Functional Similarity in Markov Environments

M. M. Hassan Mahmud and Sylvian R. Ray
Department of Computer Science
University of Illinois At Urbana Champaign
Urbana, IL 61801, USA

Abstract

In this paper we discuss the notion of functional similarity between different situations an artificial agent may encounter, and show how it may be used to transfer information across tasks. We say two situations are functionally similar (FS) if there exists an action that has a similar effect in both the situations. So for instance, many situations containing a soccer ball in an open space are FS with respect to the action "kick ball". Thus, if we can determine that a novel situation is FS to some previously observed situations with respect to an action, we can use the behavior of the previous situations as an estimate of the behavior of the new situation with respect to that action. In this paper we give a concrete definition of functional similarity for Markov Environments and briefly show how this may be used to construct and use a novel type of forward model, which we call transition prediction model (TPM), for such domains. We also mention some interesting theoretical properties of the TPM. Finally we describe possible avenues of future research and related work.

1 Introduction

In this paper we briefly look at a mechanism that artificial agents may use to infer properties of one part of an environment from another. This is particularly useful for agents, such as robots, that operate in the domains where it is expensive to acquire examples required to learn. The mechanism we look at exploits the notion of *functional similarity* (the material in this paper is described in more detail in [1]). For a qualitative example of this, consider a human player first learning to play squash and then learning to play racquetball. The games are not identical, but the courts, racquets and balls are similar. More importantly, the actions of the player, such as forehands, backhands, running towards the ball etc. has *similar effects* in both games. That is situations experienced by the player - various trajectory of the ball and tactics used by the opponent - are functionally similar with respect to her actions. Because of this similarity, the player is able to use knowledge acquired in one game in another. Indeed, functional similarity can be identified between any two tasks that share anything humans identify as objects (e.g. balls, racquets etc.)

2 Functional Similarity in Markov Environments

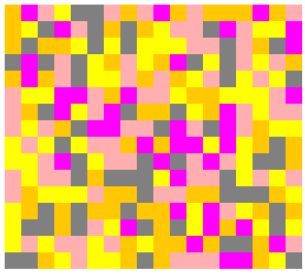
Markov Environments. We consider agents in finite Markov Environments, which are defined by the pair (S, A). S is the set of states that the agent may be in and A is the set

of actions that the agent may apply at different states, and both sets are finite. Actions are probabilistic, i.e. the next state s' observed on applying an action a at state s is given by the conditional distribution $P(s'|s,a)$. We also assume each state is described by a finite length vector of features i.e. $S = F_1 \times F_2 \times \cdots \times F_n$ where each feature F_i is finite valued.

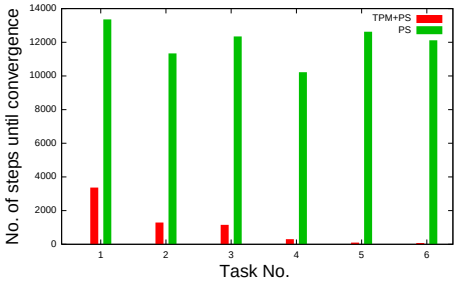
Example 2.1 To motivate the definition of functional similarity that follows, we consider the following domain that we also use in our experiments below. The domain is a 16×16 celled gridworld domain called the Sticky Room domain (see figure 1(a)). Each state is described by a vector of 4 features, the first two gives the x, y coordinate of the cell, and the second two constitutes a description of an adhesive chemical present in the cell. These take values in the range $0 - 100$. So there are $10,000$ possible adhesives. Each adhesive belongs to one of 5 possible groups. For each group there is a neutralizing chemical which, when deployed by the agent, removes its adhesive properties until the agent leaves the cell. The agent cannot leave a cell until the adhesive in the cell is neutralized. Adhesives belonging to a particular type has a learnable description - for each type there are 5 different blocks of size 20×20 spread around the 100×100 feature space. $0, 0$ value of the adhesive description features represents a neutralized chemical.

