blue team, *Blue 1* and *Blue 2*, attempt to capture the objective *Red Flag*. Defending the objective is the red team, *Red 1* and *Red 2*. The red team must defend the objective for 125 time steps. If the objective has not been captured by the 125^{th} time step, the trial is ended and the red team is awarded a victory. The choice of goals and the number of blobs on each team provide a simple scenario. Each blob is given a task (or tactic) to follow and it will attempt to fulfill the task until it is destroyed or the simulation ends (Table 1).

In this domain, *retaining* requires the blob to maintain a position near the object of the retain — the Red Flag in this example — and protect it from

State #	*State Description*	*Notes*
0	(*F1, FFR+, CFR+*)	Strong Red
1	(*F1, FFR+, CFR-*)	
2	(*F1, FFR-, CFR+*)	
3	(*F1, FFR-, CFR-*)	
4	(*F2, FFR+, CFR+*)	Strong Red
5	(*F2, FFR+, CFR-*)	
6	(*F2, FFR-, CFR+*)	
7	(*F2, FFR-, CFR-*)	Strong Blue
8	(*F3, FFR+, CFR+*)	
9	(*F3, FFR+, CFR-*)	
10	(*F3, FFR-, CFR+*)	
11	(*F3, FFR-, CFR-*)	Strong Blue

Table 2. Univariate representation of the scenario.

the enemy team. When an enemy blob comes within a certain proximity of the object of the retain, the retaining blob will attack it. *Escorting* requires the blob to maintain a position close to the escorted blob and to attack any enemy blob that comes within a certain proximity of the escorted blob. *Attacking* requires the blob to engage the object of the attack without regard to its own state. These tactics remain constant over all trials, but vary in the way they are carried out based on environmental conditions such as mass, velocity and distance of friendly and enemy units. To add further variety to the trials, there are three initial mass values that a blob can be given. With four blobs, there are 81 combinations of these three mass values. At the end of each trial, one of three ending conditions is true:

A The trial ends in less than 125 time steps and the blue team captures the flag.
B The trials ends in less than 125 time steps and the blue team is destroyed.
C The trial is stopped at the 125^{th} time step and the blue fails to complete its goal.

To capture the dynamics of the trials, we chose to define our state space in terms of the number of units engaged and force ratios. There are three possible engagement states at each time step. Red has more blobs "free" or unengaged (*F1*), both blue and red have an equal number of unengaged blobs (*F2*), or blue has more unengaged blobs (*F3*). In each of these states, either the red team or the blue team has more unengaged mass (*FFR+* or *FFR-* respectively). In each of the six possible combinations of the above states, either red or blue has more cumulative mass (*CFR+* or *CFR-* respectively). Altogether there are 12 possible world states, as shown in Table 2. The table shows states 0 and 4 to be especially advantageous for red and states 7 and 11 to be favorable to blue.

In the next section, we represent this set as the states of a univariate variable X, and show how to model the dynamics of each trial and then cluster trials having similar dynamics.

3 Clustering Markov Chains

We describe the algorithm in general terms. Suppose we have a batch of m time series, recording values of a variable X taking values $1, 2, ..., s$. We model the dynamics of each trial as a MC. For each time series, we estimate a transition matrix from data and then we cluster transition matrices with similar dynamics.

3.1 Learning Markov Chains

Suppose we observe a time series $x = (x_0, x_1, x_2, ..., x_{i-1}, x_i, ..)$. The process generating the sequence x is a MC if $p(X = x_t|(x_0, x_1, x_2, ..., x_{t-1})) = p(X = x_t|x_{t-1})$ for any x_t in x [3]. Let X_t be the variable representing the variable values at time t, then X_t is conditionally independent of $X_0, X_1, ..., X_{t-2}$ given 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 of transition probabilities

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

where $p_{ij} = p(X_t = j|X_{t-1} = i)$. Given a time series generated from a MC, we can estimate the probabilities p_{ij} from the data and store them in the matrix 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 informative, 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 frequencies of transitions $n_{ij} = n(i \rightarrow j)$ where, for simplicity, we denote the transition $X_{t-1} = i \rightarrow X_t = j$ by $i \rightarrow j$. 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 in terms of a hypothetical time series of length $\alpha + 1$ in which the α transitions are divided into α_{ij} transitions of type $i \rightarrow j$. 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 Equation 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 Equation 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 $X_t|X_{t-1} = i$ 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.

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 given by the ratio between the number n_{ij} of times the transition has been observed and the number n_i of times the variable has visited state i. In this way, the 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.

3.2 Clustering

The second step of the learning process is an unsupervised agglomerative clustering of MCs on the basis of their dynamics. The available data is a set $S = \{S_i\}$ of m time series. 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 MC to one cluster. A partition is an assignment of MCs to clusters such that each time series belongs to one and only one cluster.

