Better Generalization with Forecasts

Tom Schaul
Courant Institute of Mathematical Sciences
New York University
715 Broadway, New York, NY 10003, USA
Email: schaul@cims.nyu.edu

Mark Ring
Email: research@markring.com

Abstract

Predictive methods are becoming increasingly popular for representing world knowledge in autonomous agents. A recently introduced predictive method that shows particular promise is the General Value Function (GVF), which is more flexible than previous predictive methods and can more readily capture regularities in the agent's sensorimotor stream. The goal of the current paper is to investigate the ability of these GVFs (also called "forecasts") to capture such regularities. We generate focused sets of forecasts and measure their capacity for generalization. We then compare the results with a closely related predictive method (PSRs) already shown to have good generalization abilities. Our results indicate that forecasts provide a substantial improvement in generalization, producing features that lead to better value-function approximation (when computed with linear function approximators) than PSRs and better generalization to as-yet-unseen parts of the state space.

1 Overview

One of the grandest goals of AI is a continual-learning agent, capable of constantly extending its skills and its understanding of its world, building step by step on top of what it has already learned [Ring, 1994]. Such an agent requires a method for capturing and representing the important features and regularities of its sensorimotor stream. Prediction has emerged as a particularly powerful principle for organizing knowledge and skills, focusing the agent's representational efforts on making and testing verifiable hypotheses about the consequences of its actions [Sutton, 2001]. A recently proposed representation is the *General Value Function* (GVF) [Sutton *et al.*, 2011; Maei and Sutton, 2010; Sutton *et al.*, 2006; Modayil *et al.*, 2012], which offers a rich and expressive language for action-conditional prediction, resolving many of the limitations of previous predictive methods.

This paper is a focused experimental investigation of GVFs (sometimes called *forecasts*). We introduce no new learning techniques or algorithms; rather we examine a narrow subclass of forecasts (GVFs) and take a first look at their ability to capture important regularities. We specifically wish to compare them to an earlier predictive method, *Predictive State Representations* or PSRs [Littman *et al.*, 2002], to decide whether forecasts are likely to be a better method for capturing the useful features and regularities of a learning agent's environment. Thus, we need to test how well they *generalize*.

While generalization in static, supervised-learning problems has been the standard measure of comparison for decades, dynamic learning problems such as robotics and reinforcement-learning tasks are complicated by the interaction of the agent and do not admit so readily to such measures. Consequently tests for generalization are far less common.

Yet generalization is particularly important to continual-learning agents, which may never experience more than a minuscule fraction of the states in their environments but must nevertheless capture the most useful regularities and exploit these over a potentially vast state space.

Predictive methods of state representation differ from so-called "historical" methods in that their focus is not on remembering what the agent has seen in the past, but on predicting what the agent might see in the future.[1] PSRs are the most widely studied predictive methods, but there are others, such as Simple-Assignment Automata [Rivest and Schapire, 1994], Observable Operator Models [Jaeger, 2000], and TD Networks [Sutton and Tanner, 2005]. These methods represent the agent's state information as a set of features, each an action-conditional prediction of a future observation. A PSR feature, for example, estimates in each state the probability of making a specific observation if the agent were to take a specific sequence of actions starting in that state.[2]

Forecasts (GVFs) are similar to PSRs in the general sense that each feature estimates the outcome of following a specific course of behavior. But a crucial difference is that this course of behavior is not an open-loop sequence of actions, but a closed-loop option [Sutton *et al.*, 1999]: a mapping from states to actions (a policy) together with the conditions for the policy's initiation and termination. Thus, forecasts are more general, more flexible, and have the ability to capture more temporally indefinite regularities than PSRs. Further enhancing their capacity for abstraction and generalization, forecasts

[1] There is a broad gray area in between, where algorithms such as context trees [Willems *et al.*, 1995] or, more immediately, Temporal Transition Hierarchies [Ring, 1993], learn to make specific predictions about the future based on finding specific patterns in the past.

[2] There are actually a variety of slightly different types of PSRs.

can also be *composed* or *layered* in two ways: first, one forecast can learn to predict the (option-conditional) value of another; second, the policy learned for one can be used as the policy of another.

