

The Two-Dimensional Organization of Behavior

Mark Ring, Tom Schaul, Juergen Schmidhuber
IDSIA / University of Lugano / SUPSI
Galleria 2
6928 Manno-Lugano, Switzerland
Email: mark,tom,juergen@idsia.ch

Abstract—This paper addresses the problem of continual learning [1] in a new way, combining multi-modular reinforcement learning with inspiration from the motor cortex to produce a unique perspective on hierarchical behavior. Most reinforcement-learning agents represent policies monolithically using a single table or function approximator. In those cases where the policies are split among a few different modules, these modules are related to each other only in that they work together to produce the agent’s overall policy. In contrast, the brain appears to organize motor behavior in a two-dimensional map, where nearby locations represent similar behaviors. This representation allows the brain to build hierarchies of motor behavior that correspond not to hierarchies of subroutines but to regions of the map such that larger regions correspond to more general behaviors. Inspired by the benefits of the brain’s representation, the system presented here is a first step and the first attempt toward the two-dimensional organization of learned policies according to behavioral similarity. We demonstrate a fully autonomous multi-modular system designed for the constant accumulation of ever more sophisticated skills (the continual-learning problem). The system can split up a complex task among a large number of simple modules such that nearby modules correspond to similar policies. The eventual goal is to develop and use the resulting organization hierarchically, accessing behaviors by their location and extent in the map.

I. BACKGROUND AND PHILOSOPHY

This paper addresses the problem of continual learning by introducing and describing a two-dimensional topological map in which behaviors are organized according to their similarity. Continual learning was first described by Ring (1994) as “the constant development of complex behaviors with no final end in mind”. It is the process of “building more complicated skills on top of those already developed, ... the transfer of skills developed so far towards the development of new skills of greater complexity” [2].

The principal requirements of such systems include the ability to learn new behaviors incrementally and hierarchically based on existing behaviors while avoiding catastrophic forgetting [1]. While constantly trying to maximize its reward, a continual-learning system should be able to learn new skills indefinitely. Since new skills should be based in some way on existing ones, some kind of mechanism is required for dealing with a complex hierarchy of skills.

One of the fundamental difficulties facing any mechanism that accrues new capabilities is how to add new pieces to the existing ones. One method is to add new pieces when necessary, and then, maybe later, if resources get tight, find a different method for pruning away less useful pieces. While it

is usually easy enough to add new pieces, it can be perplexing to figure out which ones should be removed. (This was one of the principal problems faced by CHILd [1]). Once such a method is found, the system will eventually need to balance the parts added by deleting other parts, eventually reaching a kind of dynamic equilibrium such that system resources are utilized effectively.

We approach this differently. Rather than devise one method for building up and another for pruning, we instead address the resource-constraint problem at the outset as one of the core principles: there should always be exactly as many modules as the hardware can support. The system must figure out how to allocate resources as an essential component of its underlying methodology.

This paper presents a new approach for dealing with the problem and explores a new way of thinking about the hierarchical organization of behavior for continual learning. We describe a fairly general system and discuss its advantages and disadvantages. But it is not practical to give an explicit demonstration of all aspects of the system in the scope of this paper. Thus, we will discuss the system in broad terms and then choose one specific instantiation for illustration in Sections IV and V.

II. THE MOT SYSTEM

We envision a fixed-sized pool of generic modules that can be plugged into a larger architecture arbitrarily and can find a role to play there. The modules must learn to combine their individually limited powers to address the needs of the agent, spreading out to cover the most critical regions of behavior space. If there are more modules than necessary, they should redundantly be drawn to the same areas, which should increase robustness. In the more usual case, when there are too few modules, they should distribute themselves in such a way as to cover the most important areas best.

Each module is called a *mot*¹ and each mot represents a motor behavior. The mot system is a framework for investigating reinforcement-learning agents composed of a very large number (ideally thousands or millions) of learning modules, though here we will focus on one specific instantiation of the framework.

The name “mot” recognizes several aspects of the system. First, each mot represents only a very tiny piece of the agent’s

¹Pronounced “mōt” or “moot” and rhymes with “boat” as in “motor boat”.

overall behavior and is in principle individually insignificant, thus recalling the English homophone “mote”. Second, each mot represents a component of motor behavior, thus the name “mot” alludes to the mot’s role in the motor system. Third, in combination, the mots represent the agent’s entire collection of skills, its entire motor vocabulary, where each mot is like a word in that vocabulary. Just as words can be combined to produce a nearly limitless variety of meaning, the mots should combine in the same way to produce the agent’s broad repertoire of skills. Each mot individually is something like a basis function within that repertoire. Thus, “mot” recalls the French word for “word” and emphasizes the combination of different components to produce skills in contrast to a hierarchical ordering of functions, where higher-level ones represent compositions or sequences of lower-level ones. Just as new words tend to represent new ideas rather than fixed sequences of existing words, mots do not represent ever-growing sequences of simpler actions, but critical components of common behaviors.

