Learning cognitive maps for vicarious evaluation

Rajeev V. Rikhye, Nishad Gothoskar, J. Swaroop Guntupalli, Antoine Dedieu,
Miguel Lázaro-Gredilla, Dileep George
Vicarious AI

Abstract

Cognitive maps enable us to learn the layout of environments, encode and retrieve episodic memories, and navigate vicariously for mental evaluation of options. A unifying model of cognitive maps will need to explain how the maps can be learned scalably with sensory observations that are non-unique over multiple spatial locations (aliased), retrieved efficiently in the face of uncertainty, and form the fabric of efficient hierarchical planning. We propose learning higher-order graphs – structured in a specific way that allows efficient learning, hierarchy formation, and inference – as the general principle that connects these different desiderata. We show that these graphs can be learned efficiently from experienced sequences using a cloned Hidden Markov Model (CHMM), and uncertainty-aware planning can be achieved using message-passing inference. Using diverse experimental settings, we show that CHMMs can be used to explain the emergence of context-specific representations, formation of transferable structural knowledge, transitive inference, shortcut finding in novel spaces, remapping of place cells, and hierarchical planning. Structured higher-order graph learning and probabilistic inference might provide a simple unifying framework for understanding hippocampal function, and a pathway for relational abstractions in artificial intelligence.

Introduction

Vicarious trial and error [1], the ability to evaluate futures by mental time travel, is a hallmark of intelligence. To do this, animals and people need to learn mental models, such as "cognitive maps" [2] of physical or conceptual space from the stream of their experience which is often aliased: identical events could have different interpretations in different contexts, and dissimilar events could mean the same thing in some contexts [3]. A computational theory for cognitive maps should propose mechanisms for how context and location specific representations emerge from aliased sensory or cognitive events, and how the representational structure enables consolidation, knowledge transfer, and flexible and hierarchical planning. Previous attempts include modeling hippocampus as a memory index, a relational memory space, a rapid event memorizer, and systems-level models of pattern-separation and pattern completion, but these models have not reconciled the diverse functional attributes [4–6] of the hippocampus under a common framework. One recent model based on the concept of successor representation [7] attempted to capture representational

properties of place cells and grid cells [8]. Yet another recent model casts spatial and non-spatial problems as a connected graph with neural responses as efficient representations of this graph [9]. Unfortunately, both these models fail to explain several experimental observations such as the discovery of place cells that encode routes [10, 11], remapping in place cells [12], a recent discovery of place cells that do not encode goal value [13], and flexible planning after learning the environment.

Here, we propose that learning higher-order graphs of sequential events might be an underlying principle of cognitive maps, and propose a specific representational structure that aids in learning, memory integration, retrieval of episodes, and navigation. In particular, we demonstrate that this representational structure can be represented as a probabilistic sequence model – the cloned Hidden Markov Model (CHMM). We show that sequence learning in CHMMs can explain a variety of cognitive maps phenomena such as discovering spatial maps from random walks under aliased and disjoint sensory experiences, transferable structural knowledge, finding shortcuts, and hierarchical planning and physiological findings such as remapping of place cells, and route-specific encoding. Notably, all these properties emerge from a simple model that is easy to train, scale, and perform inference on.

Cloned Hidden Markov Model as a model of cognitive maps

CHMMs are based on Dynamic Markov Coding (DMC) [14], an idea for representing higher-order sequences by splitting, or cloning, observed states. For example, a first order Markov chain representing the sequences A-C-E and B-C-D will also assign high probability to the sequence A-C-D (Figure 1a). DMC makes a higher-order model by splitting the state C into multiple copies, one for each incoming connection, and then further specializes their outgoing connections through learning. The state cloning mechanism permits a sparse representation of higher-order dependencies, and the basic idea has been discovered in various domains [14–17]. With cloning, the same bottom-up sensory input is represented by a multitude of states that are copies of each other in their selectivity for the sensory input, but specialized for specific temporal contexts (Figure 1c). When a particular symbol is observed (symbol C in Figure 1c), all the different sequences that symbol participates in can be directly activated, and modulated based on their specific contextual support. This representation enables storing a large number of higher-order and probabilistic sequences without destructive interference, enables bridging between disjoint episodes of experience, and allows for the retrieval of related sequences in an efficient manner, all of which are functional requirements of cognitive maps. However, learning DMC is challenging because splitting of states relies on a greedy heuristic that results in severe sub-optimality.