One of the most important advantages of forecasts is the existence of so-called "off-policy" learning methods, allowing large numbers of them to be trained simultaneously [Sutton *et al.*, 2011]. Each learns to make different predictions about different kinds of behavior from the agent's single stream of sensorimotor data. This vital capability makes forecasts perhaps the best existing candidate for continual learning; however, in the current study, we will not need to make use of those learning methods. Because we are focusing exclusively on the issue of representation, we bypass the complexities of off-policy learning altogether and do batch updates with a full model of the environment.[3]

Furthermore, the current study investigates only a small subclass of forecasts, focusing on some of their most basic capabilities. Specifically, while a forecast is defined as a five tuple, we hold two constant and greatly constrain a third.

Because there is no existing method for discovering new forecasts in a principled way (and we are not proposing one) we simply create all possible forecasts, limited to our constraints, in a canonical ordering and look at resulting performance. This approach, inspired by a similar 2005 investigation of the representational power of PSRs [Rafols *et al.*, 2005] allows us to ask: what might happen if we had a mechanism for building new forecasts? Is there reason to believe that they would generalize well, capturing important regularities of the environment?

The results of our tests answer this question rather convincingly in the affirmative.

2 Background

Because forecasts have much in common with PSRs and TD Networks, we describe these first, and do so within the framework of Markov decision processes (MDPs). An MDP consists of a set of states ($s \in \mathcal{S}$), actions ($a \in \mathcal{A}$), observations ($o \in \mathcal{O}$) and rewards ($r \in \mathbb{R}$). At every time step, the agent receives an observation o_t and reward r_t in its current state s_t and takes action a_t which leads the agent to the next state s_{t+1}, depending on the state transition probabilities $T(s, a, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$.

PSRs represent each state as a set of features (called "tests") where each is a prediction about an observation that might result from executing a specific sequence of actions. There are several varieties of PSRs with slightly different properties. In one sufficiently general variety, a test $q(o, a^1, \ldots, a^k)$ represents the agent's probability of making a specific observation o after taking a specific string of k actions $a^1, \ldots, a^k$:

$$q(o, a^k) \equiv \Pr(o_{t+k} = o \mid a_t = a^1, \ldots, a_{t+k} = a^k). \quad (1)$$

[3]To be clear: the methods we employ for the experiments have little practical application. Because the subject under investigation is only the representation and not the learning, we choose a convenient method for generating the representations and their values.

Short sequences of actions make short-term predictions; longer sequences make longer-term predictions. If two states are distinguishable, there will be a series of actions that can be taken in each that will result in a different expected observation. Thus, each PSR feature has two components: (1) its definition (*i.e.,* specification of which observation should follow which sequence of actions), and (2) its value (the probability of Equation 1) in each state.

TD Networks contain a set of nodes that each make an action-conditional prediction either about an observation (as with PSRs) or about the value of another node in the network. As with PSRs, each node has two parts: a definition and a value. The definition describes or specifies what the node is making a prediction about, and the value is an estimate of the predicted quantity, which can vary from state to state, and is a learned function of the observations and features.

As with PSRs and TD Nets, a forecast or GVF also has the same two parts: a definition and a value. Forecasts are quite similar in spirit to the other two but are considerably more sophisticated, general, and flexible. Rather than making a prediction about the result of following a specific fixed sequence of actions, a forecast predicts the result of following an *option* until it *terminates*.

Each forecast definition consists of two parts, an *option* and an *outcome*. The *option* [Sutton *et al.*, 1999] is a 3-tuple (π, I, β), where π is a policy, which maps states (as represented by the agent) to a probability distribution over actions; $I : \mathcal{S} \to \{0, 1\}$ is the *initiation set* (specifying the states in which the policy can be started); and $\beta : \mathcal{S} \to [0, 1]$ is the *termination probability* (the probability of the option terminating in each state). The option describes a possible way for the agent to behave, along with conditions about where that way of behaving can begin and end. Each forecast predicts what the *outcome* will be if the option is followed (*i.e.,* if the agent behaves as described by the option).

The *outcome* is a tuple (c, z), where $c : (\mathcal{S} \times \mathcal{A}) \to \mathbb{R}$ is a *cumulative* value defined for every state-action pair reachable while the option is being followed, and $z : \mathcal{S} \to \mathbb{R}$ is a *termination* value, defined wherever termination may occur.

Thus, every forecast definition f^i describes a function of the state according to these five components:

$$f^i(s) \equiv f^{\pi^i, I^i, \beta^i, c^i, z^i}(s)$$

(For clarity in the description of an individual forecast, we now drop the superscript i.)