III. THE MOTMAP AND THE FUNCTIONAL ORGANIZATION OF THE MOTOR CORTEX

It seems likely that the motor cortex is not laid out as the homunculus that we learned about in our psychology and neurophysiology classes. Recent research reveals strong evidence that this part of the brain might instead be best understood as a topological map organized according to behavior, where similar behaviors occur close together and very different behaviors lie far apart [3], [4]. The key idea is that behaviors vary more-or-less smoothly from one region to another, such that as the distance between two regions decreases, the more likely they are to represent very similar behaviors. Inspired by this research, we have come to appreciate that this simple regularity might provide an excellent method for organizing large numbers of modules in a number of advantageous ways, even without reproducing all the details of the motor cortex or even requiring biological plausibility.

As envisioned, each mot thus has two aspects: its functionality and its location. Its functionality, described in detail later, refers to how the mot computes its estimate of the value function. The mot’s location is its position in a two-dimensional map, called the “motmap.” The motmap is a low-dimensional representation of the mots, organized by behavioral similarity. Organizing behavior by similarity provides multiple advantages for continual learning, both computational and functional. The most obvious of these advantages are: smoothness, robustness, automatic emergence of hierarchy, dimensionality reduction, and efficient distribution and reuse of mots (which is probably not a complete list).

Smoothness. Assuming that the map can find a smooth low-dimensional organization of policies, it must also have found critical similarities and—most important—gradients among the policies. Our eventual hope is that the best policy for a novel situation can be found using a hill-climbing search in the map.

Robustness. If each mot’s overall behavior is quite similar to that of its neighbors, then the actions suggested by a mot and its neighbors is also usually similar. If the granularity is fine enough, then each mot should closely resemble the average of any mots in the region enclosing it, which can be useful in the following ways. First, should a mot stop functioning for some reason, its neighbors can provide behavior nearly as good. Therefore, instead of ever relying on any single mot’s recommendation, the agent could in general combine mots within a region and use their average or weighted recommendations. (This method for choosing actions is discussed in the future-work section.) In effect, mots can be chosen according to vicinity, and their combined recommendations can decide the next action. Secondly, learning can also be done by vicinity rather than by individual mot. The agent presented here updates all the mots within a certain radius surrounding the winning mot. This is the primary force responsible for bringing about smooth variation between regions.

Hierarchy by region. If a small region is selected, the values of the mots can be combined (by average or weighted average) to produce a final action. The radius of the region can then determine the specificity of the action. If a large region is selected, then a very typical, generic behavior is produced. As the radius narrows, the agent’s responses become more refined and suited to the situation.

Take as an example the behavior of grasping, and imagine a library of different policies that handle different kinds of grasps for different kinds of objects: heavy, light, wide, narrow, delicate, rigid, soft, those with a handle and those without, etc. This library could be embedded in a large region, where the region as a whole might produce a generic arm movement and hand closure in the vicinity of an object. A smaller region might perform a grasp of a cup or glass, or a particular kind of ball or piece of clothing, etc. An even more focused region might produce a grasp dedicated to a specific type of glass, an empty wine glass, for example. Thus, hierarchy of behavior in this sense is about generality vs. specificity rather than about sequential combinations of policies. It emerges as a consequence of the limited ability of any individual module to deal with the entire variety of conditions that occur when trying to grasp objects. Two mots may agree about nearly everything, except in a few different cases. Thus, their policies are almost the same, but in certain situations, one is the expert, and in other cases, the other is expert, and in these cases they are different from each other. Therefore, in the general case, behaviors are not stored in just one mot in just one place; narrower regions store the exceptions to average behaviors stored across broader regions.

Notice that the traditional sense of hierarchy in RL is based on the notion of breaking up long sequences into shorter sequences defined by subgoals [5]–[8]. (CHILD is a major exception here [1]). But a policy itself in principle already provides a mechanism for chaining together a sequence of actions in response to a sequence of changing inputs.

Dimensionality reduction. In general, the biggest obstacle

in machine learning is the curse of dimensionality. Reducing the dimensionality of inputs improves learning speed by weeding out extraneous factors that might cause interference. From a computational perspective, algorithms with greater than linear complexity in the number of input parameters quickly become impractical as the dimensionality increases, but keeping the dimensionality low may allow more computationally demanding learning algorithms.

If specific behaviors occupy specific regions, then only the information needed by those behaviors has to be delivered to those regions. Sensations from the agent’s earlobes need not be transmitted to the mots that control toe wiggling. It may therefore be possible to drastically reduce the input dimensionality in any given region of policies, which would be beneficial both in terms of computation speed and learning speed.

But using heuristics to weed out input dimensions can be risky: some vital information might be encoded in the removed dimensions that only later becomes evident. The mot approach, however, keeps all the information available accessible, even if some of dimensions are removed from some regions. The smooth-gradient and neighborhood relationships in the map mean that inputs delivered to only a small region can expand their distribution in meaningful directions—outward to nearby regions; thus, individual inputs can follow these gradients to find policies where the information is useful.