Our previous work [18] showed that many of the training shortcomings of DMC can be overcome by representing it as a CHMM – a sparse restriction of an overcomplete HMM [19]. In the CHMM, many hidden states map deterministically to the same observed state (Figure 1b). The set of hidden states that map to a given observed state are referred to as the *clones* of that observed state. The probability of a sequence in a CHMM is:

$$P(x_1,\ldots,x_N) = \sum_{\{z_n \in \mathrm{hid}(x_n)\}_{n=1}^{N}} P(z_1) \prod_{n=1}^{N-1} P(z_{n+1}|z_n) \tag{1}$$

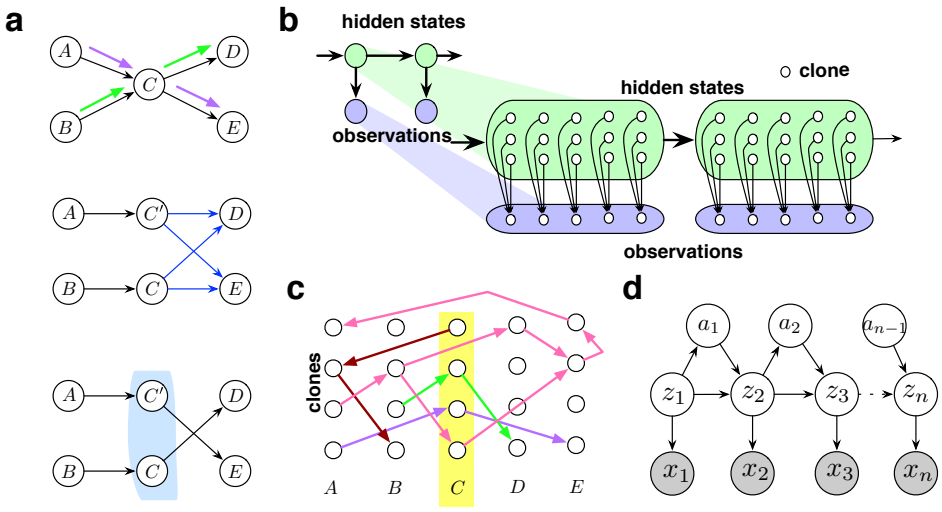


Figure 1: Cloned Hidden Markov Model (a) Dynamic Markov coding. A first order Markov chain modeling sequences A-C-E and B-C-D will also assign high probability to the sequence A-C-D. Higher-order information can be recovered by cloning the state C for different contexts. (b) The cloning structure of DMC can be represented in an HMM with structured emission matrix. (c) Cloning structure allows for storing large number of higher-order sequences with probabilistic branching, and allows for their quick retrieval. (d) Extension of CHMM to include actions.

where $z_n \in \text{hid}(x_n)$ means that the summation is only over the hidden values of z_n that emit x_n (the clones of x_n). There is a fixed, known association between hidden states and emissions, with each hidden state being associated to a single emission. This sparsity structure alleviates the credit diffusion problem typically associated with HMMs, and makes CHMMs amenable to training with a variant of the expectation-maximization (EM) algorithm (see **Methods**).