The *value* of a forecast is the expected sum of all the cumulative c values encountered while the option is being followed, plus the termination value z at option termination at some future time step k. More precisely, the forecast value for a state $s \in I$ is

$$f(s) = \mathbb{E}\left[c_1 + c_2 + \ldots + c_{k-1} + z_k \mid \pi, \beta, s_0 = s\right]. \quad (2)$$

Thus, the forecast value is a prediction about the expected sum of c values while the agent is following the option, plus the expected z value when the option terminates. To avoid infinite sums, one may constrain β to (0,1], ensuring that all options will eventually terminate.

Although c and z can be any function of the state, one useful special case occurs when c is zero everywhere, z is binary,

and $\beta = 1$ wherever $z = 1$. In this case, the forecast value represents the agent's option-conditional probability of entry into the set of states where z is 1.

Thus, forecasts are action-conditional predictions that are significantly more flexible than PSRs and TD Nets. In particular, the number of steps that might elapse until termination of a forecast is not explicit in its definition, allowing the prediction of an arbitrary condition of the state within a loose time frame. In fact, forecasts are very similar to the value function in reinforcement learning (hence the term "general value function") but can be used to predict *any* function of the state, not just the reward. Thus, a unique advantage of forecasts is that their policies can be optimized to maximize or minimize the forecasted value. We call such forecasts "active" and distinguish them from "passive" forecasts which have static policies.

Note that though "the forecast" is occasionally unambiguous, generally one must specify whether one means the forecast *definition* f (specification of the option and outcome) or the forecast's *ideal value* $f(s)$ (Equation 2). And besides these, there is also the agent's *estimate* of the ideal value $\hat{f}(s)$, because, just as with PSRs and TD Nets, the learning agent must *learn* the values of those predictions. In work published so far with GVFs, off-policy temporal-difference (TD) methods combined with linear function approximators are used to calculate and make continual improvements to all the forecast estimates at every step [Sutton *et al.*, 2011].

3 Methods

Forecasts are complex and admit many nuances too subtle to investigate thoroughly in a short space. We therefore choose to focus on a narrow but important subset of the full forecast, a greatly simplified version that retains some of their most promising properties. In particular, we consider only the cases where: $I = \mathcal{S}$ (all policies can be initiated in all possible states), $c(s) = 0$ (there is no cumulative value in the outcome), $z(s) \in \{0, 1\}$, and $\beta(s) = 1$ if $z(s) = 1$ but 0.1 otherwise, for $s \in \mathcal{S}$. Thus, we study the case where the forecast estimates the probability of terminating in a state where $z = 1$, which is inversely related to the number of steps the agent needs to reach such a state. (This is a slight departure from most papers written so far on this method, which tend to focus on the case where $z = 0$ and c is a positive value [Sutton *et al.*, 2011; Modayil *et al.*, 2012; Degris and Modayil, 2012]. We find that our simplification is as intuitive, but in some cases easier to work with.)

Many (perhaps all) papers on GVFs published so far have focused on their usefulness in the continual-learning paradigm, where learning occurs at every step and the agent is always learning new things on top of what it already knows. With that focus, an online learning algorithm is essential. Since our goal is different—merely to test the representational abilities of the forecast mechanism—we prefer to use batch methods to compute the ideal forecast values for small state sets. These batch methods assume a full model of the environment and in general cannot be used to solve the learning problem. The recent off-policy learning methods with proven convergence guarantees (as mentioned in Section 1) remain the methods of choice for the continual-learning case.

Just as Rafols *et al.* (2005) did with PSRs, we wish to create forecasts in a canonical and automated way, from simple to complex, then measure the contribution of each as it is added. Though there are in principle many possible ways to do so, we have chosen a layered approach in which new active forecasts are added that predict and attempt to achieve values of already known features.

Forecasts are created, optimized, and evaluated in an incremental process detailed in Algorithm 1 starting with forecast f^1. For simplicity, all agent observations in all our tests are binary, and the algorithm begins with a vector of observation functions $\mathbf{o}$, where each function produces a binary value in each state, $o \in \mathbf{o} : \mathcal{S} \to \{0, 1\}$. Because they are binary, one can view each observation function as describing a set of states (in which the observation is 1), and Υ is an ordered list of these sets and their complements (Line 3). For each forecast f^j, $I^j = \mathcal{S}$ (Line 6) and $c^j = 0$ in all states (Line 8). The z values are based on Υ^j, the j^{th} state set in the list; specifically, $z^j(s) = 1$ iff $s \in \Upsilon^j$. All forecasts are *active*, so policy π^j is optimized to maximize f^j according to Equation 2 (Line 15). We use a perfect model of the environment and full state information to calculate the ideal value for each forecast in each state (Line 16). The median of those ideal values then becomes a threshold (Line 17) used to split the states into two sets that are then appended to the list Υ (Line 18). Forecast creation continues until N forecasts have been created and evaluated in each state.