Consider for a moment an individual input dimension. Most likely, it will not be spread out across the entire map, needed in every region, nor will it likely be delivered randomly to various places in the map. One would expect that each input dimension should become organized into various regions where that information is necessary. Regions that require the input effectively make it available to nearby regions. If the nearby regions pick it up, the information can spread to more and more distant regions, always being picked up by neighbors with similar policies.

As a result, it is potentially safe to eliminate dimensions from a subregion of the map (provided they are still available elsewhere) because those inputs have a way to return if necessary.

Reuse of mots. Regions of policy space that are critical to the agent should become represented by a large number of mots, each slightly different from the others. But what happens when a behavior is no longer useful? Perhaps it has been replaced by another skill (e.g., the agent learns to walk and no longer needs to crawl), or perhaps the environment has changed, and old skills are no longer needed (e.g., the agent loses its job and has to get a new one at a different company).

In this case, the mots that were devoted to the obsolete skills can be used for something else. At first, the obsolete mots closest to the boundaries of existing non-obsolete skills can be recruited to assist their neighbors. Regions with an overabundance of mots can share them with their neighbors, and so on, so that the mots from the unused region will eventually be recruited for parts of the state space where more representation is necessary. Thus, redundancy, which is a

critical component of the system and is necessary for the mots to have similarity relationships, is organized by the motmap to cover the state space effectively.

State aggregation by policy similarity. With learning, each area in the motmap comes to represent the behavior or policy that is activated in that area. But the policy is simply a mapping from states to actions, where the states are only those in which this region specializes. In principle, just as with observation dimensions, all extraneous state information can therefore also be removed. Thus, each region r specializes in a set of states, S_r , and maps these states to actions, $S_r \rightarrow A$ according to some set of similar policies π_r , where S_r is defined as that set for which the policies π_r are roughly appropriate.

Continual Learning. There is a fundamental dilemma faced by every real-world continual-learning agent. This dilemma is: what should I give up? Purely theoretical solutions addressing the continual-learning problem, such as AIXI [9], sidestep this issue by assuming away all resource limitations, which equally assumes away the possibility of a real-world implementation. Every practical system must face the tradeoff between discarding possibly useful saved information and failing to adapt to important new information. The continual-learning problem is unique in this respect because it does not assume a limit to learning, nor, therefore, a bound on the resources required to achieve optimality.

Thus, every real-world system must eventually contend with saturation of resources. Once saturation occurs, a heuristic of some kind is required to address the resulting resource-allocation problem. The motmap helps implement that heuristic by representing the entire state space of interest to the agent as a single two-dimensional sheet, where the resource (the mots) can both cooperate and compete to best cover that space to meet the needs of the agent.

IV. SYSTEM DESCRIPTION

In the standard reinforcement-learning framework (see Sutton and Barto, 1998), a learning agent interacts with a Markov Decision Process (MDP) over a series of time steps $t \in \{0, 1, 2, \dots\}$. At each time step the agent takes an action $a_t \in \mathcal{A}$ from its current state $s_t \in \mathcal{S}$. As a result of the action the agent transitions to a state $s_{t+1} \in \mathcal{S}$ and a reward $r_t \in \mathbb{R}$ is received. The dynamics underlying the environment are described by the state-to-state transition probabilities $\mathcal{P}_{ss'}^a = Pr\{s_{t+1}=s' \mid s_t=s, a_t=a\}$ and expected rewards $\mathcal{R}_{ss'}^a = \mathbb{E}\{r_{t+1} \mid s_t=s, a_t=a, s_{t+1}=s'\}$. The agent’s decision-making process is described by a policy, $\pi(s, a) = Pr\{a_t=a \mid s_t=s\}$, which the agent refines through repeated interaction with the environment so as to maximize $Q(s_0, a_0) = \mathbb{E}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t=s_0, a_t=a_0\}$, the total discounted reward it can expect to receive by taking action a_0 in state s_0 and following policy π thereafter.

The learning method used for the mot system described here is based on SERL [11], a multi-modular RL method that breaks down tasks based on the limited capacity of each module. SERL is useful for learning in the motmap because

it is an online, incremental method that autonomously assigns different parts of the tasks to different modules, requiring no intervention or prior knowledge. For clarity, a brief description of SERL is provided here, but the motmap and the two-dimensional organization of behavior does not in general depend on the SERL learning mechanism.

Informally, each SERL mot finds a part of the state space where it can approximate the value function, leaving other parts to other mots. With each new observation, the mot with the highest estimate of value is chosen to recommend an action. The mots can coordinate with each other because each one also estimates its error and reduces its value-function estimate by the estimated error. The chosen mot is the one with the highest value after reduction by the estimated error.

