ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Learning predictive partitions for continuous feature spaces

Björn Weghenkel and Laurenz Wiskott

Ruhr-Universität Bochum - Institut für Neuroinformatik
44780 Bochum - Germany

Abstract. Any non-trivial agent (biological or algorithmical) that interacts with its environment needs some representation about its current state. Such a state should enable it to make informed decisions that lead to some desired outcome in the future. In practice, many learning algorithms assume states to come from a discrete set while real-world learning problems often are continuous in nature. We propose an unsupervised learning algorithm that finds discrete partitions of a continuous feature space that are predictive with respect to the future. More precisely, the learned partitions induce a Markov chain on the data with high mutual information between the current state and the next state. Such predictive partitions can serve as an alternative to classical discretization algorithms in cases where the predictable time-structure of the data is of importance.

1 Introduction

Every agent that wants to interact with its environment needs to make some (implicit) assumptions about the environment's future and about the consequences of its actions. Without such assumptions it may as well act randomly and without any strategy. This especially holds for biological agents whose computational resources are constrained but who have sensory inputs that are potentially high-dimensional, redundant, and only partially relevant. As Bialek et al. [1] argued, "*nonpredictive information is useless to the organism*", because it would be outdated already at the time the organism processed the information and is able to react to it. The task of extracting predictable aspects of a signal thus becomes an important problem of neural computation and has been modeled within different frameworks. Building on the concept of information bottlenecks [2], Creutzig et al. [3] has formulated a past-future information bottleneck as a model for how an organism may encode the incoming information that is behaviorally most relevant. For interactive learning scenarios, Still [4] has investigated how the least complex state representation and the least complex policy look like for an agent that has the goal of maximizing its predictive power in the environment. Conceptually also related are *causal states* where two states are mapped to one if they share the same future (see [5] for a short overview of *information bottlenecks*, *causal states*, and *statistical relevance bases*). In the following, by *predictiveness* we refer to the mutual information between current state and next state, given a fixed number of states (see Section 2.1).

We show that the idea of predictiveness can serve as a principled way of discretizing a continuous state-space in the absence of additional information like

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

reward. Our algorithm learns a set of partitions that are effective in predicting the dynamics of the training data. Compared to the frameworks above, our algorithm does not start with a discrete input space but learns partitions for a continuous one. It differs from classical approaches to discretization by explicitly taking into account the time structure of the data. On the other hand it does not need further information like in *Continuous U tree* [6] where partitions are formed for a specific task given as a Markov Decision Process (MDP).

2 Algorithm

In the following we describe the new learning algorithm that discretizes a continuous feature space given a time-series for training. In Section 2.1 we formalize the predictive power of a Markov chain, which will be our main objective. In Section 2.2 we describe how the partitions are organized in a tree structure, and Section 2.3 describes the splitting rule that is used to refine partitions.

2.1 Predictiveness of states

For a given set of states $\mathcal{N} = \{s_1, s_2, \ldots, s_N\}$ we consider a time-homogeneous Markov chain $X_0, X_1, \ldots$ over $\mathcal{N}$. The transition probabilities between states are described by a stochastic transition matrix $\mathbf{P}_{ij} = Prob(X_{t+1} = s_j | X_t = s_i)$ with parameters being the maximum likelihood estimate given the training data. We assume the Markov chain to be irreducible and having a unique stationary distribution π, i.e., $\pi^T = \pi^T \mathbf{P}$.

Our goal is to find a non-trivial partitioning $\epsilon : \mathbb{R}^D \to \mathcal{N}$ that maps a continuous D-dimensional vector space onto a discrete set of states such that the states of the induced Markov chain are as predictive as possible. As a measure of predictability we calculate the mutual information between successive time steps as $I(\mathbf{P}) := I(X_{t+1}; X_t) = h(X_{t+1}) - h(X_{t+1}|X_t) = h(\pi) - \sum_i \pi_i h(X_{t+1}|X_t = s_i) = h(\pi) - H(\mathbf{P})$, where $h(\pi) := -\sum_i \pi_i \log \pi_i$ is the entropy of π and $H(\mathbf{P}) := -\sum_{i,j} \pi_i \mathbf{P}_{ij} \log \mathbf{P}_{ij}$ is the entropy rate (see also [7]). As usual, $0 \log 0 := 0$.