Algorithm 1: Create N forecasts

```
1  Υ ← ∅
2  for o in o do
3      Υ ← Υ ∘ {s | o(s) = 1} ∘ {s | o(s) = 0}
4  for j ← 1 to N do
5      create forecast f^j such that :
6          I^j = S
7          for s ∈ S do
8              c^j(s) ← 0
9              if s ∈ Υ_j then
10                 z^j(s) ← 1
11                 β^j(s) ← 1
12             else
13                 z^j(s) ← 0
14                 β^j(s) ← 0.1
15         π^j ← optimal policy using policy iteration
16     compute f^j(s) for all s ∈ S
17     Θ = median_{s∈S}{f^j(s)}
18     Υ ← Υ ∘ {s | f^j(s) < Θ} ∘ {s | f^j(s) ≥ Θ}
19     delete duplicate sets from Υ
```

The above approach would not work for a true continual-learning agent, because the agent would not have access to the full state information of the environment (among other reasons). We use it because it provides a straightforward canonical method for producing new forecasts. An actual continual-

learning agent must create forecasts based on its individual experience, using a different methodology not yet developed.

The N ideal values for forecasts f^1 to f^N form a set of *features* whose quality we now wish to evaluate. To evaluate a set of features as a state representation for a reinforcement-learning agent, we combine them with the agent's observations into a feature vector and then compute the following three measurements: (1) The mean-squared error (MSE) between the true value function V (computed with a perfect model and full state information) and $\hat{V}$, the best linear approximation of V based on the feature vector. (LSTD [Bradtke *et al.*, 1996; Boyan, 1999] with a full transition model of the task MDP computes the optimal parameters of the linear function approximator.) This value will be designated "MSE" in our graphs. (2) The average value of each state according to the true value function for policy $\hat{\pi}_f$, where $\hat{\pi}_f$ is the best policy that can be computed as a linear function of the feature vector (using policy iteration with LSTD for the evaluation step). This value is designated "LSTD-PI" in the graphs. (3) Same as (1) above but $\hat{V}$ is computed using only a randomly selected fraction of the states (specifically, 50% and 90%), averaged over 25 random selections of state sets. These measurements provide an indicator of a feature set's ability to generalize to unseen parts of the state space.

Our experiments use two discrete grid worlds (Figure 1). The first is inspired by one from Rafols *et al.* [2005]. The second is designed to have regularity at varying scales. In both, the agent has two actions: go forward or rotate 90 degrees left ($|A| = 2$). In each case, the state space consists of position and orientation, so $|S| = 4p$, where p is the number of positions. The agent observes just one bit, namely whether it has a wall immediately in front of it ($|O| = 1$), and is rewarded for visiting an (invisible) goal position. Both environments are implemented with **pyvgdl** [Schaul, 2013], an open-source, video-game description language (VGDL) in Python, which allows automatic generation of different configurations, game mechanics (stochastic or deterministic), and a full MDP model (matrix of exact transition probabilities), from a simple description.

4 Results

The top graphs of Figures 2 and 3 show an incrementally increasing number of features, which are either forecast values (generated according to Algorithm 1), or for comparison, PSR-test values. PSR tests are generated according to the shortest-first method described by Rafols *et al.* [2005], in which all tests of length k (having action sequences of length k) are generated before any test of length $k + 1$, beginning with $k = 1$.

In the lower graphs (again following Rafols *et al.*), states are aggregated into as many *classes* as the forecasts (or PSRs) can disambiguate. That is, each state belongs to exactly one class, and two states belong to the same class if and only if they cannot be distinguished by any PSR test (left) or forecast (right). In these graphs, the feature vector consists of one binary feature per class, where each feature value is 1 in exactly those states that belong to the class. Note that sometimes multiple forecasts (or PSRs) need to be added before a new class arises. Other times a single forecast (or PSR) can produce many new classes at once (see the jump in Figure 2 from 50 to 90 classes with just one additional forecast).