More formally, SERL consists of a set of modules, \mathcal{M} . For the purposes of this paper, SERL's input is provided as an observed feature vector $\mathbf{o} \in \mathcal{O}$, which uniquely identifies the state. Each module $i \in \mathcal{M}$ contains two components: a controller function,

$$f^{c,i} : \mathcal{O} \rightarrow \mathbb{R}^{|\mathcal{A}|},$$

which generates a vector of action-value estimates; and a predictor function,

$$f^{p,i} : \mathcal{O} \rightarrow \mathbb{R}^{|\mathcal{A}|},$$

which generates a vector of predicted action-value errors. At every time step, each module generates values based on the current observation vector, \mathbf{o}_t :

$$\begin{aligned} \mathbf{q}_t^i &= f^{c,i}(\mathbf{o}_t) \\ \mathbf{p}_t^i &= f^{p,i}(\mathbf{o}_t) \end{aligned}$$

These are combined for each module to create an $|\mathcal{M}| \times |\mathcal{A}|$ matrix L_t of *lower confidence* values such that

$$L_t^i = \mathbf{q}_t^i - |\mathbf{p}_t^i|,$$

where L_t^i is the i^{th} row of L_t .

Winner. At every time step there is a winning module, w_t . In most cases the winner is a module whose highest L value matches L_t^* , the highest value in L_t . But this rule is modified in an ε -greedy fashion [10] to allow occasional random selection of winners, based on a random value, $x_t \sim U(0, 1)$:

$$\begin{aligned} W &= \{i \in \mathcal{M} : \max_a L_t^{ia} = L_t^*\} \\ Pr\{w_t = i | L_t^i\} &= \begin{cases} \frac{1}{|\mathcal{M}|} & \text{if } x_t < \varepsilon \\ \frac{1}{|W|} & \text{if } x_t \geq \varepsilon \text{ and } i \in W \\ 0 & \text{otherwise,} \end{cases} \end{aligned}$$

where L_t^{ia} is the value for action a in L_t^i . SERL then calculates an ε -greedy policy based on the winner's L values: $L_t^{w_t}$. (In principle, the two ε values for action and module selection can be different but were the same for the illustration given in Section V. Both appear to be necessary for the system to function as desired.)

Learning. The function approximators are updated with targets generated by TD-learning [12]. Unlike plain SERL,

which updates the controller for the winner only, the learning algorithm for the mot system updates the controllers for a whole a set of mots, w_t^+ at every time step. The update itself is done using Q -learning [13]; thus for each $i \in w_t^+$ the target for $\mathbf{q}_t^{ia_t}$ (the component of \mathbf{q}_t^i corresponding to action a_t) was: $r_t + \gamma L_{t+1}^*$.

Every mot's predictor is updated at every step; the target is the magnitude of the mot's TD error:

$$\delta_t^i = r_t + \gamma L_{t+1}^* - \mathbf{q}_t^{ia_t}.$$

For the illustration section below, the predictors and controllers are all linear function approximators. (SERL is capable of learning non-linearly separable classifications even when its controllers and predictors are only linear function approximators [11].)

Learning in the Motmap. The mot system coordinates a large number of individual mots by way of *selection mechanisms*, which choose certain mots from among the rest. In particular, certain mots are selected to be the winner(s), to be updated by learning, or to be moved in the motmap.

We have considered two ways to organize the motmap so as to accomplish the goals set out in Section III. One is to move the components around the map according to a similarity-distance metric such that the similarity in their behaviors is reflected (as well as possible) in their two-dimensional projection onto the map. In this case, each mot actively seeks out regions whose average behavior most closely resembles that of the mot itself. While promising and worthy of substantial elaboration elsewhere, it is not what we focus on here. Instead we focus on a simpler method that in many ways resembles the self-organizing maps (SOMs) of Kohonen [14] (though there are many and substantial differences). In this case, the components always remain in the same places, the topology does not change; instead, each component gradually changes what it represents so as to preserve similarity between neighboring units as much as possible.

The motmap described here is laid out in a fixed, two-dimensional grid.² Each mot has a specific fixed coordinate in the grid (look ahead to Figure 3 below), and the distances between the coordinates of different mots are easily calculated, thus allowing selection within a given radius of any location in the map.

V. ILLUSTRATION

While we have explored the mot framework in many different configurations, we give a detailed illustration of just one of these here. The intention of this illustration is to convey a proof of concept: that learning can be done in the system and that a map emerges having local similarity relationships, thus in principle providing the foundation for the benefits described

²We have chosen two dimensions for three primary reasons: (1) it is easy to visualize; (2) it provides for the most drastic useful reduction in dimensionality, and (3) it seems to work quite well in the brain—the cortex, though it consists of multiple layers, can for many purposes be meaningfully approximated as a two-dimensional surface. However, we cannot say with certainty that two dimensions are necessarily optimal.

