Neurocomputing 192 (2016) 49–60

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Temporally extended features in model-based reinforcement learning with partial observability

Robert Lieck*, Marc Toussaint

Universität Stuttgart, Machine Learning and Robotics Lab, Universitätsstraße 38, 70569 Stuttgart, Germany

A R T I C L E   I N F O

Article history:
Received 13 July 2015
Received in revised form
17 November 2015
Accepted 17 December 2015
Available online 3 March 2016

Keywords:
Reinforcement learning
Model learning
Feature learning
Partial observability
Partially observable Markov decision process
Non-Markov decision process

A B S T R A C T

Partial observability poses a major challenge for a reinforcement learning agent since the complete history of observations may be relevant for predicting and acting optimally. This is especially true in the general case where the underlying state space and dynamics are unknown. Existing approaches either try to learn a latent state representation or use decision trees based on the history of observations. In this paper we present a method for explicitly identifying relevant features of the observation history. These *temporally extended features* can be discovered using our Pulse algorithm and used to learn a compact model of the environment. Temporally extended features reveal the temporal structure of the environment while empirically outperforming other history-based approaches.

1. Introduction

In most real-world environments the true (Markov) state cannot directly be observed. For instance we only perceive blurred figures in a haze and cannot discern whether a door is locked or not. This implies that any observation from the past may be relevant in predicting the future and acting optimally. If the underlying latent state space and its dynamics are known – like the fact that a door can be locked using a key – this can be exploited, for instance, by using *belief states* [1]. However, frequently even this information is not available. In this general case, which is our concern in this paper, classical methods for partially observable Markov decision processes (POMDPs) that assume the underlying MDP to be known are not applicable. In this situation, we may either try to identify a latent state space that is sufficient for predicting or acting optimally – as done when learning a *predictive state representation* [2] or a *finite state controller* [3] – or directly work with the given history of observations, as *context tree* methods [4–7] do.

Apart from noisy observations, such as a robot with noisy sensor data, a particularly relevant cause for partial observability are delayed causalities, which means that we only observe the effect of an action while the chain of events leading to this observation remains hidden. In this case, which will be in our focus in this paper, distinct events in the past lead to a specific observation in the future – like switching on the electric kettle leads to boiling water at a later time. This is a scenario of practical relevance, for instance, for household robots, and the key challenge is to identify and represent these delayed causalities based on the observed history.

The aim of our work described in this paper is to find a small set of distinct history-based features that gives structural insight by making this temporal structure explicit and accessible, and allows an intuitive integration of prior domain knowledge – an important means to improve autonomous artificial agents. Since an explicit representation of the temporal structure is not possible when it is concealed within the dynamics of a latent state space we focus on history-based approaches.

Our main contribution is the definition of *temporally extended features* and the associated learning algorithm Pulse (*P*eriodical *U*ncovering of *L*ocal *S*tructure *E*xtensions) on a common basis by using a generating operation N^+ that spans a space of features with increasing complexity and temporal extent tailored to represent delayed causalities. We provide convergence guarantees and use Pulse for solving partially observable domains in a model-based and model-free fashion showing that, in terms of achieved rewards as well as the number of required features, our model-based agent outperforms existing methods.

In the course of this paper, we will first establish the connection to related work on context trees, and feature expansion and selection in Section 2. We will then introduce our method by defining *temporally extended features*, describing our discovery

* Corresponding author.
E-mail address: robert.lieck@ipvs.uni-stuttgart.de (R. Lieck).

http://dx.doi.org/10.1016/j.neucom.2015.12.107
0925-2312/© 2016 Elsevier B.V. All rights reserved.

algorithm Pulse in detail, and discussing convergence properties and the richness of the generated representation in Section 3. In Section 4, we show how Pulse can be used to train reinforcement learning agents. Finally, we discuss our empirical evaluations in Section 5 and conclude with Section 6.

2. Related work

Since we cannot cover the whole breadth of approaches to solving problems with partial observability we once again explicitly state what our focus is on:

Non-MDP versus POMDP: We are dealing with problems where the underlying MDP is *not* known, that is, we are *not* dealing with classical POMDP problems. While this kind of problem is frequently subsumed under the term "POMDP" we think it would actually be more appropriately described by the term non-Markov decision process (non-MDP) since with respect to the observations these problems are *non-Markov* and the potential existence of an underlying MDP is purely hypothetical. In non-MDP problems any methods that assume the underlying MDP to be known – specifically all methods relying on belief states [1] – are *not* applicable.

History-based: We focus on *history-based* approaches because we think in many cases these reveal the structure of the non-MDP more clearly than approaches that learn dynamics in a latent state space. Specifically, this excludes methods like *predictive state representations* [2] or *finite state controller* [3].

Feature-based: We focus on *feature-based* approaches for the same reason. Even from an exceedingly well performing, say, recurrent neural network (RNN) it is often difficult to draw deeper insights into the problem structure. While the branch or research on RNNs applied to reinforcement learning [see e.g. [8]] is highly interesting we will for this reason not discuss these approaches here.

Model-based: While we also present a model-free reinforcement learning (RL) agent and compare against a model-free method, our focus clearly is on *model-based* RL. There are various arguments concerning the general choice between model-based versus model-free RL. The most popular one in favor of model-free RL is that learning a value function is less complex than learning a complete model so that learning rates may be better. The most popular argument in favor of model-based RL is that the transition model is independent of the reward model so that model-based RL generally performs better in case of changing reward functions, that is, in transfer learning where different tasks are to be fulfilled in the same or a closely related environment. We do not intend to joint this general discussion but rather have another, more specific reason to focus on model-based methods, namely that – again – a learned model frequently reveals a more structured insight into the given problem than a value function.

2.1. Context trees

Existing history-based approaches use context trees [4–7] to build a decision tree such that any given history can be associated to a specific leaf node. Based on this classification additional learning steps are performed. Specifically we will use the *utility tree* (U-Tree) algorithm [5] as a reference method in our evaluations. In its original version U-Tree is a model-free method. The leaf nodes take the role of an abstract state to learn an action value function using standard Q-iteration [see for instance [9]]. Learning the action values is interleaved with expanding the decision tree where the Kolmogorov–Smirnov test is used to find the expansion that best discriminates the distribution of action values. U-Tree is easily modified to become model-based, which we do in order to also have a model-based comparison method. The difference is that instead of learning the action value we learn a prediction for the next observation and reward and therefore replace the Kolmogorov–Smirnov test with the chi-square test. To differentiate the two U-Tree versions we write *U-Tree (value)* or *U-Tree* (*model*) from now on. Note, that in both U-Tree versions the classification of histories into leaf nodes, while being sufficient for predicting the action values or the next observation and reward, respectively, is *not* sufficient for predicting the leaf node in the next time step. The leaf nodes thus do not represent the Markov state of the environment even though for U-Tree (model) we can predict the dynamics based on the current history.

Context tree methods and our proposed temporally expanded features have in common that atomic basis features of the history are used. In context trees each internal node corresponds to such a basis feature while in our method we combine multiple basis features to more complex ones (see Section 3 for the details). A closer comparison reveals three major differences:

Feature values: When building a decision tree, the employed basis features have to return discrete output values. This is necessary even if their inputs (i.e. the observations) may be of arbitrary type. In contrast, for our method we use a weighted sum of features allowing for general scalar valued feature outputs, which could not be used in a decision tree.

Descriptive power: Even if using only discrete basis features a set of n temporally extended features still has an exponentially larger descriptive power than a decision tree with n leaf nodes (cf. Section 3.5). This is because multiple features can be active at once in arbitrary combinations where in a decision tree the classification into leaf nodes is mutually exclusive.

Flexibility: A common problem, for instance, in transfer learning tasks or with changing environments, is to adapt an existing model to new data without starting to learn from scratch. In this situation it might be beneficial to drop or change only part of a model. With a decision tree this may become cumbersome since, for instance, changing an internal node influences many leaf nodes at once. In contrast, in a set of temporally extended features we may easily modify or drop one or more features as needed. In fact, this is an inherent part of our learning algorithm Pulse, which restructures the feature set by growing and shrinking it throughout the learning process.

2.2. Feature expansion

Feature expansion techniques were successfully applied to learn (conditional) random field models for text [10,11] and, in reinforcement learning, for linear approximations of the value or transition function [12,13].

The idea of feature expansion is to successively build an increasingly large feature set by using a set of simple basis features and combining them to form more complex features.[1] The basis

[1] The process of choosing features to include into the final set is also called *forward selection* of features, whereof *feature expansion* is a special case.

features are usually binary features that are combined via logical conjunctions to form more complex (again binary) features. This approach is particularly useful if applying feature selection (see Section 2.3) based on the complete feature set is impossible because the number of features is infinite or too large for practical purposes. To construct a feature set with feature expansion, at any time a relatively small set of candidate features is maintained. These candidate features are scored to decide which ones are included in the final set. Since the scoring method alone determines which features are included it is the crucial part in all feature expansion methods.

If the feature weights are optimized "offline" (that is, for a fixed data set as e.g. in [10,11]) a candidate feature can be scored by temporally including it into the set, then optimizing its weight (while keeping the other weights fixed to speed up the process), and using the increase in data likelihood as score value. The highest scored features are then successively included to build the final feature set.