CHMMs can be extended to include actions of an agent. Consider an agent that moves in a room and experiences a (possibly non-unique) sensory observation at each location. The observed stream is (x_1, a_1), $(x_2, a_2) \ldots (x_{N-1}, a_{N-1}), (x_N, -)$ where $x_n \in \mathbb{Z}^*$ are the agent's sensory experiences and $a_n \in \mathbb{Z}^*$ are the actions reported by the agent's proprioception. The observed actions are simply non-negative integers with unknown semantics (i.e., the agent observes $a_1 = 0$ happened, but does not know that the action means "move north in the room"). The simplest way to augment a CHMM with actions is conditioning the state transition on the action. However, such a model could only predict future observations given actions, and would be clueless as to which actions are feasible when in a given state. Instead, we propose a generative action-augmented CHMM in which the action is a function of the current hidden state and the future hidden state is a function of both the current hidden state and the action taken, which makes intuitive sense given our understanding of actions and observations. The graphical model is depicted in Figure 1d. Mathematically, the joint observation-action density is

$$P(x_1, \ldots, x_N, a_1, \ldots, a_{N-1}) = \sum_{\{z_n \in \text{hid}(x_n)\}_{n=1}^{N}} P(z_1) \prod_{n=1}^{N-1} P(z_{n+1}, a_n | z_n). \tag{2}$$

A trained CHMM can be used for planning to attain goals by performing inference on it using message-

passing algorithms [20]. The goal can be specified as a desired observation or as a specific clone of that observation. Once the goal is specified, planning can be accomplished by clamping the current clone at the current time-step, the target clone or observation at a future time step and inferring the intermediate sequence of observations, and, in the action-augmented CHMM, the required actions. It is easy to determine how far into the future we have to set our goal by running a forward pass and determining the feasibility of the goal at each step. Then the backward pass will return the required sequence of actions.

Because the graphical model is inherently probabilistic, handling of uncertainty is built in. This enables it to deal with actions with uncertain outcomes and with noisy observation data.

Results

We performed several experiments to test the ability of CHMMs to model cognitive maps. We specifically tested for known functional characteristics such as learning spatial maps from random walks under aliased and disjoint sensory experiences, transferable structural knowledge, finding shortcuts, and supporting hierarchical planning and physiological findings such as remapping of place cells, and route-specific encoding.

Emergence of spatial maps from aliased sequential observations

From purely sequential random-walk observations that do not uniquely identify locations in space, CHMMs can learn the underlying spatial map, a capability that is similar to people and animals. Figure 2a shows a 2D room with the sensory observations associated with each location. The room has 48 unique locations, but only 4 unique sensory inputs (represented as colors), and an agent taking a random walk observes a sequence of these sensory inputs. A first-order sequence model would severely under-fit, and pure memorization of sequences will not learn the structure of the room because the same sequence hardly ever repeats. In contrast, a CHMM discovered the underlying 2D graph of the room perfectly (Figure 2b). As the number of unique randomly placed observations increases, learning becomes easier (see Supplementary Results).

Remarkably, CHMM can learn the spatial topology even when most of the observations are aliased like those from a large empty room where distinct observations are produced only near the walls as in Figure 2c. The combination of high correlation between observations, and severe aliasing makes this a challenging learning problem. Despite this, the CHMM is able to perfectly learn the topology of the 6x8 room (Figure 2d). Of course, this capability degrades as the room gets larger, but the degradation is graceful. For example, the periphery of a 9×11 room is well modeled, but the CHMM is unable to distinguish a few locations in the middle (see Supplementary Results).

Transitive inference: disjoint experiences can be stitched together into a coherent whole

Transitive inference, the ability to infer the relationships between items or events that were not experienced at the same time, is attributed to cognitive maps [5]. Examples include realizing $A > C$ from knowing $A > B$