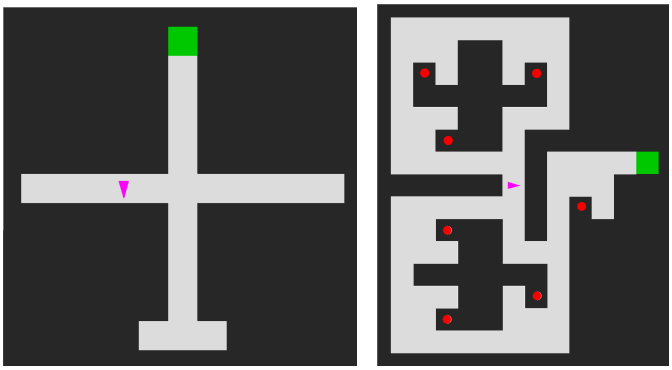

Figure 1: Two grid worlds used for the experiments. **Left**: A cross-shaped corridor with 23 positions (92 states) and a reward (green square, not visible to the agent) in one of three identical-looking arms. The fourth arm can potentially be distinguished by the agent and used for orientation. **Right**: An 82-position (328 state) world with 7 identical-looking rooms; each with two exits, one marked by a protruding wall (red dot for our view only, invisible to the agent).

Figure 2 shows results for the cross task, designed to test whether forecasts can discern and use the disambiguating feature at the end of one of the hallways, something that the short-action-sequence PSRs should not be able to do. The results show that the task can be solved by all measures with as few as 80 forecast features, while PSRs fall short.

Figure 3 highlights the generalization capability of forecasts in the larger environment. While PSR features start overfitting long before they allow for reasonable performance (pink curve going off the chart in the upper left graph), forecast features generalize very well: performance degrades only minimally, even when half the states are never seen (upper right graph). However, generalization is severely impaired, both for PSRs and forecasts, when class features formed by state aggregation are used instead (bottom two plots).

Figure 4 visualizes the features produced by PSRs and by forecasts. When studied closely, these images reveal the heart of the investigation: forecasts find important regularities in the environment.

5 Discussion

Figures 2 and 3 show that as new forecasts are added, approximation of the value function steadily improves and average reward steadily increases, even though the reward signal plays no role in the construction of the forecasts.

The agent's immediate sensorimotor stream is minimally informative, yet forecasts are able to produce features that distinguish subtle spatial structures. Furthermore, they can carry this information to distant states, allowing the agent to distinguish regions of the world that are nearly identical (in the sense that they generate identical responses to all short- and medium-length sequences of actions). In order to choose the best action in most of the states of Task 1, it is essential for the agent to know which arm of the cross it is in, yet the only information that can distinguish the arms is located at the