If the feature weights are incrementally adapted in an "online" setting (that is, as new data are continuously coming in as e.g. in [12,13]) the feature scores, too, are updated in each step. To this end, the score of all "active" candidate features (that is, those that would change the model for the current data point if they were included) is incremented by some error metric of the current model – like the temporal difference error in value function approximation. Candidate features are added to the final feature set as soon as their score surpasses a fixed threshold.

The approach we present in this paper can be seen as a kind of feature expansion method since we also construct complex features from simple basis features. However, there are two major difference with respect to existing feature expansion methods:

1. Existing methods comprise two separate parts, one for including candidate features – the scoring heuristic – and one for learning the feature weights of the resulting feature set, which follows an objective that is distinct from the scoring method.
2. Existing methods grow a feature set of increasing size without providing a sound mechanism for removing existing features in case they turn out to be sub-optimal.

We address both issues at the same time by *including and excluding* features solely based on the objective function that is also used to learn the model. The resulting advantage over existing feature expansion methods is that we do not need to come up with a scoring heuristic that may be inconsistent with the objective function and our method may produce smaller feature sets by excluding superfluous features.

2.3. Feature selection

The idea of feature selection is to start with a large feature set that contains all features possibly being relevant and then successively remove features.[2] The most common way of finding features to remove is to use an L_1-regularized objective and to remove all features that end up having zero weight after the optimization. In terms of the general idea, most closely related to our work is the strand of research on L_1-regularized temporal difference learning [14–18] that mostly deals with the question of how to efficiently optimize the L_1-regularized objective.

The main difference in our approach is that we cannot apply feature selection to our problem in a "brute-force" manner because we have intractably large or even infinite feature sets and can therefore not start with a complete feature set containing all possible features. Instead we use feature selection as a subroutine to discard some of the candidate features in a way that makes both the overall algorithm as well as the feature selection step more efficient (see Section 3.2 for the details).

To sum up, one can say that our Pulse algorithm joins the concepts of *feature expansion* and *feature selection* on a common basis by using an expansion operation N^+ in conjunction with an L_1-regularized objective to both grow *and* shrink the feature set in each iteration. As a result the shortcomings of both approaches – growing indefinitely large feature sets and using a possibly inconsistent scoring heuristic for *feature expansion*; having to solve intractable optimization problems due to huge initial feature sets for *feature selection* – are solved or at least mitigated.

[2] In this paper, we use the term *feature selection* in this specific sense, which is also called *backward selection* of features and must not be confused with *forward selection* methods, such as *feature expansion*.

3. Discovering temporally extended features

3.1. Temporally extended features

In a non-Markov decision process the next observation $o \in \mathcal{O}$ and reward $r \in \mathcal{R} \subseteq \mathbb{R}$ depends not only on the last action $a \in \mathcal{A}$ but also on the entire history $h \in \mathcal{H}$ up to that point, where $\mathcal{H} = (\mathcal{A} \times \mathcal{O} \times \mathcal{R})^*$ is the set of all possible sequences of action-observation-reward triplets. The set $\mathcal{T}$ of *temporally extended features* then is the infinite set of all maps

$$\mathcal{T} := \{\mathcal{H} \to \mathbb{R}\} \tag{1}$$

from histories to the real numbers. Since in its generality this set is of little avail we will in practice always work with a subset $\mathcal{T}_{N^+} \subseteq \mathcal{T}$ that is generated by an operation

$$N^+ : \mathcal{P}(\mathcal{T}) \to \mathcal{P}(\mathcal{T}) \tag{2}$$

where $\mathcal{P}(\mathcal{T})$ is the power set of $\mathcal{T}$. That is, N^+ takes a subset of $\mathcal{T}$ and returns another subset of $\mathcal{T}$. More precisely, $\mathcal{T}_{N^+}$ then is the smallest subset of $\mathcal{T}$ that is closed under applying N^+ on any of its subsets

$$\mathcal{T}_{N^+} := \min\left\{\mathcal{F}' \subseteq \mathcal{T} \mid \forall_{\mathcal{F}'' \subseteq \mathcal{F}'} N^+(\mathcal{F}'') \subseteq \mathcal{F}'\right\}. \tag{3}$$

This definition reflects the workings of our Pulse algorithm (detailed in Section 3.2), which shapes the feature set by performing an expansion operation using N^+ and a selection operation using the objective function in an alternating manner. In Eq. (3), the min-operation originates in the fact that Pulse starts with an empty features set; the universal quantifier $\forall_{\mathcal{F}'' \subseteq \mathcal{F}'}$ is due to the fact that the subset $\mathcal{F}''$ selected by optimizing the objective function is not known in advance; and $N^+(\mathcal{F}'')$ corresponds to the expansion of $\mathcal{F}''$ based N^+. While $\mathcal{T}_{N^+}$ may still be infinitely large, at any time Pulse will only use a finite subset (provided N^+ is suitably defined, as detailed in the next section), so that temporally extended features become practically applicable.

3.1.1. Defining N^+

In our Pulse algorithm N^+ takes the role of the feature expansion operation by using $N^+(\mathcal{F})$ as the set of new candidate features for a given set $\mathcal{F}$. The only general requirement therefore is that $N^+(\mathcal{F})$ be a finite set. The specific choice for N^+ may depend on the problem at hand as well as the specific learning method (i.e. the kind of model and objective) being used. As already mentioned in the introduction we will focus on delayed causalities as the source of partial observability. This means that if we aim at predicting a specific observation we expect large parts of the history to be irrelevant and having to take only single, distinct events into account – the events that actually cause the observation. In our definition of N^+ we therefore adopt the approach of feature expansion methods and use a set of basis features $\mathcal{B}$ with each basis feature indicating whether a specific event took place in the past. In our case an "event" is the occurrence of a specific action, observation, or reward at a specific point in time. For a given feature set $\mathcal{F}$, N^+ now constructs a set of candidate features as conjunctions of an existing feature $f \in \mathcal{F}$ and a basis feature $b \in \mathcal{B}$, that is,

$$N^+(\mathcal{F}) = \left\{ g \in \mathcal{T} \mid \exists_{\substack{f \in \mathcal{F} \\ b \in \mathcal{B}}} : g = f \wedge b \right\}. \tag{4}$$

$\mathcal{B}$ is formally defined as

$$\begin{aligned} \mathcal{B} &= \mathcal{B}_{\mathcal{A}} \cup \mathcal{B}_{\mathcal{O}} \cup \mathcal{B}_{\mathcal{R}} \quad \text{with} \\ \mathcal{B}_{\mathcal{A}} &= \left\{ f_{a',t} \in \mathcal{T} \mid \exists_{a' \in \mathcal{A}, t \in \mathbb{Z}_0^-} \; f_{a',t} = \mathbb{I}(a', h_{[a,t]}) \right\} \\ \mathcal{B}_{\mathcal{O}} &= \left\{ f_{o',t} \in \mathcal{T} \mid \exists_{o' \in \mathcal{O}, t \in \mathbb{Z}_0^-} \; f_{o',t} = \mathbb{I}(o', h_{[o,t]}) \right\} \\ \mathcal{B}_{\mathcal{R}} &= \left\{ f_{r',t} \in \mathcal{T} \mid \exists_{r' \in \mathcal{R}, t \in \mathbb{Z}_0^-} \; f_{r',t} = \mathbb{I}(r', h_{[r,t]}) \right\} \end{aligned} \tag{5}$$

where $\mathbb{I}(\cdot,\cdot)$ is the indicator function that takes a value of one if the two arguments are equal and zero otherwise, and $h_{[a,t]}$, $h_{[o,t]}$ and $h_{[r,t]}$ denote the action, observation and reward at time t in history h. That is, for instance, feature $f_{a,t}$ indicates whether action a was performed at time t, where $t = -2$ would refer to entries two time steps ago and $t=0$ to the action (or hypothetical observation or reward) that is about to come next.

There are some special cases and additional details to consider:

Gradual temporal extension: To introduce a bias towards the near past and obtain a finite set of candidate features N^+ only uses basis features that go one step further into the past than the existing features. That is, if we always added all candidate features to our set and performed n iterations of feature expansion we would look n steps farther into the past.

Fixed horizon: If we know that looking back k steps into the past is enough for optimal performance (as is the case in our experiments in Section 5) we put a hard constraint of $t_{min} = -k$ on the basis features.

Empty feature sets: If the current feature set is empty N^+ cannot form conjunctions to use as candidate features and instead returns all basis features separately.

Model type: Depending on what we aim to learn using the temporally extended features the basis features with time index $t=0$ need to be treated in a special way. These basis features indicate the next action, observation, and reward – the action being chosen by the agent and the observation and reward being the hypothetical response of the environment. If we learn a probabilistic model all basis features with $t=0$ are needed. If, on the other hand, we learn an approximation of the action value function only action features with $t=0$ are allowed. And in the case of U-Tree, which we use as comparison method with the same basis features, features are only used to classify the history, that is, basis features with $t=0$ are not used at all.

Note that the definition of N^+ plays a crucial role since at this point most of the prior knowledge enters the method. It is also a necessary step in order to tame the complexity of the general set of temporally extended features. In our case it is the assumption of partial observability and, more specifically, delayed causalities which led us to our definition of N^+ via basis indicator features. However, our learning algorithm Pulse (described below) will equally well work with any other definition of N^+. Carefully choosing a definition for N^+ is thus the most important means to tailor Pulse to a specific domain.