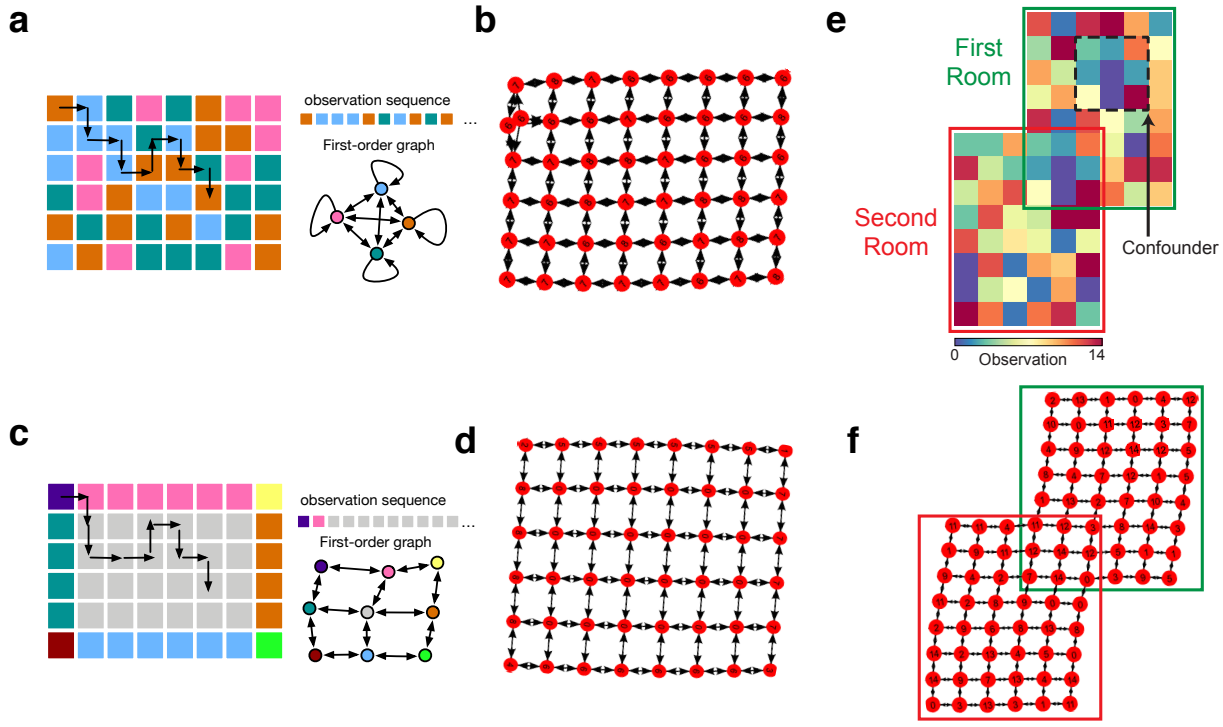


Figure 2: Spatial representations emerge from aliased sequential random-walk observations. (a) A room with only four unique observations. An agent taking a random walk in this room will observe a sequence consisting of 4 different symbols (colors), which is severely aliased as reflected in the first order Markov chain. (b) Graph learned by CHMM on observation sequences from the room in a. (c) A room with a uniform interior produces observation sequences that are highly correlated in time (d) Graph learned by CHMM on observation sequences from the room in c. (e) An agent experiences two different, but overlapping rooms in disjoint episodes of sequential experience. (f) CHMM can perform transitive inference and stitch together the disjoint experience into a coherent global map.

and $B > C$, or inferring a new way to navigate a city from landmarks and their relative positions experienced on different trips [21].

We tested CHMMs on a challenging problem designed to probe multiple aspects of transitive inference and found that it can stitch together disjoint episodes of sequential experience into a coherent whole. The experimental setting consisted of overlapping rooms (Figure 2e), each with aliased observations like in the previous experiment. Moreover, the first room had an additional portion which was identical to the overlapping section between the two rooms. This design allows to test whether an agent that experiences only first room or second room exclusively and sequentially, can correctly figure out the relationship between the rooms and their overlaps. The combination of a large state-space, aliased observations, and nested relationships make the problem setting significantly harder than previous attempts [22]. We collected two independent sequences of action-observation pairs on each room by performing two separate random walks, and trained a single CHMM on both sequences. The result of training is visualized in Figure 2f. The learned