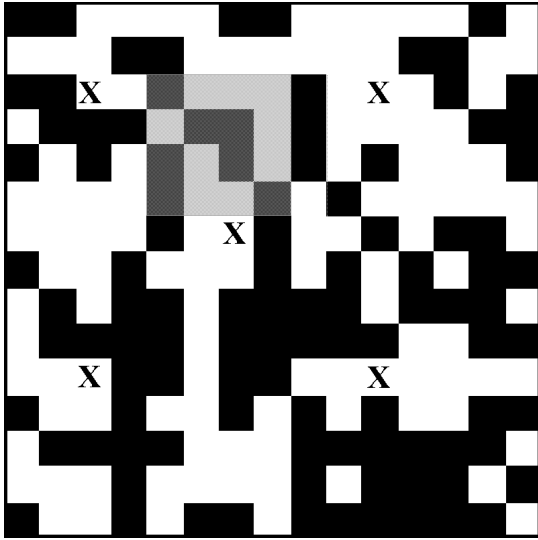


Fig. 1. The moving-eye task. Through its “eye” (shown in gray) the agent observes a 4×4 (16 bit) piece of the underlying black-and-white image. The agent can move the eye one black or white square to the North, South, East or West. It receives a reward of +1 when the upper-left corner of its “eye” moves over an “X”, after which the eye “saccades” to a random position.

in Section III (i.e., to avoid possible confusion, it is not the intention of this illustration to demonstrate better performance than other systems on a reinforcement-learning testbed).

Functionally, the mots implement the SERL algorithm as described above. In the motmap, mots are laid out in a fixed two-dimensional grid and their positions do not change over time. A winner is chosen as in the SERL algorithm, and the controller of the winner is updated together with the controllers of all the mots within a neighborhood of radius 0.1 surrounding the winner (where each side of the motmap has a length of 1.0). This configuration allows illustration of many, but not all, of the issues discussed above.

The illustration is shown using an artificial-eye task (the same one used as a demonstration of SERL [11]), which allows easy visualization of the task and the results (Figure 1). The agent’s observation is determined by the location of its “eye” (a 4×4 square—shaded gray) above a 15×15 random black and white bitmap. Each pixel in the bitmap represents a value of 1 (black) or 0 (white) within a 16-dimensional observation vector, which is always expanded with a bias feature of 1. The image was generated randomly but checked to ensure that every 4×4 square is unique. In this task, the agent has four actions that move its eye one pixel North, East, West, or South. If there is no room to move in the chosen direction, the eye remains in place. There are 5 rewarding squares (shown as an X) which, when reached with the upper left-hand corner of the eye, provide a reward of +1 and force the eye to saccade to a random position in the image. All other actions result in a reward of 0. Thus, to achieve best performance, the agent must learn which reward is closest and move toward that closest one. For this task $\gamma = 0.9$, $\alpha = 0.05$, and $\varepsilon = 0.02$.

The agent begins training with zero ability to solve the

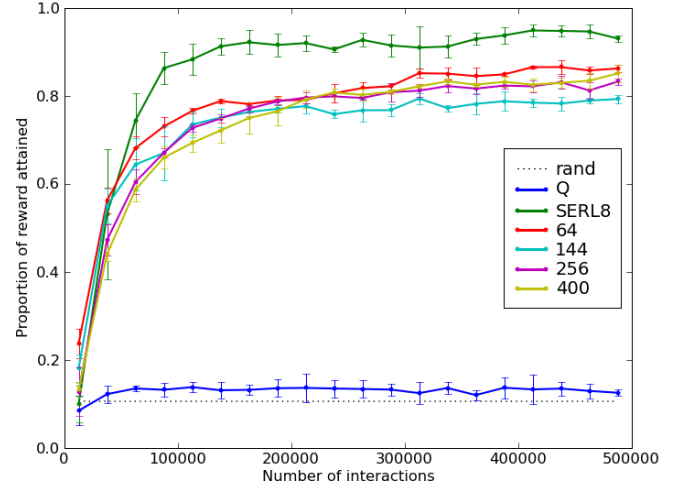


Fig. 2. With experience, the agent learns to increase its reward on the artificial-eye task. The vertical axis shows reward received, normalized between 0 and 1 (optimal policy), while the horizontal axis shows the number of actions taken. Motmaps are shown with 64, 144, 256 and 400 mots. Plain SERL with eight modules is shown for comparison. The blue line (“Q”) displays the results for a simple Q -learning agent using a linear function approximator and the same parameters as with the modular system. The dotted line shows the baseline performance of the uniform random policy.

task, but over time it achieves increasingly greater reward as shown in Figure 2. All results are shown in proportion to the maximum achievable reward for this task, which is represented by a value of 1 on the vertical axis. The horizontal axis shows the total number of steps taken. (For comparison, learning with a simple Q -learner using a linear function approximator (and the same parameters) did not achieve high reward.)

The purpose of this example is twofold. First, it shows that the agent can learn non-trivial tasks. (The task exhibits non-linearly separable decision boundaries, which cannot be drawn by a single linear function approximator, and therefore requires the cooperation of multiple mots).

Second, it allows demonstration of the formation of the motmap. Note that in this illustration, formation of the motmap comes at a slight cost, as training speed and performance for the motmap are not as good as for SERL alone, which is understandable, given that SERL alone has none of the constraints (or benefits) of the motmap.

Figure 3 shows a plain motmap of 400 mots arranged in a 20×20 fixed two-dimensional grid. Each mot is represented by a dot, where the red dots indicate the mots whose controllers are to be updated next. The center red dot is the winning mot.