Algorithm 1. The Pulse algorithm.

```
Input: $N^+, \mathcal{O}, D$
Output: $\mathcal{F}, \Theta$
1: Initialize: $\mathcal{F} \leftarrow \emptyset$, $\Theta \leftarrow \emptyset$
2: repeat
3:     GROW $\mathcal{F}, \Theta, N^+$
4:     $\Theta \leftarrow \mathrm{argmin}_\Theta \; \mathcal{O}(\mathcal{F}, \Theta, D)$
5:     SHRINK $\mathcal{F}, \Theta$.
6: until $\mathcal{O}$ did not change
7: return $\mathcal{F}, \Theta$
8: function GROW $\mathcal{F}, \Theta, N^+$
9:     Initialize: $\mathcal{F}^+ \leftarrow N^+(\mathcal{F})$
10:    for all $f \in \mathcal{F}^+$ do
11:        if $f \notin \mathcal{F}$ then
12:            $\Theta_f \leftarrow 0$
13:        end if
14: end for
15:    $\mathcal{F} \leftarrow \mathcal{F} \cup \mathcal{F}^+$
16: end function
17: function SHRINK $\mathcal{F}, \Theta$
18:    for all $f \in \mathcal{F}$ do
19:        if $\Theta_f$ is 0 then
20:            $\mathcal{F} \leftarrow \mathcal{F} \setminus f$
21:        end if
22:    end for
23: end function
```

3.2. The Pulse algorithm

The idea of Pulse is to exploit the structure that N^+ defines on the feature set $\mathcal{T}_{N^+}$ to explore the local vicinity of the current feature set $\mathcal{F}$. Intuitively, in each iteration we add all candidates $N^+(\mathcal{F})$ to $\mathcal{F}$, then optimize the L_1-regularized objective, and remove any zero-weight features. This leads to a pulsating dynamic of the feature set, guided by N^+ and the objective, driving $\mathcal{F}$ towards an optimum. The Pulse algorithm is detailed in Algorithm 1. The main loop consists of

- growing the feature set using N^+ and assigning zero weight to any new features (line 3)
- optimizing the objective function $\mathcal{O}$ with respect to the feature weights Θ given the current feature set $\mathcal{F}$ and data D (line 4), and
- shrinking $\mathcal{F}$ by eliminating any zero-weight features (line 5).

The crucial step is the optimization of the feature weights (line 4) that effectively determines which features are kept and which ones are removed. For a good performance of Pulse the objective function $\mathcal{O}$ should fulfill two properties

1. to retain convergence guarantees (details in Section 3.3) features with zero weight should not affect the objective value
2. to keep the overall feature set sparse $\mathrm{argmin}_\Theta \mathcal{O}(\mathcal{F}, \Theta, D)$ should be sparse, too, that is, after optimizing $\mathcal{O}$ many Θ_f should be zero.

$\mathcal{O}$ can otherwise be chosen freely. Property 1 also implies that zero-weight features may be treated separately and the objective function may in many cases be computed more efficiently.

3.2.1. Optional use of scoring functions

First adding all candidate features and then removing some of them might seem wasteful compared to greedily adding single candidate features by using a scoring function, as classical feature expansion methods do. Note, however, that these "bad" candidate features are added with zero weight and are then removed because their weight was not changed in the optimization step (line 4 in Algorithm 1). That is, their weight was already at the optimum initially. The number of iterations in the optimization step, when using e.g. gradient descent methods, is thus solely determined by the non-zero-weight features we are actually interested in.

Nevertheless, if we wish to further reduce the dimensionality and speed up the optimization it is still possible to additionally use a scoring function to limit the growth of the feature set before optimizing the weights. It is important to note that a scoring function, when used with our method, takes a completely different role compared to classical feature expansion methods. Its purpose now is solely computational speed-up via pre-selecting candidate features whereas the final say on inclusion of candidates lies with the objective function. In contrast to the N^+ operation a scoring function has the advantage of taking the data into account but it may also be inconsistent with the objective function (as already mentioned in Section 2.2). On the other hand, these inconsistencies are not as grave with our method because of the different role of the scoring function, which may now be chosen rather permissive without impairing the final result of the selection process. We had positive experience with using the gradient of the unregularized objective as score in conjunction with a threshold corresponding to the L_1-regularization coefficient. This has the effect that candidate features that would in a first gradient descent step not become non-zero are not added in the first place. However, the amount of speed-up depends a lot on the specific objective and optimization method. For our experiments in Section 5 we did not use a scoring function.

3.3. Convergence guarantees

Optimality of the feature weights (line 4 in Algorithm 1) entirely depends on the objective and optimization method and is thus outside the realm of the PULSE algorithm itself. We will therefore assume that the globally optimal feature weights can always be found (which is for instance the case in our experiments) and focus on convergence of the features set.

PULSE is guaranteed to converge to a locally optimal feature set provided the objective fulfills condition 1 from above (zero-weight features must not change the objective value). In that case the objective value is retained across iterations despite adding and removing features so that the optimization step greedily improves the objective. A globally optimal feature set can be guaranteed under certain conditions, too. To see this, we first define the transition graph of PULSE:

Definition 1. The *transition graph* $G(N^+, \mathcal{O}, D)$ of PULSE for a specific expansion operation N^+, objective function $\mathcal{O}$, and data D is defined as

$$G(N^+, \mathcal{O}, D) := \left(\mathcal{P}(\mathcal{T}_{N^+}),\ A_{\text{grow}} \cup A_{\text{shrink}}\right) \tag{6}$$

with

$$A_{\text{grow}} := \left\{ \left(\mathcal{F}, \mathcal{F} \cup N^+(\mathcal{F})\right) \mid \mathcal{F} \subseteq \mathcal{T}_{N^+} \right\} \tag{7}$$

$$A_{\text{shrink}} := \left\{ (\mathcal{F} \cup \mathcal{F}_0, \mathcal{F}) \mid f \in \mathcal{F}_0 \Leftrightarrow \Theta^*_f = 0 \right\} \tag{8}$$

$$\Theta^* = \operatorname{argmin}_\Theta \mathcal{O}(\mathcal{F} \cup \mathcal{F}_0, \Theta, D) \tag{9}$$

where $\mathcal{P}(\mathcal{T}_{N^+})$ is the power set of $\mathcal{T}_{N^+}$.

That is, the vertices of G are subsets of $\mathcal{T}_{N^+}$ while A_{grow} and A_{shrink} describe the *grow* and *shrink* step of PULSE as arcs in this transition graph. There are two necessary and sufficient conditions for PULSE to converge globally:

1. there exists an alternating route of *grow*- and *shrink*-arcs in G from the initial feature set to the globally optimal feature set
2. the objective function improves strictly monotonically along this route so that PULSE does not stop prematurely.

Note that such a route is unique since a vertex in G may have any number of incoming arcs but has exactly two outgoing arcs, one in A_{grow} and one in A_{shrink}.

We can prove global convergence for a specific class of problems that is highly relevant for delayed causalities. This is the problem of predicting a future event y that depends on a number of past events $x_1, \ldots, x_n$. To expand on our example from above, consider that whether you will be able to open a closed door may depend on preconditions such as "putting the key in the lock *and* turning it *and* pushing the handle *and* pulling".

Definition 2. A binary random variable $y \in \{0,1\}$ is a *conditional event* with independent *preconditions* $x_1, \ldots, x_n \in \{0,1\}$ if and only if

$$p(y=1 \mid x_1, \ldots, x_n) = \begin{cases} p_y & \text{if } x_1 = 1 \wedge \ldots \wedge x_n = 1 \\ \overline{p}_y & \text{else} \end{cases} \tag{10}$$

$$p_y \neq \overline{p}_y \tag{11}$$

$$p(y, x_1, \ldots, x_n) = p(y \mid x_1, \ldots, x_n) p(x_1) \cdots p(x_n) \tag{12}$$

$$\forall_{x_i} 0 < p(x_i) < 1. \tag{13}$$

In the special case where $p_y = 1$ and $\overline{p}_y = 0$ the preconditions are each strictly necessary and jointly sufficient for y to occur.

Definition 3. For a conditional event y with preconditions $x_1, \ldots, x_n$ the *optimal k-predictor* using preconditions $x_1, \ldots, x_k$ is

$$\pi_k(y, x_1, \ldots, x_n) := \sum_{x_1, \ldots, x_k} p(y \mid x_1, \ldots, x_n) p(x_1) \cdots p(x_k). \tag{14}$$

Definition 4. The *quality* of a predictor π for a conditional event y with preconditions $x_1, \ldots, x_n$ is

$$q(\pi) := \mathbb{E}[\pi(y, x_1, \ldots, x_n)]_{p(y, x_1, \ldots, x_n)} \tag{15}$$

where $\mathbb{E}[\cdot]_p$ is the expected value for a distribution p.