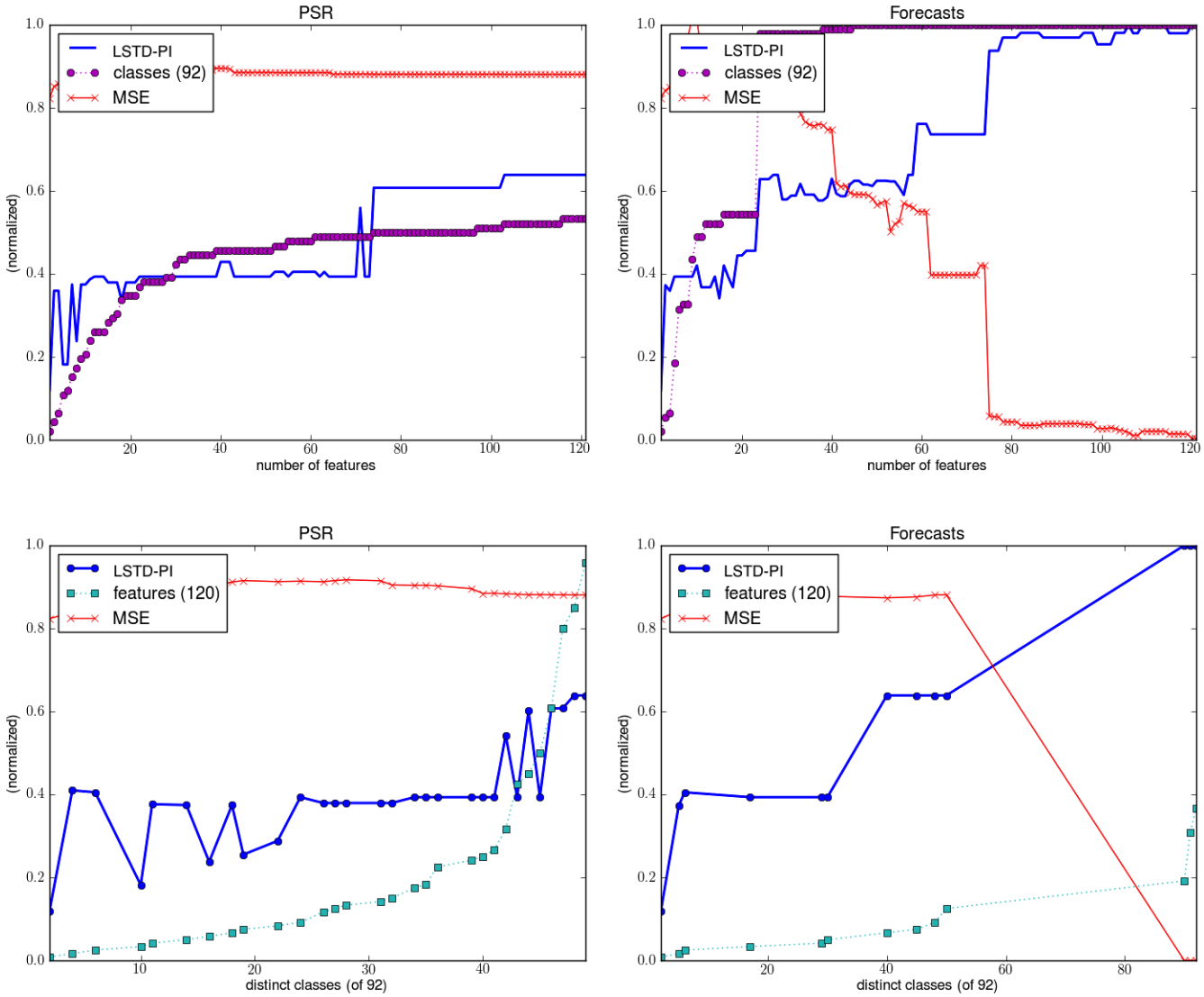

Figure 2: Quality of the 120 first features generated in the cross-shaped corridor world. Left column: PSR tests; right column: forecasts. **Above**: The horizontal axis shows the (increasing) cumulative number of features used. Two quality measures are shown, normalized between 0 and 1:"MSE" is the mean-squared error for the best linear approximation of the optimal value function with these features, and "LSTD-PI" shows the average expected reward for the best feature-based policy averaged across all states. The violet circles indicate how many classes of states can be distinguished using the features: if this curve reaches 1, then *all* states can be disambiguated in principle (but not necessarily by a linear function approximator). **Below**: Performance curves as above but feature vectors now identify which class each state belongs to. The light blue squares indicate the fraction of total features required to distinguish the classes.

distant end of one arm. It is surprising both how readily forecasts are able to capture this information and how easily they are then able to use it to distinguish all the states of the MDP. In the case of PSRs, it is clear from Figure 2 that a very large number of features must be constructed in shortest-first order before this kind of information will be available everywhere in the MDP. Furthermore, there is essentially no hope that the fixed-length PSR features found useful for Task 1 would be particularly useful if the length of the arms were extended. In contrast, it seems quite likely that the forecasts generated for a smaller cross would still be useful in a larger one.

Figure 3 investigated the generalization ability of forecasts and showed that even with exposure to only 50% of the states, the forecast features are sufficient to produce a good policy and to get good evaluation of the value function using a linear function approximator. The similarity of Figures 2 and 3 is striking, despite the very different environments. (Although omitted here for space considerations, we did the same experiments in numerous other grid-world environments and saw similar graphs.)

Perhaps the most striking results, though, are those of Figure 4. The distinction between the PSR approach and the GVF approach are apparent almost instantly. Though the first few features look similar, beyond that the differences