We regard the task of clustering MCs as a Bayesian model selection problem. In this framework, the model we are looking for 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 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, where each state of C represents a cluster of MCs. 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

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

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 $p(M_c|S)$, it is sufficient to consider $p(M_c)p(S|M_c)$. Furthermore, if all models are *a priori* equally likely, the comparison can be based on the *marginal likelihood* $p(S|M_c)$, which is a measure of how likely the data are if the model M_c is true.

The quantity $p(S|M_c)$ can be computed from the marginal distribution (p_k) of C and the conditional distribution (p_{kij}) of $X_t|X_{t-1} = i, C_k$ — where C_k represents the cluster membership of the transition matrix of $X_t|X_{t-1}$ — using a well-known Bayesian method [2]. 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 that are merged into 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 $p(S|M_c)$, which is the product of two components: $f(S, C)$ and $f(S, X_{t-1}, X_t, C)$. 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 uniquely assign each time series to a particular 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. Once created, the transition probability matrix of a cluster C_k — obtained by merging m_k time series — can be estimated as $\hat{p}_{kij} = (\alpha_{kij} + n_{kij})/(\alpha_{ki} + n_{ki})$.

In principle, we just need a search procedure over the set of possible partitions and the posterior probability of each partition as a scoring metric. However, the number of possible partitions grows exponentially with the number of MCs to be considered and, therefore, a heuristic method is required to make the search feasible. The solution we propose is to use a measure of similarity between estimated transition probability matrices to guide the search. Let P_1

and P_2 be transition probability matrices of two MCs. We adopt, as measure of similarity, the average Kulback-Liebler distance between the rows of the two matrices. Let p_{1ij} and p_{2ij} be the probabilities of the transition $i \to j$ in P_1 and P_2. The Kulback-Liebler distance of these two probability distributions is $D(p_{1i}, p_{2i}) = \sum_{j=1}^{s} p_{1ij} \log p_{1ij}/p_{2ij}$ and the average distance between P_1 and P_2 is then $D(P_1, P_2) = \sum_i D(p_{1i}, p_{2i})/s$.

Our algorithm performs a bottom-up search by recursively merging the closest MCs (representing either a cluster or a single trial) and evaluating whether the resulting model is more probable than the model where these MCs are separated. When this is the case, the procedure replaces the two MCs with the cluster resulting from their merging and tries to cluster the next nearest MCs. Otherwise, the algorithm 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 behind this ordering is that merging closer MCs first should result in better models and increase the posterior probability sooner. Note that the agglomerative nature of the clustering procedure spares us the further effort of assigning each single time series to a cluster, because this assignment comes as a side effect of clustering process.

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 the hyper-parameters α_k associated with the prior estimates of p_k. We note that, since $\Gamma(x)$ is defined only for values greater than zero, the hyper-parameters α_{kij} must be non-negative.

4 Clusters of Dynamics

The 81 times series generated with AFS for the Capture the Flag scenario consist of 42 trials in which the blue team captures the red flag (end state A), 17 trials in which the blue forces are defeated (end state B) and 22 which were stopped after 125 time steps (end state C).

We used our clustering algorithm to partition the times series according to the dynamics they represent. A choice of a prior global precision $\alpha = 972$ — corresponding to the initial assignment $\alpha_{kij} = 1/12$ in the 81 transition probability matrices — yields 8 clusters. Table 3 gives the assignment of time series to each of the 8 clusters. By analyzing the dynamics represented by each cluster, it is possible to reconstruct the course of events for each trial. We did this "by hand" to understand and evaluate the clusters, to see whether the algorithm divides the trials in a significant way. We found that, indeed, the

Cluster	*A*	*B*	*C*	*Total*
C_1	5	1	3	9
C_2	2	0	2	4
C_3	7	0	0	7
C_4	14	0	12	26
C_5	1	0	1	2
C_6	8	16	4	28
C_7	2	0	0	2
C_8	3	0	0	3
Total	42	17	22	81

Table 3. Summary of the clusters identified by the algorithm.

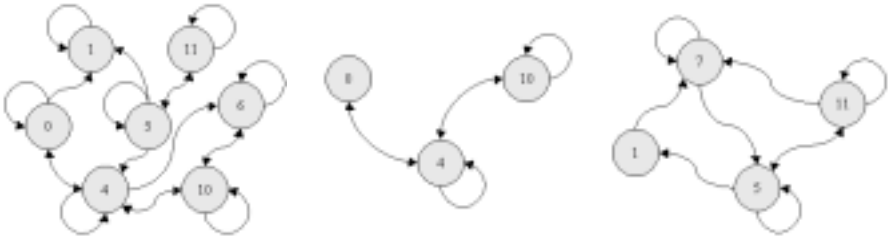
clusters correspond not only to end states, but different prototypical ways in which the end states were reached.