Theorem 1. *For a conditional event y with preconditions $x_1, \ldots, x_n$ if*

1. *using N^+ as defined in* Eq. (4)
2. *the basis feature set $\mathcal{B}$ contains indicator features for $x_1, \ldots, x_n$ and y*
3. *optimizing the objective function $\mathcal{O}$ for a feature set $\mathcal{F}$ containing multiple k-fold conjunctions produces a predictor with a quality equal to the best corresponding optimal k-predictor and assigns a non-zero weight to that conjunction*
4. *the objective function $\mathcal{O}$ is a strictly monotone function of the predictor quality*

then PULSE *converges to the globally optimal feature set in at most $n-i$ iterations for an initial feature set containing at least one i-fold conjunction of preconditions.*

Proof (*sketch*). Using N^+ as defined in Eq. (4) – assumption 1 – means that *grow*-arcs in the transition graph G will add all possible extensions of existing conjunctions with any basis feature. That is, by assumption 2, if $\mathcal{F}$ contains some k-fold conjunction of preconditions then, as long as $k < n$, $N^+(\mathcal{F})$ will contain at least one $k+1$-fold conjunction generated by extending the existing conjunction. Furthermore, we can show that extending a k-predictor to a $k+1$-predictor by adding one precondition strictly improves the quality (see Appendix A for the proof). Therefore, by assumption 3, the quality of the learned predictor improves as long as $k<n$. Assumption 4 assures that Pulse does not terminate as long as this is the case.

Altogether, as long as $k<n$, for a feature set $\mathcal{F}_k$ containing at least one k-fold conjunction of preconditions the outgoing *grow*-arc leads to a feature set $\mathcal{F}_{k+1}$ with at least one $k+1$-fold conjunction while the shrink arc retains this conjunction. By induction the highest-order conjunction therefore grows in each iteration until reaching the globally optimal feature set with the full (unique) n-fold conjunction of preconditions. This establishes the alternating route with strictly monotonically improving objective required for global convergence of the feature set. □

If global convergence cannot be guaranteed in this manner for the specific N^+ and objective being used it is of course possible to use stochastic methods like simulated annealing [19,20] (within both N^+ as well as the optimization of the objective) that still guarantee *asymptotic* convergence to the global optimum. This, however, is a much weaker guarantee and may heavily impair the empirical performance.

3.4. Richness of the representation

The generated representation entirely depends on the definition of N^+ so that general statements cannot be made. However, the specific definition of N^+ suggested in Section 3.1.1 generates a set of temporally extended features that has a descriptive power similar to that of a tabular k-MDP representation (that is, using the last k observations as state representation).

If using a *fixed horizon* of $t_{min}=-k$ and no *gradual temporal extension* Pulse generates a feature set with the same descriptive power as a tabular k-MDP representation after k iterations. If using *gradual temporal extension*, after running Pulse for n iterations the features reach back n steps into the past and after $n=k+|\mathcal{O}|$ iterations they have a descriptive power at least as high as a k-MDP representation, where $|\mathcal{O}|$ is the number of possible observations.[3] If additionally using a *fixed horizon* it is strictly equal to a k-MDP representation otherwise it is higher because we have additional features (with lower-order conjunctions) reaching even farther into the past. All this assumes that the regularization part of the objective does not eliminate the corresponding features.

In general, Pulse can be used with any N^+ that produces finite sets of candidate features $N^+(\mathcal{F})$. This provides a powerful framework that can be extended in various ways beyond what is presented in this paper. Some examples that we think would be worth exploring are:

Continuous observations and time: For continuous observations N^+ could use the tensor product (outer product) of basis functions in different observation sub-spaces to select a sparse subset of basis functions for the whole observation space. For instance, combining plane waves in x-, y-, and z-direction would produce the Fourier basis for 3D-space. In case of continuous time, the basis features could be basis functions in the temporal dimension for each possible observation. A combination of both would be possible, too.

Towards latent state representations: If features are not computed from the current history but instead updated in every time step we move from a history-based approach towards learning a latent space representation. Finite state machines and their continuous generalization, multiplicity automata [21], provide suitable building blocks that could be recombined by N^+. Another option would be structure learning in deep recurrent neural networks (RNNs) by using smaller RNNs as building blocks. Which choice of N^+ could generate, for instance, a set of temporally extended features equivalent to k-order predictive state representations [2] is an interesting non-trivial question.

[3] We use the term “observation” here in a general sense in that it may include the action and reward, too.

3.5. Comparison to decision trees

A decision tree can be converted into a corresponding set of binary temporally extended features, as illustrated in Fig. 1 for the case of binary basis features. For a k-valued basis feature in the decision tree the temporally extended features will instead use k binary indicator features, one for each possible outcome, within their basis features. After the conversion, there is a one-to-one correspondence between leaf nodes and temporally extended features such that at any time exactly one temporally extended feature is active (`true`), which corresponds exactly to the classification of the decision tree. That is, a decision tree with n leaf nodes becomes a set of the same size.

In contrast, converting a set of temporally extended features into a decision tree is only possible if the temporally extended features are discrete but not for real-valued features. For such a conversion, the temporally extended features *themselves* (not their basis features) need to be used in the decision tree so that each possible assignment of feature values is represented by a separate leaf node. This means that a set of n binary features becomes an exponentially larger decision tree with 2^n leaf nodes.

In that sense, temporally extended features have a higher descriptive power than decision trees. This is also the reason why

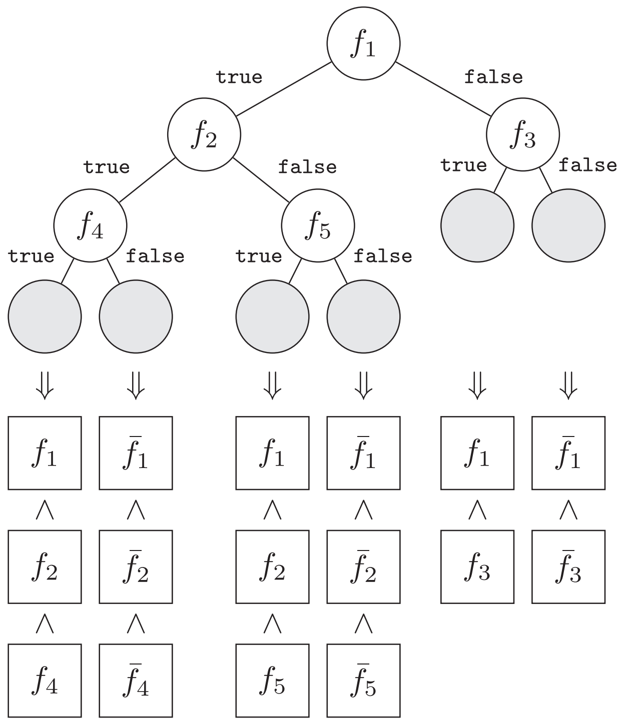
f_1
true
false
f_2
f_3
true
false
true
false
f_4
f_5
true
false
true
false
f_1 $\bar{f}_1$ f_1 $\bar{f}_1$ f_1 $\bar{f}_1$
f_2 $\bar{f}_2$ f_2 $\bar{f}_2$ f_3 $\bar{f}_3$
f_4 $\bar{f}_4$ f_5 $\bar{f}_5$

Fig. 1. Transforming a binary decision tree into a set of temporally extended features: for each leaf node we build one temporally extended feature. For all nodes on the path from the root node to the corresponding leaf node we take the conjunction of the corresponding basis features f_i or their negation $\bar{f}_i$, respectively.

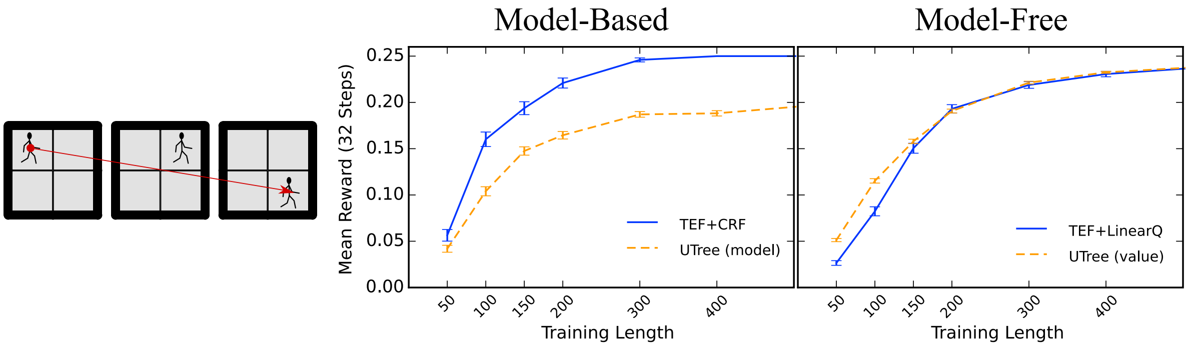

Fig. 2. 2×2 Maze (1600 k-MDP states, 12 latent states): on the left the maze was "unrolled" for a successful *activation* and *collection* of the single reward, which has a delay of $\Delta t = 2$ and is depicted by the red arrow. The two plots on the right show the mean reward with standard error of the mean for the tested methods (see text for details and the web version of this paper for interpretation of the references to color).

the conversions between decision trees and sets of temporally extended features are not inverse to each other – transforming a decision tree into a set of temporally extended features and simply back again results in a blown up version of the decision tree.