After training on the task for 500,000 steps, the mots have become organized by similarity on the motmap. Figure 4 shows a composite view of all the mots’ error predictors. The figure consists of 144 squares corresponding to the 144 possible positions of the eye over the image, and thus 144 possible agent observations. Each square shows a snapshot of the motmap for the given observation. Every pixel in the image represents a single mot’s error prediction for that observation.

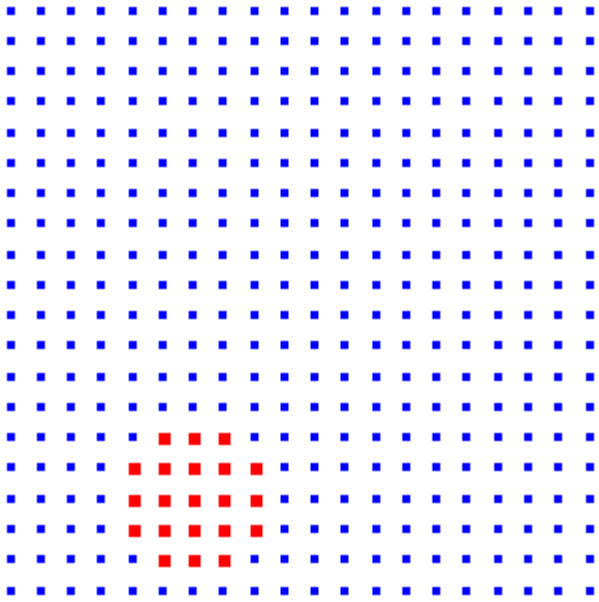


Fig. 3. The motmap, where each mot's location on the map is shown as a blue dot, except for those mots chosen for updating, which are shown in red.

Darker values indicate lower predicted error. The values are shown at different gray levels, on a logarithmic scale such that the darkest spots mean that the predicted error is 10^{-3} or smaller, and the white pixels mean that the predicted error was 1 or larger. The entire figure pertains to only a single action; figures for the other three actions are not shown, but are similar in appearance.

Figure 5 shows a progression of the motmap for a particular set of subimages and a particular action as the motmap evolves over time.

VI. DISCUSSION AND FUTURE DIRECTIONS

The figures reveal that after training, the map has become organized such that similar values tend to be generated by nearby mots in a fairly smooth fashion. Thus, the system is able to learn *and* to organize the behaviors it has learned by similarity. Furthermore, the progression from the beginning of training to the end showed an evolving map, with different parts becoming specialists at different times.

Continuous Selection Mechanisms. The mot system illustrated above chooses actions by selecting single mots according to the SERL algorithm, but it could in principle produce actions by combining the outputs from all the mots within a region, following the assumption that nearby mots produce similar behaviors. In that case, other methods for choosing actions may be useful, such as hill-climbing in the motmap itself, or by directly choosing regions in the map and representing action selection as the selection of a region. Furthermore, both the action and update selection mechanisms could themselves be smooth functions (e.g., Gaussians) rather than the binary mechanism used here; i.e., instead of updating a mot if and only if it falls within a radius of the winner, the

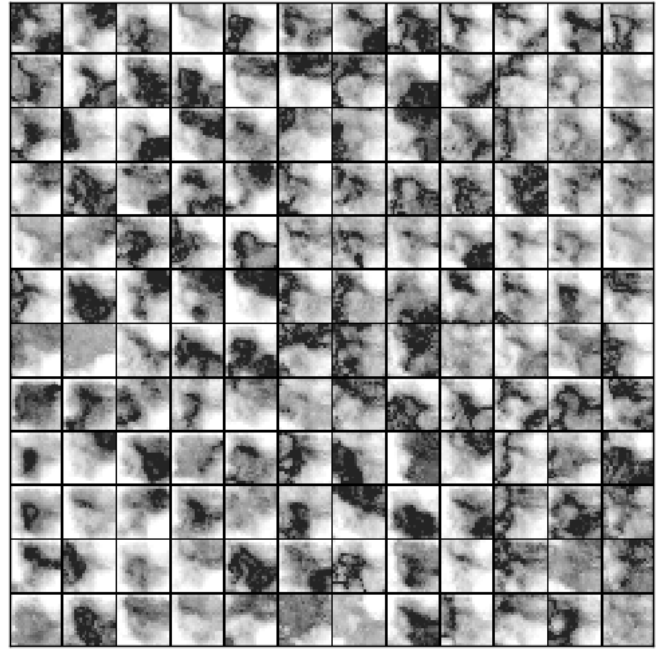


Fig. 4. The prediction values for the motmap, reproduced for all 144 possible observations of the task. Each square shows all 400 mots, one pixel each, for the observation corresponding to that square. Dark pixels represent very low estimated error. Light pixels represent high estimated error.