2.2 Partitioning

We organize the partitioning of the feature space as a binary tree $\mathcal{S}$ with nodes either being leaves or comprising a splitting rule $\Gamma(\mathbf{x}) : \mathbb{R}^D \to \{\text{child}_1, \text{child}_2\}$. The leaves of the tree correspond to the learned states $\mathcal{N} = \{s_1, s_2, \ldots, s_N\}$.

States are split successively into new states to increase the mutual information of the model. To that end, for each state s_i a split is calculated, resulting in a candidate for a new partitioning $\mathcal{S}^{(i)}$ with $N + 1$ states and new transition probabilities $\mathbf{P}^{(i)}$. We select the partitioning $\mathcal{S}^* = \arg\max_{\mathcal{S}^{(i)}} (I(\mathbf{P}^{(i)}) - I(\mathbf{P}))$ with the highest gain in mutual information. This step is repeated until the gain falls below a certain threshold or until a desired number of partitions is reached. To avoid overfitting, an existing partition is only split if it contains enough data samples.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

2.3 Splitting of states

There is an infinite number of possibilities how to split an existing state into two new states. The question is: How can we split the state such that it contributes to the predictability of the whole Markov chain, i.e., $I(\mathbf{P})$. To motivate the final algorithm, we start with a related but simpler problem: We assume that instead of the training data we already know the Markov chain with transition matrix $\hat{\mathbf{P}}$ that produced the data. How can this Markov chain be aggregated into just two states, such that these are as predictive as possible? Deng et al. [7] have presented a relaxation to this generally hard bi-partition problem that reduces to a simple eigenvalue problem. Let $\mathbf{\Pi}$ be a matrix with stationary distribution $\check{\pi}$ of $\check{\mathbf{P}}$ on the diagonal and $\mathbf{Q} := \frac{1}{2}(\check{\mathbf{P}} + \mathbf{\Pi}^{-1}\check{\mathbf{P}}^T\mathbf{\Pi})$. Now consider the eigenvalue problem $\mathbf{Qu} = \lambda\mathbf{u}$ and let $\mathbf{u}^{(2)}$ be the solution corresponding to the second largest eigenvalue. Finally, each state of the original Markov chain is assigned to a partition according to the sign structure of the vector $\mathbf{u}^{(2)}$ [7].

The bi-partition algorithm can not be applied directly for splitting a state s_i because we do not know the underlying Markov chain that generated the data $\mathcal{X}^{(i)} := \{\mathbf{x} : \epsilon(\mathbf{x}) = s_i\}$ within that state. We can, however, approximate one from the data observed. We start by building a graph with each data point being one node. Edges are added for every pair of succeeding data points $\{(\mathbf{x}_t, \mathbf{x}_{t+1}) : \mathbf{x}_t, \mathbf{x}_{t+1} \in \mathcal{X}^{(i)}\}$. To generalize the observed dynamics spatially, each node $\mathbf{x}_t$ also inherits the transitions of its k nearest neighbors $\mathcal{K}_t$: $\{(\mathbf{x}_t, \mathbf{x}_{i+1}) : \mathbf{x}_i \in \mathcal{K}_t\}$.

A connection matrix $\mathbf{W}$ is constructed by setting $\mathbf{W}_{i,j} = \mathbf{W}_{j,i} = 1$ for every pair $(\mathbf{x}_i, \mathbf{x}_j)$ of connected nodes. Also, a small constant transition probability of $1e^{-6}$ is added to every possible combination (i, j) to ensure a fully connected graph and a unique stationary distribution.

The resulting matrix $\mathbf{W}$ is normalized row-wise to form a stochastic transition matrix $\hat{\mathbf{P}}$. From that, $\mathbf{Q}$ and $\mathbf{u}^{(2)}$ are calculated as described above and every data point $\mathbf{x}_i$ is labeled and assigned to one of two classes according to the sign of the ith component of $\mathbf{u}^{(2)}$. To be able to generalize to data points not in the training set, we train a classifier Γ on the labeled training data. The decision boundary of the classifier serves as a splitting rule for state s_i.