4. Reinforcement learning with Pulse

In the preceding section we described the workings of our Pulse algorithm in general. In this section we describe more concretely how Pulse can be used in a reinforcement learning agent. We already gave a definition for N^+ in Section 3.1.1 that is tailored to the kind of problem we are dealing with – delayed causalities. The second important ingredient is the objective function, which depends on the kind of agent we aim to train. We will present the two agents that we used in our experiments, one model-based and one model-free.

4.1. Model-based agent (TEF+CRF)

For our model-based agent we use temporally extended features in conjunction with a conditional random field (CRF) [22,23] to learn a predictive model of the environment. We refer to this agent as *TEF+CRF*. The CRF models the conditional probability for the next observation and reward given the current history and action. It is a log-linear model of the form

$$p(\overline{o},\overline{r}\,|\,\overline{h},\overline{a}) = \frac{1}{Z(h)} \exp \sum_{f \in \mathcal{F}} \theta_f f(h) \tag{16}$$

$$\text{with} \quad Z(h) = \sum_{\overline{o},\overline{r}} \exp \sum_{f \in \mathcal{F}} \theta_f f(h) \tag{17}$$

where $(\overline{a},\overline{o},\overline{r})$ is the last action-observation-reward triplet in history h and $\overline{h}$ is the remaining part of h. We optimize the weights Θ to maximize the likelihood of data $D = \{(a_1,o_1,r_1),\dots,(a_n,o_n,r_n)\}$ or equivalently minimize the neg-log-likelihood

$$\ell(\Theta) = -\log \prod_{i=1}^{n} p(o_i, r_i\,|\,h_{i-1}, a_i) \tag{18}$$

$$\ell(\Theta) = -\sum_{i=1}^{n} \left[\sum_{f \in \mathcal{F}} \theta_f f(h_i) - \log Z(h_i) \right] \tag{19}$$

via gradient descent using L-BFGS [24,25]. That is, ℓ is the objective used by Pulse. To enforce sparseness we additionally use a L_1-regularization of variable strength ρ

$$\ell(\Theta,\rho) = \ell(\Theta) + \rho \sum_{f \in \mathcal{F}} |\theta_f|. \tag{20}$$

This objective fulfills all requirements needed to guarantee global convergence to an optimal feature set (cf. Section 3.3).

4.2. Model-free agent (TEF+Linear Q)

For our model-free agent we learn a linear approximation of the action-value function (Q-function) and refer to this agent as *TEF+Linear Q*. The Q-function is computed as

$$Q(\overline{a},\overline{h}) = \sum_{f \in \mathcal{F}} \theta_f f(h) \tag{21}$$

where $\overline{a}, \overline{h}$, and h are defined as in Eq. (16). TEF+Linear Q is trained via least-squares policy iteration [26]. Again a L_1-regularization of variable strength is used. We have not proved global convergence of the feature set for this objective but local convergence is guaranteed nonetheless.

5. Experiments

We performed evaluations in four different deterministic partially observable maze environments that we believe exhibit a prototypical structure for temporally delayed causalities. The 2×2 and 4×4-maze (Figs. 2 and 3) contain delayed rewards that are "activated" at one location and later "collected" at different one with a fixed temporal delay Δt. For both mazes we used the maximum Δt as a fixed time horizon in all compared methods. The 4×4-maze additionally contains doors that are opened by taking a step into the wall at the location of the switch and remain open for 2 more time steps. The small and large *Cheese Maze* (Figs. 4 and 5), where the agent only perceives adjacent walls, were introduced in [5] and adapted from [27], respectively. While these mazes cannot be formulated as a k-MDP the time horizon of $t_{min} = -2$, which we used for all compared methods, is still sufficient for performing optimally. In the Cheese Mazes the agent receives a reward of -1 for bumping into a wall, a reward of $+1$ for reaching the goal location (indicated by the cheese) and a reward of -0.1 otherwise. Upon reaching the goal location the agent is immediately relocated to a random location (small Cheese Maze) or the start location (large Cheese Maze, indicated by the mouse).

For each maze we performed a number of trials with a training phase of varying length (using random policy) and an evaluation phase of fixed length (using the agent's optimal policy). The plots on the right of Figs. 2–5 show the mean reward during evaluation with the standard error of the mean as error bars. The number of trials for each data point varies depending on environment, method, and training length and is not explicitly indicated to keep the plots clear. For Planning with the model-based methods we

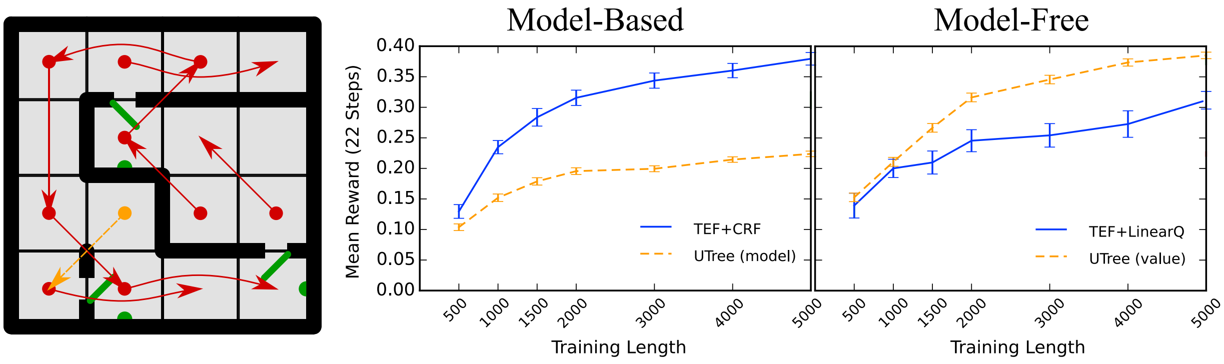

Fig. 3. 4×4 Maze (4 096 000 k-MDP states; 34 012 224 latent states): on the left the 4×4 maze is shown in compact form. Red solid arrows depict rewards with a delay of $\Delta t=2$, the orange dashed arrow depicts a reward with delay $\Delta t=3$. Doors are depicted in green with their switch being the nearby semicircle. The two plots on the right show the mean reward with standard error of the mean for the tested methods (see text for details and the web version of this paper for interpretation of the references to color).

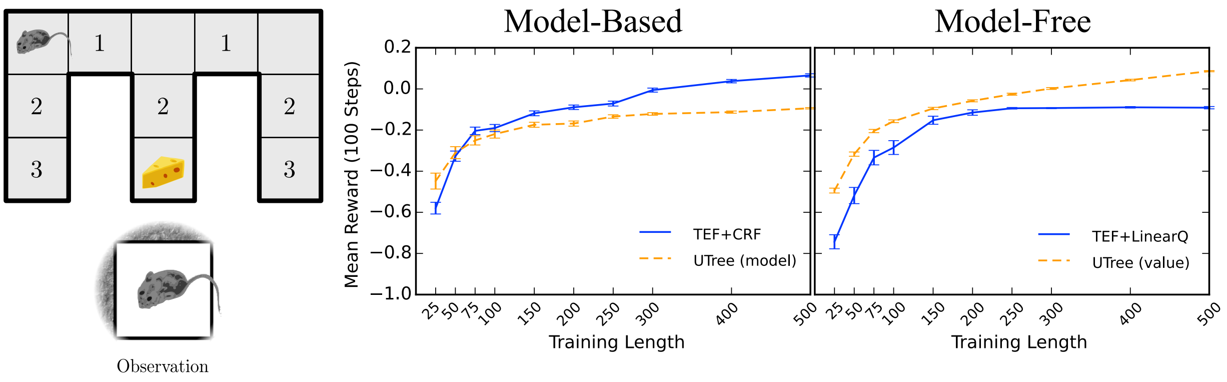

Fig. 4. Small Cheese Maze (∞ k–MDP states; 11 latent states): on the top left the maze is shown and locations that are indistinguishable to the agent are numbered accordingly. On the bottom left the current observation (representing only the existence or non-existence of the adjacent walls) is illustrated. The two plots on the right show the mean reward with standard error of the mean for the tested methods (see text for details).

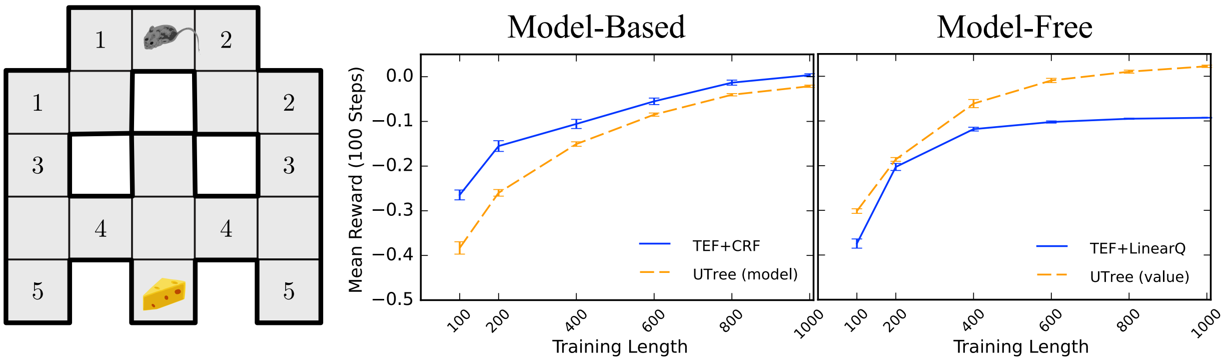

Fig. 5. Large Cheese Maze (∞ k–MDP states; 18 latent states): This is a larger version of the maze described in Fig. 4. Its structure is depicted on the left, again with aliased locations numbered accordingly. The two plots on the right show the mean reward with standard error of the mean for the tested methods (see text for details).

used a variant of Monte-Carlo tree search [28] that exploits the probabilities given by the model together with the known bounds for the reward.