At each step the agent can try to deploy one of 8 possible chemicals (5 of which are the neutralizing ones) or move in one of the 4 cardinal directions. Each action works with probability 0.8 and does something random with probability 0.2. We call the states containing the same type of adhesive $\xi = 0$ *functionally similar*, with respect to the Deploy-Chemical-X where X is the corresponding neutralizer. This is because the $L1$ norm between the distribution over the *effect* of the action Deploy-Chemical-X is 0 in all these states i.e. with probability 0.8 the adhesive is neutralized and with probability 0.2 something random happens. The same holds true for the Move-NORTH action (and for other motion actions) for states that are at least one cell away from the boundary, and for the same reasons.

The central idea is this. Given the agent has observed sufficiently many states that are $\xi = 0$ FS with respect to action Deploy-Chemical-X, it can learn the feature space description of such states (which is learnable) and identify if a cell contains the same type of adhesive. Now it can predict the effect of the action on the cell with error close to zero. This becomes useful in planning since if we can predict the distribution over the effects of an action on a novel state, we can also predict the resulting next state (consider using classical planning like methods in domains with actions with unknown and probabilistic effects). □



(a) Each color is a different adhesive type.



(b) Results for the Sticky Room domain

Figure 1: Sticky Room Domain. Figure 1(a) shows the domain. Figure 1(b) shows no. of actions/steps required for the each agent to converge to the optimal cost policy to goal state.

Functional Similarity. Now let $\mathcal{F}_k$ ('feature functions') be the set of functions of the form $f : S \to S$, The set $\mathcal{F}_k$ corresponds to the set of possible effects of actions. We assume $\mathcal{F}_k$

is given as prior knowledge (see **Theoretical Properties of the TPM**). The feature function in the above example are vector functions which apply one of the following operations to each feature : $x \rightarrow x+1, x \rightarrow x-1$ and $x \rightarrow 0$ where x is a feature value.

We further assume that if $f_i, f_j \in \mathcal{F}_k$ with $i \neq j$, then $f_i(x) \neq f_j(x)$. This constraint implies that for each distribution over next states $P(s'|s,a)$, there is a unique distribution over $f \in \mathcal{F}_k$ given by $P(s' = f(s)|s,a) = P(f|s,a)$. This distribution is unique because for each s' there is at most one $f \in \mathcal{F}_k$ with $s' = f(s)$. Now, following the example above, we define two states s, s' to be ξ functionally similar with respect to an action a if the $L1$ norm between the distributions $P(f|s,a)$, $P(f|s',a)$ is less than ξ , that is

$$L1[P(f|s,a), P(f|s'a)] = \sum_{f \in \mathcal{F}_k} |p(f|s,a) - p(f|s',a)| < \xi \tag{2.1}$$

Transition Prediction Model. Now as mentioned above, for each $P(s'|s,a)$ there is a unique $P(f|s_i,a)$ with $P(s'|s,a) = P(f|s,a)$. So $L1[P(f|s_i,a), P(f|s_j,a)] < \xi \Leftrightarrow L1[P(s'|s_i,a), P(s'|s_j,a)] < \xi$. Following the example above, the idea here is to use this fact to construct a *Transition Prediction Model* to predict for some novel state s_{nov} the distribution $P(f|s_{nov},a)$ over the effects, and hence the distribution $P(s'|s_{nov},a)$ over next states, with error $< \xi$ for some suitable value of ξ. A TPM is defined by the set of pairs $\{(Sa, Ca) : a \in A\}$. Each $Sa = \{Sa_i : 1 \leq i \leq n_i\}$ is a set of disjoint sets of states, with each Sa_i corresponding to a group of ξ_{Sa_i} (see below) similar states. The classifier Ca (SVM, Neural Nets etc.) is trained with each Sa_i as a target concept. Given a novel state s_{nov}, the Ca is used to determine which Sa_i the state belongs to. Then the distribution $P(f|s,a)$ for some $s \in Sa_i$ is used as a prediction for $P(f|s_{nov},a)$. We assume the estimate of $P(f|s_{nov},a)$ given by the TPM is used to compute some function v, and the value of ξ_{Sa_i} for each Sa_i is set such that the error in estimating v is acceptable. *The value function in 3.1 is such a v function and the TPM may be used to estimate it.*