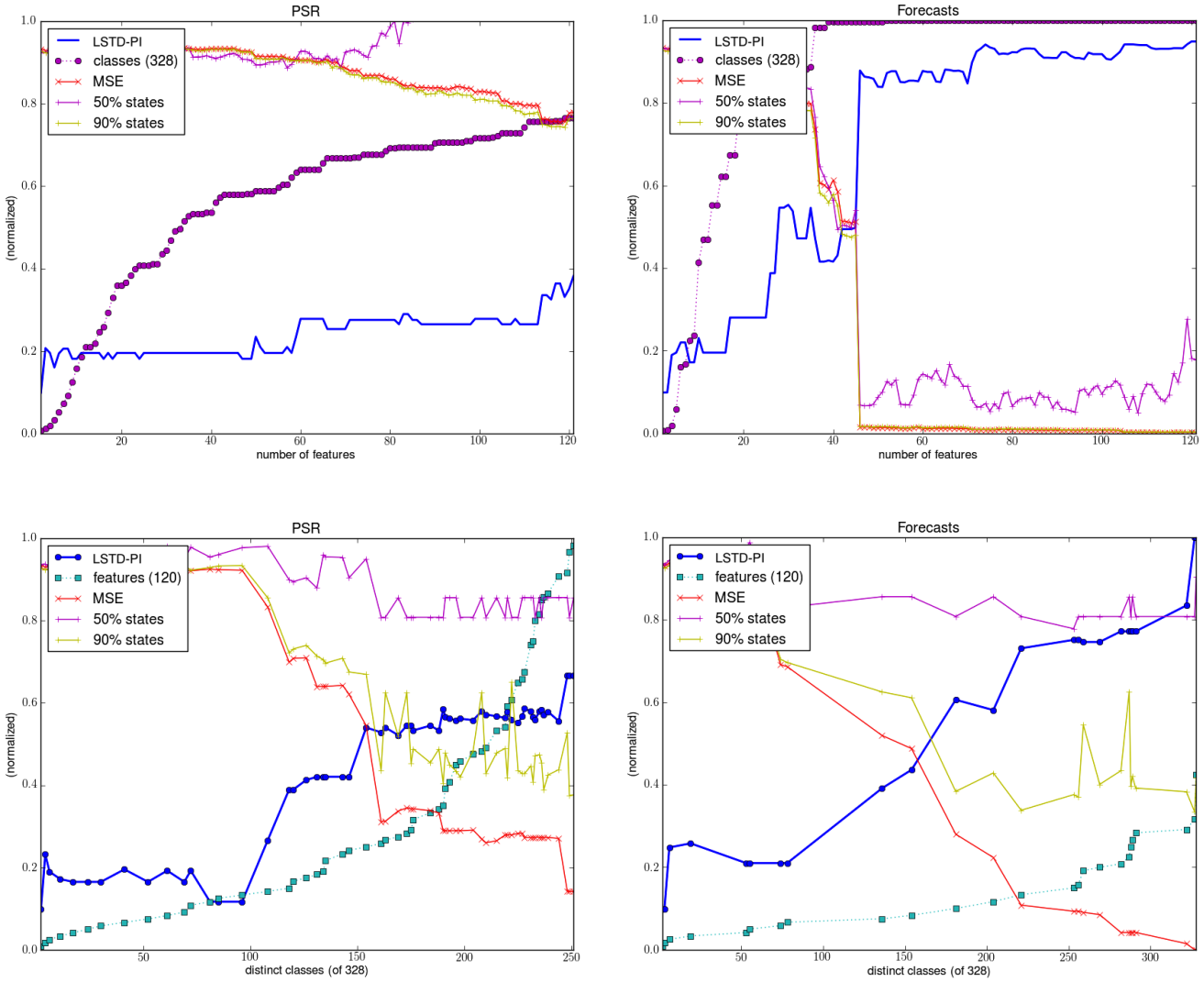

Figure 3: Generalization quality of the 120 first features in the 7-room grid world: PSRs (left), forecasts (right). Forecasts again drastically outperform PSRs. Generalization is measured by leaving out a fraction (10%, yellow curve, or 50%, pink curve) of all states from the transition data used by LSTD to generate the parameters of the linear function approximator. Both curves (each the median over 25 runs) measure MSE, and should be compared to the red line, which is computed from complete state data.

are rather extraordinary. While nothing about Algorithm 1 requires or encourages human readability, it is surprisingly easy to identify what many of the forecasts have captured (as shown in the caption). In contrast, virtually none of the PSR features can be interpreted in any meaningful way. Though this is not necessarily a weakness of PSRs, it may be a result of their inability to capture non-local relationships, which is also immediately apparent from the figure.

6 Summary and Conclusions

The goal of this work was to investigate the representational promise of forecasts (GVFs) in an impartial manner. We created a simple mechanism for producing simple forecasts in a breadth-first fashion and then tested their suitability as features for a reinforcement-learning agent. We devised two tasks rife with non-annotated structural regularity to see whether forecasts could identify and use some of that structure. And finally, we compared the results to PSRs using a similar generative mechanism. The results are surprisingly clear and do unquestionably support the hypothesis that forecasts are capable of capturing significant and substantial environmental regularity of at least those kinds tested. They are able to capture small-scale and large-scale relationships and encode them in a way that is useful for task achievement and strong generalization.