5.1. Results

First, note that model-based and model-free methods cannot be compared on the same basis since they solve different tasks and have different advantages and disadvantages [29,30]. For this reason, we plot them on the same scale but next to each other. Generally, learning a complete predictive model of the environment is a more complex task compared to only learning the action values, which is also reflected by the fact that U-Tree (model) and U-Tree (value) display a large difference in performance although the underlying method is identical.

Having said that, notice that our model-based agent TEF+CRF not only outperforms U-Tree (model) by a large margin in all four environments but even performs better than U-Tree (value) in the 2×2-maze an the initial learning phase of the large Cheese Maze. Our model-free method TEF+Linear Q, on the other hand, generally falls behind U-Tree (value) and has a performance only similar to that of U-Tree (model) or even worse (in case of the large Cheese Maze).

The reason behind the strong performance of TEF+CRF, we think, is that the CRF allows an efficient decomposition of the probability distributions, especially in case of deterministic

Table 1
Feature set learned by TEF+CRF in the 2 × 2-maze based on 1500 training steps (37 features): each row contains a feature (mostly pairwise conjunctions of basis features) as well as the feature weight θ in the conditional random field (cf. (16)). An icon represents a basis feature while the subscript indicates the specific time index. For example, the basis feature ▘$_{-1}$ indicates whether the agent was at the top left location in the last time step; $\surd_0$ indicates whether it is (hypothetically) going to get a reward in the next step; and $\leftarrow_{-1}$ indicates that the last move was to the left. Generally, large positive weights result in high probabilities when that feature is active and large negative weights in low probabilities. However, multiple active features sum up and features that are active for the other outcomes influence the probabilities via the normalization. We partitioned the table into semantic blocks and sub-blocks.

Reward probability	
$\surd_0$	$\theta = -11.91$
$\times_0$	$\theta = 11.91$

Correlation between rewards and locations	
$\surd_0 \wedge$ ▗$_0$	$\theta = 13.50$
$\times_0 \wedge$ ▘$_{-1}$	$\theta = 8.22$
$\times_0 \wedge$ ▗$_{-1}$	$\theta = 6.59$
$\surd_0 \wedge$ ▘$_{-2}$	$\theta = 7.71$
$\times_0 \wedge$ ▘$_{-2}$	$\theta = -7.71$

What location can be reached from what other location	
▘$_0 \wedge$ ▘$_{-1}$	$\theta = 12.95$
▘$_0 \wedge$ ▖$_{-1}$	$\theta = 7.15$
▘$_0 \wedge$ ▝$_{-1}$	$\theta = 7.03$
▝$_0 \wedge$ ▝$_{-1}$	$\theta = 16.02$
▝$_0 \wedge$ ▗$_{-1}$	$\theta = 8.18$
▝$_0 \wedge$ ▘$_{-1}$	$\theta = 7.73$
▖$_0 \wedge$ ▖$_{-1}$	$\theta = 15.60$
▖$_0 \wedge$ ▗$_{-1}$	$\theta = 7.97$
▖$_0 \wedge$ ▘$_{-1}$	$\theta = 7.37$
▗$_0 \wedge$ ▗$_{-1}$	$\theta = 14.36$
▗$_0 \wedge$ ▖$_{-1}$	$\theta = 9.34$
▗$_0 \wedge$ ▝$_{-1}$	$\theta = 6.28$

What location can be reached with what action	
$\uparrow_0 \wedge$ ▘$_0$	$\theta = 14.84$
$\uparrow_0 \wedge$ ▝$_0$	$\theta = 15.11$
$\downarrow_0 \wedge$ ▖$_0$	$\theta = 15.43$
$\downarrow_0 \wedge$ ▗$_0$	$\theta = 15.69$
$\leftarrow_0 \wedge$ ▘$_0$	$\theta = 14.86$
$\leftarrow_0 \wedge$ ▖$_0$	$\theta = 14.78$
$\rightarrow_0 \wedge$ ▗$_0$	$\theta = 15.60$
$\rightarrow_0 \wedge$ ▝$_0$	$\theta = 14.96$

Other correlations	
▗$_0 \wedge$ ▘$_{-2}$	$\theta = -6.67$
▘$_0 \wedge$ ▗$_{-2}$	$\theta = -3.37$
▖$_0 \wedge$ ▝$_{-2}$	$\theta = -2.37$
▝$_0 \wedge$ ▖$_{-2}$	$\theta = -0.95$
▗$_0 \wedge \leftarrow_{-1}$	$\theta = -3.23$
▘$_0 \wedge \uparrow_{-1}$	$\theta = 2.78$
▘$_0 \wedge \leftarrow_{-1}$	$\theta = 2.56$
▗$_0 \wedge \rightarrow_{-1}$	$\theta = 1.31$
$\times_0 \wedge$ ▝$_{-1}$	$\theta = -1.13$
$\surd_0 \wedge$ ▝$_{-1}$	$\theta = 1.13$

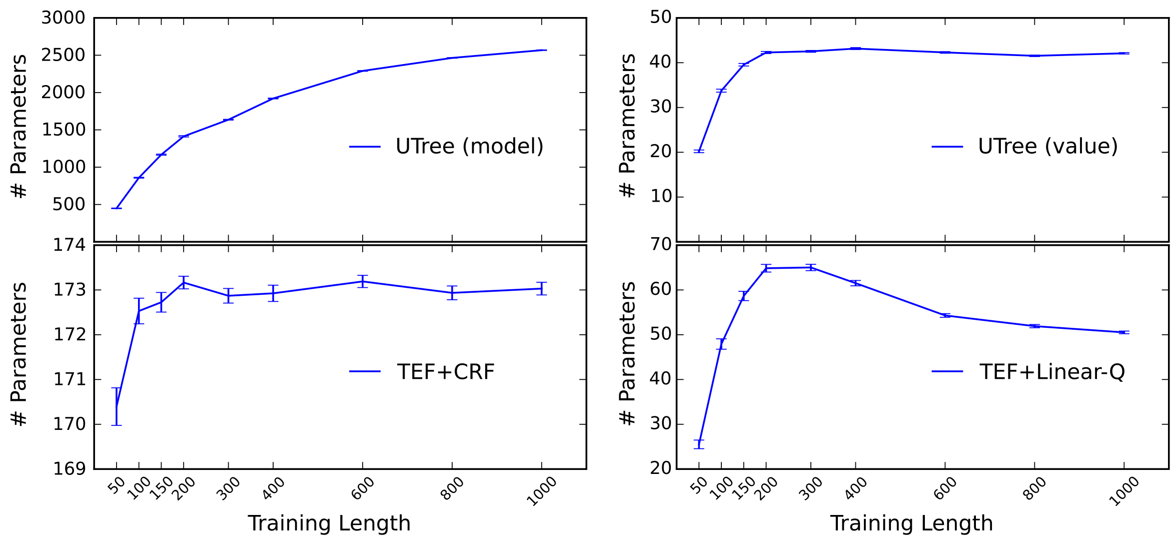

Fig. 6. Growth of feature set and decision tree, respectively, in the 2 × 2-maze for all methods: The growth is measured by the number of learned parameters, that is, the number of features for TEF+CRF and TEF+Linear Q and the number of leaf nodes times 5 and times 40 for U-Tree (value) and U-Tree (model), respectively. Plots are rescaled since we are interested in the characteristics of the curves rather than the absolute values.

environments: all probabilities are close to zero or one and can therefore be described as the product of a few feature conjunctions (also confer Table 1).

Conversely, we think TEF+Linear Q performs poorly because the action values are diverse for the different locations and cannot be described as the sum of a few feature conjunctions. TEF+Linear Q thus requires a relatively large feature set that is harder to train given limited data. Also note that in both the small and the large Cheese Maze TEF+Linear Q does not significantly surpass the -0.1 margin, which corresponds to a policy that avoids bumping into the walls but does not reach the goal location. This indicates that the poor performance of TEF+Linear Q might partly be due to a problem with the policy-update step during training via policy iteration.