Theoretical Properties of the TPM. Our results for the TPM show that under reasonable conditions (see below), states in each Sa_i, constructed by the TPM, contain similar 'objects'. Here, an object is any part of the environment that behaves similarly with respect to different actions and has a learnable physical/feature space description. This property is also what identifies real world objects, and so this an interesting result. The 'reasonable conditions' above roughly means that the elements of $\mathcal{F}_k$, given as prior knowledge, have the property that their domains are a union of learnable concepts (in the PAC sense). That is $\mathcal{F}_k$ should contain elements that cover many transitions (s,s') we are likely to observe and also has a compact algorithmic description. We contend that providing a $\mathcal{F}_k$ with this property is not an onerous requirement since the experimenter decides what the features are and therefore she should be in a position to determine which feature functions are likely to be observed.

3 Using the TPM

We empirically demonstrated the efficacy of TPM by using it to learn tasks in certain types of goal directed MDPs (see [2]). The type of goal directed MDPs we consider have constant cost for each state action pair (s,a). Formally, the MDPs we consider are defined by the tuple (S, A, I, G, d) where S and A are as defined above, I is a set of start states, G is a set of goal states and d is the constant cost per action. The task of the agent is to go to any $s_G \in G$ starting from any $s_I \in I$ by selecting action a_{min} at each step:

$$a_{min} = \arg\min_{a \in A} Q'_a(s, s_g) = \arg\min_{a \in A} \left(d + \sum_{s' \in S} p(s'|s,a) \gamma \min_{a' \in A} Q'_a(s', s_g) \right) \tag{3.1}$$

We compared the performance of Prioritized Sweeping (PS) [3] and PS augmented by a classical planner style mechanism that uses the predictions made by TPM to predict a path from the current state to any goal state when PS has not converged. We used the Sticky Room domain described above for this comparison. A particular task of the agent in this domain consists of the agent going from some given start state to some given goal state. Each agent solved 6 different tasks in sequence, with each task differing in the goal state, start state, and configuration of the chemicals and the chemicals that are present. We expected the PS+TPM agent to do better by transferring information across tasks and states because, as described in the example 2.1, states in different tasks are functionally similar with respect to all the actions. The results, which agree with expectations, are shown in figure 1(b).

4 Discussion

At this point we are actively engaged in extending our system in continuous valued MEs and are working towards implementing it in a mobile robot to transfer information across tasks involving pushing around different kinds of objects. We are also working towards extending our method to work on goal directed MDPs with non-constant costs and regular MDPs. We are also constructing theories to determine where FS fits in as a knowledge transfer mechanism for rational agents in general. It will also be interesting to extend our work to apply functional similarity at multiple levels of abstraction within the same task.

Now let us look at some relevant previous work. The difference between previous forward models (FMs) such as, [4] and, [5] (Belief net based) is that these learn to approximate the feature functions directly while we use the prior knowledge about the feature functions (which we claim is available for many domains of interest). Hence we only need to solve a classification task (the Cas) instead of the much more difficult regression task being computed by the FMs, and so can learn to predict with much fewer examples and greater accuracy. Also, our theoretical results suggest principled ways to learn the feature functions which bears further investigation. Our method is also closely related to state space aggregation methods such as [5]. The difference between these methods and ours is that the aim of aggregation is to reduce computation time rather than improving prediction. It should be interesting to explore applying FS ideas to aggregated state spaces. The general research area that our method is most closely related to is probably, Lifelong Learning ([6]). In these methods, the aim is to have an agent that lives for a long time and gradually accumulates knowledge. It then uses this knowledge to make itself better. The difference between these methods and ours is the use of FS to acquire knowledge.

Acknowledgement We would like to thank Samarth Swarup and Kiran Lakkaraju for their comments.

References

[1] M. M. Hassan Mahmud. Exploiting functional similarity for information transfer (unpublished). *https://netfiles.uiuc.edu/mahmud/shared/FS.pdf*, 2005.

[2] A.G. Barto, S.J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

[3] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[4] A. Karniel. Three creatures named 'forward model'. *Neural Network*, 15:305–307, 2002.

[5] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121:41–107, 2000.

[6] S. Thrun and L. Y. Pratt, editors. *Learning To Learn*. Kluwer Academic Publishers, Boston, MA, 1998.