It is worth reiterating that our experiments used only a narrow subclass of forecasts chosen to be easiest to work with, and our forecast-creation mechanism was brute force. (We resisted the temptation to make this process more intelligent, though the possibilities here for future work immediately present themselves.) Nevertheless, the power of this

Figure 4: Each of the 120 images shows the value of a given feature for each state in the 7-room grid world, as produced by PSR tests (top) and forecasts (bottom). Walls are black, and feature values are color coded on a normalized scale from red (maximum) to white (minimum). Each is labeled with an index indicating in which order the feature was generated. Note how many forecasts appear to encode interesting properties of the environment: forecast 8 locates the markers; 38 identifies the dead-end corridor (where the goal will be hidden); 50 reveals a high-level symmetry based on multiple rooms (the upper 3 are a rotated version of the lower 3); 61, corridor crossings; 69, distance to the dead end; 76, non-room corridors that connect rooms; 88, a room that is not in a group of 3; 100, furthest point from the dead end, etc.

representation is unmistakable.

Acknowledgments

This work was funded in part through AFR postdoc grant number 2915104, of the National Research Fund Luxembourg.

References

[Boyan, 1999] Justin A. Boyan. Least-squares temporal difference learning. In *In Proceedings of the Sixteenth International Conference on Machine Learning*, pages 49–56. Morgan Kaufmann, 1999.

[Bradtke *et al.*, 1996] Steven J. Bradtke, Andrew G. Barto, and Pack Kaelbling. Linear least-squares algorithms for temporal difference learning. In *Machine Learning*, pages 22–33, 1996.

[Degris and Modayil, 2012] Thomas Degris and Joseph Modayil. Scaling-up knowledge for a cognizant robot, 2012.

[Jaeger, 2000] Herbert Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6):1371–1398, 2000.

[Littman *et al.*, 2002] Michael L. Littman, Richard S. Sutton, and Satinder Singh. Predictive representations of state. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[Maei and Sutton, 2010] Hamid R. Maei and Richard S. Sutton. GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *AGI'10*, 2010.

[Modayil *et al.*, 2012] Joseph Modayil, Adam White, Patrick M. Pilarski, and Richard S. Sutton. Acquiring diverse predictive knowledge in real time by temporal-difference learning. In *International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems*, Montpellier, France, 2012.

[Rafols *et al.*, 2005] Eddie J. Rafols, Mark B. Ring, Richard S. Sutton, and Brian Tanner. Using predictive representations to improve generalization in reinforcement learning. In L. P. Kaelbling and A. Saffiotti, editors, *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 835–840, 2005.

[Ring, 1993] Mark B. Ring. Learning sequential tasks by incrementally adding higher orders. In C. L. Giles, S. J. Hanson, and J. D. Cowan, editors, *Advances in Neural Information Processing Systems 5*, pages 115–122, San Mateo, California, 1993. Morgan Kaufmann Publishers.

[Ring, 1994] Mark B. Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, University of Texas at Austin, Austin, Texas 78712, August 1994.

[Rivest and Schapire, 1994] Ronald L. Rivest and Robert E. Schapire. Diversity-based inference of finite automata. *J. ACM*, 41(3):555–589, 1994.

[Schaul, 2013] Tom Schaul. PyVGDL: a video game description language in python. https://github.com/schaul/py-vgdl, 2013.

[Sutton and Tanner, 2005] Richard S. Sutton and Brian Tanner. Temporal-difference networks. In *Advances in Neural Information Processing Systems 17*. MIT Press, 2005.

[Sutton *et al.*, 1999] Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[Sutton *et al.*, 2006] Richard S. Sutton, Eddie J. Rafols, and Anna Koop. Temporal abstraction in temporal-difference networks. In *Advances in Neural Information Processing Systems 18 (NIPS*05)*, pages 1313–1320. MIT Press, 2006.

[Sutton *et al.*, 2011] Richard S. Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, and Doina Precup. Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '11, pages 761–768, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

[Sutton, 2001] Richard S. Sutton. Verification, the key to ai. http://webdocs.cs.ualberta.ca/ sutton/IncIdeas/KeytoAI.html/, 2001. Available online.

[Willems *et al.*, 1995] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41:653–664, 1995.