5.1.1. *Characteristics of the discovered features*

We will now compare in more detail the features discovered with Pulse to the decision tree learned by U-Tree. First note that we cannot directly compare the number of features to the number of leaf nodes because, for instance, U-Tree (model) in the 2 × 2-maze learns 40 parameters per leaf node (one probability for each action-observation-reward triplet). Rather we are interested in characteristic differences in the growth of the feature set and decision tree, respectively, over the training length. This is shown in Fig. 6 for all methods in the 2 × 2-maze. There are two main observations:

1. For the two best methods (TEF+CRF and U-Tree (value)) that reach near-optimal performance the growth flattens out earlier than their performance (cf. Fig. 2), which suggest that they

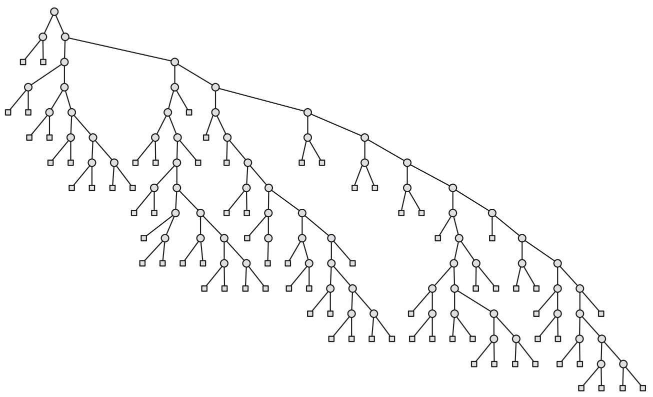

Fig. 7. U-Tree (model) learned from 1500 training steps in the 2×2-maze (74 leaf nodes; depth 2–15): each internal node corresponds to one basis feature and each of the leaf nodes contains 40 parameters (the transition probabilities). Both were omitted for the sake of clarity.

first discover the optimal feature set and decision tree, respectively, and then keep on improving by learning better parameters.

2. The curves for the other two methods (TEF+Linear Q and U-Tree (model)) are opposed to each other. While U-Tree (model) grows an increasingly large decision tree, TEF+Linear Q as the only method has a *negative* growth from some point on, which suggests that the richer data set allows TEF+Linear Q to more efficiently select features while improving performance. Although TEF+Linear Q does not overall display an outstanding performance this still proves that being able to shrink the feature set – a major strength of our method compared to pure feature expansion – is actually relevant in practice.

Another goal we had in mind when designing our method was to make the temporal structure of the environment explicit and accessible. In Table 1 we list the complete feature set (including all weights) learned by TEF+CRF in the 2×2-maze; in Fig. 7 we sketched the corresponding decision tree learned by U-Tree (model). Keeping in mind that in Fig. 7 we not only left out the concrete features for the internal nodes but also each leaf node actually contains 40 parameters, we think that the feature set learned with PULSE is not only much more compact but also more intuitive to interpret (see caption texts for details).

6. Conclusion

We considered the problem of uncovering temporally delayed causalities in partially observable reinforcement learning domains. To this end we introduced *temporally extended features* along with a training method called PULSE that efficiently and incrementally discovers a sparse set of relevant temporally extended features. We provided convergence guarantees and evaluated our approach empirically showing that in terms of achieved rewards as well as the number of required features PULSE clearly outperforms its competitors in a model-based setting. While in this paper we considered very simple basis features, we discussed how the general framework provided by PULSE can be extended to learn much richer representations.

Acknowledgments

Robert Lieck is financially supported by the German National Academics Foundation.

Appendix A. Proof

Theorem 2. *For a conditional event y with preconditions $x_1, \ldots, x_n$ extending an optimal k-predictor π_k to an optimal $k+1$-predictor π_{k+1} by adding one more precondition strictly improves the quality*

$$q(\pi_{k+1}) > q(\pi_k) \tag{A.1}$$

Proof. We will first show that an optimal $n-1$-predictor has lower quality than an optimal n-predictor. We abbreviate notation by writing p_i for $p(x_i = 1)$. The quality of π_{n-1} using all preconditions except (w.l.o.g.) x_1 is (cf. Eqs. (14) and (15))

$$q(\pi_{n-1}) = \mathbb{E}[p(y|x_2, \ldots, x_n)] \tag{A.2}$$

$$q(\pi_{n-1}) = \sum_{y,x_1,\ldots,x_n} p(y, x_1, \ldots, x_n) p(y\,|\,x_2, \ldots, x_n) \tag{A.3}$$

$$q(\pi_{n-1}) = \sum_{y,x_1,\ldots,x_n} p(x_1)\ldots p(x_n) p(y\,|\,x_1, \ldots, x_n) \left[\sum_{x_1'} p\left(y\,|\,x_1', \ldots, x_n\right) p\left(x_1'\right)\right] \tag{A.4}$$

$$q(\pi_{n-1}) = \sum_{x_2,\ldots,x_n} p(x_2)\ldots p(x_n) \sum_y \left[\sum_{x_1} p(y|x_1, \ldots, x_n) p(x_1)\right]^2 \tag{A.5}$$

$$q(\pi_{n-1}) = p_2 \ldots p_n \underbrace{\sum_y \left[\sum_{x_1} p(y|x_1, \ldots, x_n) p(x_1)\right]^2}_{\alpha(x_2,\ldots,x_n)} + \cdots$$
$$+ \sum_{\substack{x_2,\ldots,x_n \\ x_2 = 0 \vee \ldots \vee x_n = 0}} p(x_2)\ldots p(x_n) \underbrace{\sum_y \left[\sum_{x_1} p(y|x_1, \ldots, x_n) p(x_1)\right]^2}_{\beta(x_2,\ldots,x_n)}, \tag{A.6}$$

where in the last step we took the term with preconditions x_2 to x_n being all fulfilled out of the sum. The quality of the optimal n-predictor using all preconditions is (following the same structure)

$$q(\pi_n) = \mathbb{E}[p(y|x_1, \ldots, x_n)] \tag{A.7}$$

$$q(\pi_n) = \sum_{y,x_1,\ldots,x_n} p(y, x_1, \ldots, x_n)\, p(y|x_1, \ldots, x_n) \tag{A.8}$$

$$q(\pi_n) = \sum_{y,x_1,\ldots,x_n} p(x_1)\ldots p(x_n) p(y\,|\,x_1, \ldots, x_n)^2 \tag{A.9}$$

$$q(\pi_n) = \sum_{x_2,\ldots,x_n} p(x_2)\ldots p(x_n) \sum_y \left[\sum_{x_1} p(y|x_1, \ldots, x_n)^2 p(x_1)\right] \tag{A.10}$$

$$q(\pi_n) = p_2 \ldots p_n \underbrace{\sum_y \left[\sum_{x_1} p(y|x_1, \ldots, x_n)^2 p(x_1)\right]}_{\alpha'(x_2,\ldots,x_n)} + \cdots$$
$$+ \sum_{\substack{x_2,\ldots,x_n \\ x_2 = 0 \vee \ldots \vee x_n = 0}} p(x_2)\ldots p(x_n) \underbrace{\sum_y \left[\sum_{x_1} p(y|x_1, \ldots, x_n)^2 p(x_1)\right]}_{\beta'(x_2,\ldots,x_n)}. \tag{A.11}$$

β and β' in Eqs. (A.6) and (A.11) are constant since

$$x_2 = 0 \vee \ldots \vee x_n = 0 \tag{A.12}$$

for all terms of the sum, so that

$$\beta = \sum_y \left[\sum_{x_1} p(y|x_1, \ldots, x_n) p(x_1)\right]^2 \Bigg|_{x_2 = 0 \vee \ldots \vee x_n = 0} \tag{A.13}$$

$$\beta = \left[(1-\overline{p}_y)(1-p_1) + (1-\overline{p}_y)p_1\right]^2 + \left[\overline{p}_y(1-p_1) + \overline{p}_y p_1\right]^2 \tag{A.14}$$

$$\beta = \overline{p}_y^2 + (1-\overline{p}_y)^2 \tag{A.15}$$

and

$$\beta' = \sum_y \left[\sum_{x_1} p(y|x_1, \ldots, x_n)^2 p(x_1) \right] \Bigg|_{x_2 = 0 \vee \ldots \vee x_n = 0} \tag{A.16}$$

$$\beta' = (1-\overline{p}_y)^2 (1-p_1) + (1-\overline{p}_y)^2 p_1 + \overline{p}_y^2 (1-p_1) + \overline{p}_y^2 p_1 \tag{A.17}$$

$$\beta' = \overline{p}_y^2 + (1-\overline{p}_y)^2 \tag{A.18}$$

$$\beta' = \beta. \tag{A.19}$$

Likewise α and α' are constant since

$$x_2 = 1 \wedge \ldots \wedge x_n = 1 \tag{A.20}$$

so that

$$\alpha = \sum_y \left[\sum_{x_1} p(y|x_1, \ldots, x_n) p(x_1) \right]^2 \Bigg|_{x_2 = 1 \wedge \ldots \wedge x_n = 1} \tag{A.21}$$

$$\alpha = \left[(1-\overline{p}_y)(1-p_1) + (1-p_y) p_1 \right]^2 + \left[\overline{p}_y (1-p_1) + p_y p_1 \right]^2 \tag{A.22}$$

$$\alpha' = \sum_y \left[\sum_{x_1} p(y|x_1, \ldots, x_n)^2 p(x_1) \right] \Bigg|_{x_2 = 1 \wedge \ldots \wedge x_n = 1} \tag{A.23}$$

$$\alpha' = (1-\overline{p}_y)^2 (1-p_1) + (1-p_y)^2 p_1 + \overline{p}_y^2 (1-p_1) + p_y^2 p_1. \tag{A.24}$$

Using this, we show

$$q(\pi_n) > q(\pi_{n-1}) \tag{A.25}$$

$$0 > \mathbb{E}[p(y|x_2, \ldots, x_n)] - \mathbb{E}[p(y|x_1, \ldots, x_n)] \tag{A.26}$$

$$\Leftrightarrow 0 > \alpha - \alpha' \tag{A.27}$$

$$\Leftrightarrow 0 > 2p_1 (p_1\, p_y^2 - 2p_1 p_y\, \overline{p}_y + p_1 \overline{p}_y^2 - p_y^2 + 2p_y \overline{p}_y - \overline{p}_y^2) \quad |p_1 > 0 \tag{A.28}$$

$$\Leftrightarrow 0 > p_1 (p_y^2 - 2p_y \overline{p}_y + \overline{p}_y^2) - (p_y^2 + 2p_y \overline{p}_y - \overline{p}_y^2) \tag{A.29}$$

$$\Leftrightarrow 0 > (p_1 - 1)\,(p_y - \overline{p}_y)^2 \quad |p_y \neq \overline{p}_y \tag{A.30}$$

$$\Leftrightarrow 1 > p_1 \tag{A.31}$$

where we used assumptions (Eqs. (11) and (13)).