Clusters C_2, C_4 and C_5 consist entirely of trials in which blue captured the flag or time expired (end state A and C). While this may at first be seen as the algorithm's inability to distinguish between the two events, a large majority (though it is not possible to judge how many) of the "time-outs" were caused by the blue team's inability to capitalize on a favorable circumstance. A good example is a situation in which the red team is eliminated, but the blue blobs overlap in their attempt to reach the flag. This causes them to slow to a speed at which they were unable to move to the flag before time expires. Only a handful of "time-outs" represent an encounter in which the red team held the blue team away from the flag. Clusters C_2, C_4 and C_5 demonstrate that the clustering algorithm can identify subtleties in the dynamics of trials, as no information about the end state is provided, implicitly or explicitly, by the world state.

Clusters C_1 and C_6 merge trials of all types. C_1 is an interesting cluster of drawn out encounters in which the advantage changes sides, and blobs engage and disengage much more than in the other clusters. For example, C_1 is the only cluster in which the MC visits all states of the variable and, in particular, is the only cluster in which state 8 is visited. By looking at the transition probabilities, we see that state 8 is more likely to be reached from state 6, and to be followed by state 0. Thus, from a condition of equal free units (*F2*) we move to a situation in which blue disengages a unit and has a free unit advantage (*F3*), which is immediately followed by a situation in which red has a free units advantage (*F1*). The "time-outs" (end state C) in this cluster represent the red team holding off the blue team until time runs out.

Cluster C_6, on the other hand, contains all but one of the trials in which the red team eliminated all of the blue units (end state B), as well as very similar trials where the red blobs appear dominant, but the blue team makes a quick move and grabs the flag. The cluster is characterized by having transitions among states 0, 4 and 10, with a large probability of staying in state 0 (in which the red forces are dominant) when reached. The large number of trials in which the blue team wins (especially large when we realize that C-endings are blue wins

but for the fact that overlapping forces move very slowly) is a result of Blue 1 being tasked to escort Blue 2, a tactic which allows Blue 1 to adapt its actions to a changing environment more readily than other unit's tactics, and in many trials, gives blue a tactical advantage.



Cluster 3 Cluster 7 Cluster 8

Fig. 1. Markov Chains representing clusters C_3, C_7 and C_8.

Clusters C_3, C_7 and C_8 merge only times series of end state A, in which the blue team always captured the flag. Figure 1 displays the MC representing the three clusters (in which we have removed transitions with very low probability). Each cluster captures a different dynamics of how a blue victory was reached. For example, cluster C_8 is characterized by transitions among states 1, 5, 7 and 11 in which the blue team maintains dominance, and transitions to states 4 and 8 — in which the red forces are dominant — are given a very low probability. Indeed, the number of time steps of the trials assigned to cluster C_8 was always low, as the blue team maintained dominance throughout the trials and states 4 and 8 were never visited.

The trials in cluster C_7 visited states 0, 4, and 10 frequently and correspond to cases in which the blue team won despite a large mass deficit. In these cases, the objective was achieved by a break away of one of the blue blobs that outruns the red blobs to capture the flag. The trials assigned to cluster C_7 concluded with victory of the blue team despite a large mass deficit (the objective was achieved by a break away of one of the blue blobs that outruns the red blobs to capture the flag). Cluster C_3 displays transitions among states 0, 1, 4, 5, 6, 10 and 11 and represents longer, more balanced encounters in which the blue team was able to succeed.

5 Conclusions

Our overriding goal is to develop a program that automatically generates explanations of time series data, and this paper takes a step toward this goal by introducing a new method for clustering by dynamics. This method starts by modeling the dynamics as MCs and then applies a Bayesian clustering procedure

to merge these MCs in a smaller set of prototypical dynamics. Explaining half a dozen clusters is much easier than explaining hundreds of time series. Although the explanations offered in this paper are still generated by human analysts — we have not yet achieved fully-automated explanation — the explanatory task is made much easier by our method.

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. This research was developed while Paola Sebastiani and Marco Ramoni were visiting fellows at the Department of Computer Science, University of Massachusetts, Amherst.

References

1. M. Atkin, D. L. Westbrook, P. R. Cohen, and G D. Jorstad. AFS and HAC: The last agent development toolkit you'll ever need. In *Proceedings of the AAAI Workshop on Software Tools for Developing Agents*. American Association for Artificial Intelligence, 1998.
2. G.F. Cooper and E. Herskovitz. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.
3. S.M. Ross. *Stochastic Processes*. Wiley, New York, 1996.
4. P. Sebastiani, M. Ramoni, and P. Cohen. Bayesian clustering of sensory inputs by dynamics. Technical report, Experimental Knowledge Systems Laboratory, University of Massachusetts, 1999.
5. R. St. Amant and P. R. Cohen. Evaluation of a semi-autonomous assistant for exploratory data analysis. In *Proceedings of the First International Conference on Autonomous Agents*, pages 355–362, 1997.