Algorithm 1

Input: data $\{\mathbf{x}_t\}$, θ_{min_gain}, $\theta_{max\_n\_states}$
initialize partitioning $\mathcal{S}$ with single state s_1
repeat
  for all s_i **do**
    $\mathcal{S}^{(i)} \leftarrow \texttt{SPLIT}(\mathcal{S}, s_i)$
    $gain_i \leftarrow I(\mathcal{S}^{(i)}) - I(\mathcal{S})$
  end for
  $\mathcal{S} \leftarrow \mathcal{S}^{(i)}$ with $i = \text{argmax}_i\, gain_i$
until $max(gain_i) < \theta_{min\_gain}$ or size$(\mathcal{S}) \geq \theta_{max_n_states}$

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Algorithm 2 SPLIT

Input: partitioning $\mathcal{S}$, target state s_n, data $\{\mathbf{x}_t\}$
 # create graph (V, E) for all transitions inside state s_n:
$V \leftarrow \{\mathbf{x}_t, \mathbf{x}_{t+1} : \epsilon(\mathbf{x}_t) = \epsilon(\mathbf{x}_{t+1}) = s_n\}$
$E \leftarrow \{(\mathbf{x}_t, \mathbf{x}_{t+1}), (\mathbf{x}_{t+1}, \mathbf{x}_t) : \epsilon(\mathbf{x}_t) = \epsilon(\mathbf{x}_{t+1}) = s_n\}$
if $|E| <$ min_size: **abort**
 # add transitions of neighbors:
for all $\mathbf{x}_i \in V$ **do**
  $\mathcal{K} \leftarrow k_1$ nearest neighbors of $\mathbf{x}_i$ in V
  for all $\mathbf{x}_j \in \mathcal{K}$ **do**
    $E \leftarrow E \cup \{(\mathbf{x}_i, \mathbf{x}_{j+1}), (\mathbf{x}_{j+1}, \mathbf{x}_i)\}$
  end for
end for
$\mathbf{W} \leftarrow$ connection matrix for graph (V, E)
ensure completely connected graph: $\forall i, j : \mathbf{W}_{i,j} \leftarrow \mathbf{W}_{i,j} + 1e^{-6}$
$\hat{\mathbf{P}} \leftarrow \mathbf{W}$ normalized row-wise
$\pi \leftarrow$ stationary distribution of $\hat{\mathbf{P}}$
$\mathbf{\Pi} \leftarrow \mathrm{diag}(\pi)$
$\mathbf{Q} \leftarrow \frac{1}{2}(\hat{\mathbf{P}} + \mathbf{\Pi}^{-1}\hat{\mathbf{P}}^T\mathbf{\Pi})$
eigendecomposition $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \leftarrow \mathbf{Q}$
train classifier Γ for $\mathbf{x}_i \in V$ with labels from 2nd largest eigenvector: $\mathrm{sign}(\mathbf{u}^{(2)})$
partitioning $\mathcal{S}^{(n)} \leftarrow \mathcal{S}$ with Γ being the splitting rule for state s_n
Output: $\mathcal{S}^{(n)}$

3 Experiments

We conducted two experiments that simulated the movement of an agent through a small, two-dimensional feature space. In every setting we generated 5000 data points and learned a discrete partitioning of the feature space with the algorithm above. For the construction of the graph the size of the neighborhood was set to $k_1 = 5$. As the classifier Γ we used *k-nearest neighbor* with $k_2 = 50$ for every node.

Problem A was generated by performing a random walk along a one-dimensional manifold, namely a spiral. Each new data point was generated by adding Gaussian noise ($\sigma = 0.5$) to the angular value of the last data point. Figure 1.a shows an example of the first 16 of the learned partitions. Figure 2.a shows how the mutual information between present and future increases with the number of partitions. For comparison, the performance of a naive discretization is plotted as well.

Problem B consisted of another random walk in two dimensions. The random walk itself was generated by adding Gaussian noise ($\sigma = 0.1$) to the x-position at every time step. The y-position on the other hand was determined only by uniformly distributed noise and was therefore unpredictable. As can be seen from the example in Figure 1.b, the first eight of the learned partitions focus