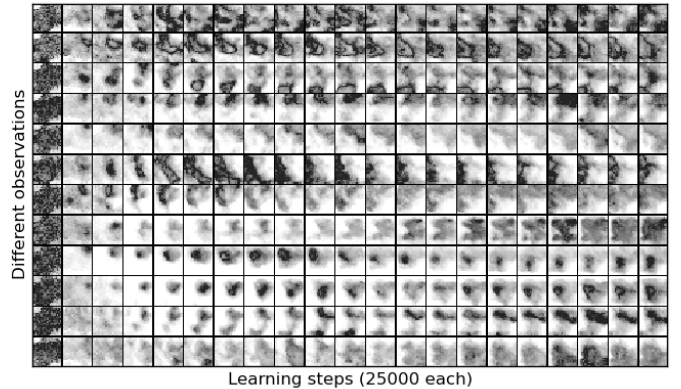


Fig. 5. Motmap prediction values for a subset of observations (rows) evolving over time (columns) during training on the task. One column is sampled every 10,000 time steps, starting at time step 10,000 for the first column. Each square shows all 400 mots, one pixel each, as in Figure 4. Note that the last column above corresponds to the first column of Figure 4.

learning rate of each mot could be proportional to the distance of the mot from the winner.

Heterogeneity. The loose framework above allows enormous flexibility while also providing a way that parts can be easily mixed and matched. Among these parts are the learning algorithms for the individual mots (for their controllers and predictors) which can be any online supervised learning algorithm. Specific kinds of learning environments may require specific kinds of learning algorithms with specific kinds of capabilities.

Most importantly, the framework allows heterogeneity: dif-

ferent mots can have different learning algorithms. This will allow specialization according to capability. Some simple function approximators might do very well in certain parts of the state space but may be too simple for others. More complex algorithms may be perfect for other parts of the state space, but learn too slowly in the simple places. We expect mot systems functioning in real-world domains to be heterogeneous.

Furthermore, while SERL was used as the multi-modular reinforcement-learning method in this paper, in general there is no strict dependence on this method. Other modular reinforcement-learning methods [5]–[8], [15]–[23] that break the task down in other ways on the basis of other constraints might also be adaptable to a two-dimensional map, yielding systems with different properties. Ideally, however, for maximum applicability such methods should be incremental, on-line, and non-episodic, breaking down the task autonomously and requiring no external intervention.

Temporal contiguity. It should be noted that the illustration above only examined single responses to single observations. Temporal contiguity of behavior is not demonstrated in this paper but should be understood to be a clearly plausible extension that can be accomplished in at least three ways. The first is to include state information (using any kind of recurrent neural network, for example, as a mot’s function approximator). Since mots within the same local region are likely to make similar decisions in most situations, features encoding useful state information (e.g., hidden-unit activations) could then be shared among nearby mots within a neighborhood, with the size of the neighborhood growing and/or shrinking over time (as described in Section III). If the motmap were physically implemented as a grid in actual hardware, these relationships could potentially be exploited to reduce communication costs. The second is to update regions based not just on spatial neighborhood relationships but also on temporal neighborhood relationships, thus encouraging regions of the motmap to encode extended trajectories of temporally contiguous behavior. As described above, the region of the map can become associated with certain states, and therefore location of an active area in the map also can provide some degree of state information. Finally, the motmap could in principle be combined with an external selection mechanism that chooses behaviors in the map by location and extent (coordinates and radius). This external mechanism might choose behaviors depending on its own record of the state, thus exhibiting sequential regularities.

Curiosity. The idea behind the mot system, of course, is to do continual learning: learning new skills on top of previous skills, but this is greatly benefited by a method for exploration that pushes the agent to learn things the agent does not yet know. Thus, it is reasonable to ask, how can the mot system be combined with mechanisms for curiosity [24], [25]? Curiosity could be pursued at either the global or mot level.

Goal and subgoals. The discussion and illustrations above did not specifically consider providing goal or subgoal information to the mots. Doing so, for example as an explicit part

of the observation, could allow the mots to further organize and differentiate with respect to this information. This could be useful for building a library of useful skills independent of the value function; for example, a collection of different grasps or target-reaching behaviors.

Scaling. The mot system allows the task to be split among a large collection of modules, organized by two-dimensional similarity relationships, but how will the potential number of stored behaviors scale with the size of the motmap? Much further work is needed to answer this question, even for the illustrative example given here; however, we can at least speculate, and note that complexity is handled with two separate mechanisms. First, each function approximator has its own VC-dimension, its own capacity limitation, but when multiple modules are combined together, the capacity of the system as a whole increases, roughly linearly with respect to the number of modules. Second, useful generalization is the primary mechanism whereby small numbers of mots can handle large numbers of specific cases. One of the primary motivations of the two-dimensional organization of behaviors is to promote useful generalization. (Most of the benefits of the motmap described in Section III are mechanisms for increasing useful generalization.) Verifying these hypotheses, though beyond the scope of this paper, will be revealing. Of particular interest will be the scaling behavior of the system with respect to the dimensionality of the observations. Near-term future work will also partially focus on applying the mot framework to actual robotics systems where specific examples more directly related to those of Section III (such as grasping) can be evaluated.