Now we show that any optimal $n-1$-predictor for a conditional event y with preconditions $x_1, \ldots, x_n$ also is an optimal $n'-$predictor for a different conditional event y' with preconditions $x'_1, \ldots, x'_{n'}$ with $n' = n-1$. This can be trivially shown by defining

$$y' = y \tag{A.32}$$

$$\forall_{i \in \{1, \ldots, n-1\}} x'_i = x_i \tag{A.33}$$

$$p(y', x'_1, \ldots, x'_{n'}) = \sum_{x_n} p(y, x_1, \ldots, x_n). \tag{A.34}$$

By induction follows that an optimal k-predictor π_k has a higher quality than an optimal l-predictor π_l if the preconditions used by π_k are an extension of the preconditions used by π_l. Thus extending an optimal k-predictor π_k to an optimal $k+1$-predictor π_{k+1} by adding one more precondition strictly improves the quality. □

References

[1] W.S. Lovejoy, A survey of algorithmic methods for partially observed Markov decision processes, Ann. Oper. Res. 28 (1) (1991) 47–65.

[2] M.L. Littman, R.S. Sutton, S. Singh, Predictive Representations of State, In: Advances in Neural Information Processing Systems, vol. 14, MIT Press, Cambridge, 2002, pp. 1555–1561.

[3] E.A. Hansen, An improved policy iteration algorithm for partially observable mdps, Adv. Neural Inf. Process. Syst. (1998) 1015–1021.

[4] F.M. Willems, Y.M. Shtarkov, T.J. Tjalkens, The context-tree weighting method: basic properties, IEEE Trans. Inf. Theory 41 (3) (1995) 653–664.

[5] A.K. McCallum, Reinforcement learning with selective perception and hidden state (Ph.D. thesis), Computer Science Department, University of Rochester, 1996.

[6] J. Veness, K.S. Ng, M. Hutter, D. Silver, Reinforcement learning via aixi approximation, in: AAAI, 2010.

[7] P. Nguyen, P. Sunehag, M. Hutter, Context tree maximizing reinforcement learning, In: Proceedings of the 26th AAAI Conference on Artificial Intelligence, 2012, pp. 1075–1082.

[8] B. Bakker, Reinforcement learning with long short-term memory, In: NIPS, 2001, pp. 1475–1482.

[9] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning), The MIT Press, Cambridge, 1998,, URL ⟨http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html⟩.

[10] S. Della Pietra, V. Della Pietra, J. Lafferty, Inducing features of random fields, IEEE Trans. Pattern Anal. Mach. Intell. 19 (4) (1997) 380–393, http://dx.doi.org/10.1109/34.588021.

[11] A. McCallum, Efficiently inducing features of conditional random fields, In: Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, 2002, pp. 403–410.

[12] A. Geramifard, F. Doshi, J. Redding, N. Roy, J. How, Online discovery of feature dependencies, In: L. Getoor, T. Scheffer (Eds.), Proceedings of the 28th International Conference on Machine Learning (ICML-11), ICML '11, ACM, New York, NY, USA, 2011, pp. 881–888.

[13] N.K. Ure, A. Geramifard, G. Chowdhary, J.P. How, Adaptive planning for Markov decision processes with uncertain transition models via incremental feature dependency discovery, In: Machine Learning and Knowledge Discovery in Databases, Springer, Heidelberg, 2012, pp. 99–115.

[14] J.Z. Kolter, A.Y. Ng, Regularization and feature selection in least-squares temporal difference learning, In: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, New York, 2009, pp. 521–528.

[15] M. Ghavamzadeh, A. Lazaric, M. Hoffman, R. Munos, Finite-sample analysis of lasso-td, In: Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 1177–1184.

[16] C. Painter-Wakefield, R. Parr, L1 Regularized Linear Temporal Difference Learning, Technical Report, Duke CS Technical Report TR-2012-01, 2012.

[17] M.W. Hoffman, A. Lazaric, M. Ghavamzadeh, R. Munos, Regularized least squares temporal difference learning with nested ℓ_2 and ℓ_1 penalization, In: Recent Advances in Reinforcement Learning, Springer, Heidelberg, 2012, pp. 102–114.

[18] M. Daswani, P. Sunehag, M. Hutter, Q-learning for history-based reinforcement learning, In: Asian Conference on Machine Learning, 2013, pp. 213–228.

[19] S. Kirkpatrick, Optimization by simulated annealing: quantitative studies, J. Stat. Phys. 34 (5–6) (1984) 975–986.

[20] A. Dekkers, E. Aarts, Global optimization and simulated annealing, Math. Progr. 50 (1–3) (1991) 367–393.

[21] M. Schützenberger, On the definition of a family of automata, Inf. Control 4 (2–3) (1961) 245–270, http://dx.doi.org/10.1016/S0019-9958(61)80020-X ⟨http://www.sciencedirect.com/science/article/pii/S001999586180020X⟩.

[22] J.D. Lafferty, A. McCallum, F.C.N. Pereira, Conditional random fields: probabilistic models for segmenting and labeling sequence data, in: Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001, pp. 282–289, URL ⟨http://dl.acm.org/citation.cfm?id=645530.655813⟩.

[23] Y. Altun, I. Tsochantaridis, T. Hofmann, et al., Hidden Markov support vector machines, In: ICML, vol. 3, 2003, pp. 3–10.

[24] R.H. Byrd, J. Nocedal, R.B. Schnabel, Representations of quasi-Newton matrices and their use in limited memory methods, Math. Progr. 63 (1–3) (1994) 129–156.

[25] N. Okazaki, A library of limited-memory Broyden-Fletcher-Goldfarb-Shanno, ⟨http://www.chokkan.org/software/liblbfgs/⟩.

[26] M.G. Lagoudakis, R. Parr, Least-squares policy iteration, J. Mach. Learn. Res. 4 (2003) 1107–1149 ⟨http://dl.acm.org/citation.cfm?id=945365.964290⟩.

[27] T. Shibuya, T. Hamagami, Complex-valued reinforcement learning: a context-based approach for pomdps, In: Advances in Reinforcement Learning, INTECH Open Access Publisher, Rijeka, 2011.

[28] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, S. Colton, A survey of Monte Carlo tree search methods, IEEE Trans. Comput. Intell. AI Games 4 (1) (2012) 1–43, http://dx.doi.org/10.1109/TCIAIG.2012.2186810.

[29] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: a survey, CoRR cs.AI/9605103, URL ⟨http://arxiv.org/abs/cs.AI/9605103⟩.

[30] R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, M.L. Littman, An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning, in: Proceedings of the 25th International Conference on Machine Learning, ICML '08, ACM, New York, NY, USA, 2008, pp. 752–759, http://dx.doi.org/10.1145/1390156.1390251, ⟨http://doi.acm.org/10.1145/1390156.1390251⟩.

Robert Lieck is a PhD student in the group of Marc Toussaint at the University of Stuttgart since 2012. He studied physics and philosophy at FU Berlin and received his degree in 2010. Before starting his PhD he stayed at Zuse Institute Berlin in Stefan Zachow's group for visual data analysis. His research focuses on probabilistic modeling and representation learning as well as on decision making and planning with an emphasis on solving non-Markov learning and planning problems.

Marc Toussaint is a full professor for Machine Learning and Robotics at the University of Stuttgart since 2012. He studied Physics and Mathematics at the University of Cologne and received his PhD from Ruhr-University Bochum in 2004 before staying with the University of Edinburgh as a post-doc. In 2007 he received an assistant professorship and Emmy Noether research group leadership at TU & FU Berlin. His research focuses on the combination of decision theory and machine learning, motivated by fundamental research questions in robotics. Reoccurring themes in his research are appropriate representations (symbols, temporal abstractions, and relational representations) to enable efficient learning and manipulation in real world environments, and how to achieve jointly geometric, logic and probabilistic learning and reasoning. He is currently a coordinator of the German research priority programme *Autonomous Learning*, member of the editorial board of the Journal of AI Research (JAIR), reviewer for the German Research Foundation, and programme committee member of several top conferences in the field (UAI, R:SS, ICRA, IROS, AIStats, ICML). His work was awarded best paper at R:SS'12, ICMLA'07 and runner up at UAI'08.