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

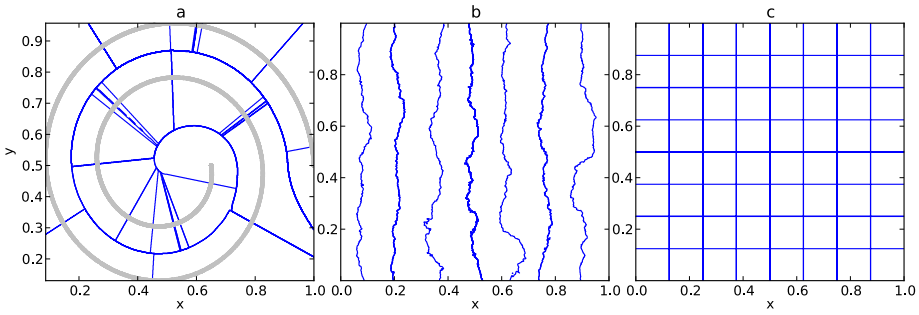

Fig. 1: Examples of learned partitions for two different data sets. (a) and (b) show the first 16 and 8 of the learned partitions on Problem A and Problem B, respectively. (c) shows the naive discretization. Note that some of the boundaries are noisy in (a).

on the predictable aspects of the data by only dividing the feature space along the x-axis, not the noisy y-axis. Only later, when more (useless) partitions are added, is the y-axis divided by different partitions (not shown).

Both experiments show that – compared to naive discretization – a significantly smaller number of partitions is necessary to reach the same amount of predictive power.

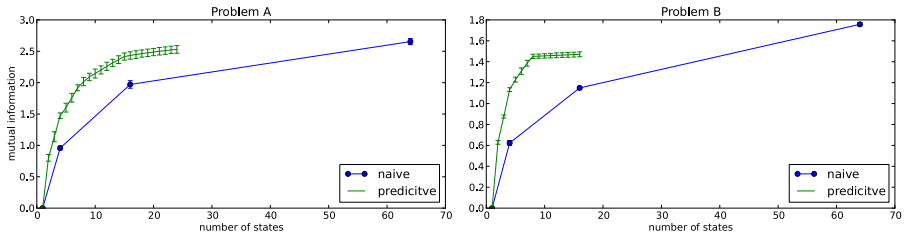

Fig. 2: Mutual information of naive and predictive partitions for problem A ("spiral") and problem B ("noise"). Plotted are the mean and the standard deviation over ten randomly generated datasets each.

4 Conclusion

We have shown how predictability can serve as a principle for learning partitions in the absence of other information like class labels or reward. Our algorithm considers the time structure of the training data to create predictive partitions

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

for a continuous feature space. Experimentally we have shown how the learned partitions ignore aspects of the feature space that are irrelevant for prediction as well as the ability to adapt to a manifold if required by the data.

The scalability of the algorithm mainly depends on the costly eigendecomposition of a connection matrix of data points. Therefore it does not scale well with the number of training points. There are different approaches to solve this issue. First, the graph could be build on only a subset of the training data. Second, the splitting rule of a state could be calculated by (linear) approaches that scale better with the number of training samples. Without giving details here, we particularly hint at Slow Feature Analysis [8] and its connection to the low-dimensional representation of graphs when edges are given by neighborhood in time (similar to the above) [9]. A topic of future research will be the generalization to different actions, i.e., Controlled Markov chains and Markov Decision Processes.

References

[1] William Bialek, Ilya Nemenman, and Naftali Tishby. Predictability, complexity, and learning. *Neural Computation*, 13:2409–2463, Nov 2001.

[2] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. arXiv.org e-Print archive, April 2000. URL `http://arxiv.org/abs/physics/0004057`.

[3] Felix Creutzig, Amir Globerson, and Naftali Tishby. Past-future information bottleneck in dynamical systems. *Physical Review E*, 79:041925, 2009.

[4] Susanne Still. Information-theoretic approach to interactive learning. *EPL (Europhysics Letters)*, 85(2):28005, 2009.

[5] Cosma Rohilla Shalizi and James P. Crutchfield. Information bottlenecks, causal states, and statistical relevance bases: How to represent relevant information in memoryless transduction. *Advances in Complex Systems*, 5(01):91–95, 2002.

[6] William T. B. Uther and Manuela M. Veloso. Tree based discretization for continuous state space reinforcement learning. In *AAAI/IAAI*, pages 769–774. American Association for Artificial Intelligence, 1998. ISBN 0-262-51098-7.

[7] Kun Deng, Prashant G. Mehta, and Sean P. Meyn. Optimal Kullback-Leibler aggregation via spectral theory of Markov chains. *Automatic Control, IEEE Transactions on*, 56(12): 2793–2808, 2011.

[8] Laurenz Wiskott and Terrence Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, April 2002.

[9] Henning Sprekeler. On the relation of slow feature analysis and Laplacian eigenmaps. *Neural Computation*, 23(12):3287–3302, 2011.