VII. CONCLUSIONS

This paper has introduced and described the first multi-modular, continual-learning system whose modules self-organize in a two-dimensional map according to behavioral similarity. We have described a broad and flexible framework (the “mot” system), lightly inspired by recent research on the motor cortex. We have discussed the advantages of the general approach, and have provided a proof-of-concept demonstration of a particular, important instance of the framework. This example system learned to achieve reward in a reinforcement-learning task while organizing a large number of modules (mots) in a two-dimensional topological map such that nearby mots represented similar behaviors. The paper opens up a new way of addressing the continual-learning problem, based on fixed-pool resource allocation and the two-dimensional, topological organization of behavior.

VIII. ACKNOWLEDGMENTS

We are grateful to Vincent Graziano, Tino Gomez, Leo Pape, and Tobias Glasmachers for helpful discussions in the long development of these ideas. This research was funded in part through the following grants: EU projects IM-Clever (231722) and NeuralDynamics (grant 270247), and SNF grant 200020-122124.

REFERENCES

- [1] M. B. Ring, "Continual learning in reinforcement environments," Ph.D. dissertation, University of Texas at Austin, Austin, Texas 78712, August 1994.
- [2] —, "CHILD: A first step towards continual learning," *Machine Learning*, vol. 28, pp. 77–104, 1997.
- [3] M. S. A. Graziano and T. N. Affalo, "Rethinking cortical organization: moving away from discrete areas arranged in hierarchies," *Neuroscientist*, vol. 13, no. 2, pp. 138–47, 2007.
- [4] M. Graziano, *The Intelligent Movement Machine: An Ethological Perspective on the Primate Motor System*. Oxford University Press, 2009.
- [5] M. Wiering and J. Schmidhuber, "HQ-learning," *Adaptive Behavior*, vol. 6, no. 2, pp. 219–246, 1998.
- [6] R. S. Sutton, D. Precup, and S. P. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.
- [7] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," *J. Artif. Intell. Res. (JAIR)*, vol. 13, pp. 227–303, 2000.
- [8] S. P. Singh, "Transfer of learning by composing solutions of elemental sequential tasks," *Machine Learning*, vol. 8, pp. 323–339, 1992.
- [9] M. Hutter, *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Berlin: Springer, 2004.
- [10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998. [Online]. Available: <http://www-anw.cs.umass.edu/rich/book/the-book.html>
- [11] M. Ring and T. Schaul, "Q-error as a selection mechanism in modular reinforcement-learning systems," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2011, to appear.
- [12] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.
- [13] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, May 1989.
- [14] T. Kohonen, *Self-Organization and Associative Memory*. Springer, second edition, 1988.
- [15] C. A. Rothkopf and D. H. Ballard, "Credit assignment in multiple goal embodied visuomotor behavior," *frontiers in Psychology*, vol. 1, no. 0, November 2010.
- [16] D. Caligiore, M. Mirolli, D. Parisi, and G. Baldassarre, "A bioinspired hierarchical reinforcement learning architecture for modeling learning of multiple skills with continuous states and actions," in *Proceedings of the Tenth International Conference on Epigenetic Robotics (EpiRob2010)*, ser. Lund University Cognitive Studies, B. Johansson, E. Sahin, and C. Balkenius, Eds., Sweden, November 5–7 2010.
- [17] J. Tenenbergs, J. Karlsson, and S. Whitehead, "Learning via task decomposition," in *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, J. A. Meyer, H. Roitblat, and S. Wilson, Eds. MIT Press, 1993, pp. 337–343.
- [18] P. Dayan and G. E. Hinton, "Feudal reinforcement learning," in *Advances in Neural Information Processing Systems 5*, C. L. Giles, S. J. Hanson, and J. D. Cowan, Eds. San Mateo, California: Morgan Kaufmann Publishers, 1993, pp. 271–278.
- [19] B. Bakker and J. Schmidhuber, "Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization," in *Proc. 8th Conference on Intelligent Autonomous Systems IAS-8*, F. G. et al., Ed. Amsterdam, NL: IOS Press, 2004, pp. 438–445.
- [20] T. Kohri, K. Matsubayashi, and M. Tokoro, "An adaptive architecture for modular q-learning," in *IJCAI (2)*, 1997, pp. 820–825.
- [21] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Networks*, vol. 11, no. 7-8, pp. 1317–1329, 1998.
- [22] K. Samejima, K. Doya, and M. Kawato, "Inter-module credit assignment in modular reinforcement learning," *Neural Networks*, vol. 16, no. 7, pp. 985–994, 2003.
- [23] K. Doya, K. Samejima, K. ichi Katagiri, and M. Kawato, "Multiple model-based reinforcement learning," *Neural Computation*, vol. 14, no. 6, pp. 1347–1369, 2002.
- [24] J. Schmidhuber, "Curious model-building control systems," in *Proceedings of the International Joint Conference on Neural Networks, Singapore*, vol. 2. IEEE press, 1991, pp. 1458–1463.
- [25] —, "Formal theory of creativity, fun, and intrinsic motivation (1990-2010)," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 3, pp. 230–247, 2010.