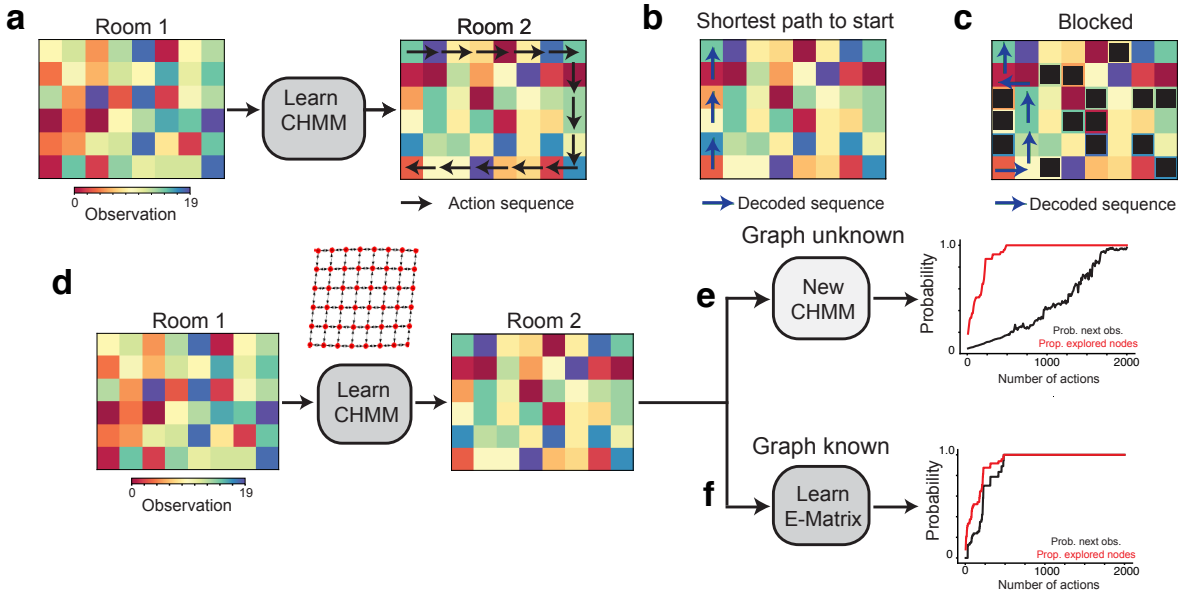


Figure 3: Learned spatial maps form a reusable structure to explore similar environments. (a-c) A CHMM trained on one room (a) and partial observations in a second, previously unseen room, utilizes the learned structure of the room to rapidly find both the shortest path to the origin (b) and navigate around obstacles (c). (d-f) The transition matrix (graph) learned in one room can be used as a re-usable structure to quickly learn a new room with similar layout but different observations.

transition matrix (shown as a graph) has stitched together the compatible region of both rooms, creating a single, larger spatial map that is consistent with both sequences while reusing clones when possible. The confounding additional patch in the first room remains correctly unmerged, and in the right relative position in the first room, despite looking identical to the overlapping region.

Discovering the correct latent global map enables CHMM to make transitive generalizations. Although the agent has never experienced a path taking it from regions that are exclusive to Room 1 to regions exclusively to Room 2, it can use the learned map to vicariously navigate between any two positions in the combined space. Just like in the earlier experiment, the learning is purely relational: no assumptions about Euclidean geometry or 2D or 3D maps are made in the model.

Learned graphs form a reusable structure to explore similar environments

The generic spatial structure learned in one room can be utilized for exploring, planning, and finding shortcuts in a novel room, much like the capabilities of hippocampus based navigation [23]. To test this, we first trained the CHMM on Room 1 based on aliased observations from a random walk. As before, CHMM learned the graph of the room perfectly. Next, we placed the agent in Room 2 which is unfamiliar (Figure 3a). We kept the transition matrix of the CHMM fixed and re-initialized the emission matrix to random values. As the agent walks in the new room, the emission matrix is updated with the EM algorithm. Even without visiting all the locations in the new room, the CHMM is able to make shortcut travels between

visited locations through locations that have never been visited (Figure 3b). After a short traversal along the periphery as shown in Figure 3a, we queried to find the shortest path from the end state to the start state. The CHMM returned the correct sequence of actions, even though it obviously cannot predict the observations along the path. Interestingly, Viterbi decoding [24] reveals the same hidden states that you would get if you Viterbi decoded the same path in Room 1. Querying the CHMM on the shortest path from the bottom left corner of the room to the start position, reveals the path indicated by the blue arrows in Figure 3b. This solution is the Djikstra's shortest path through the graph obtained from Room 1. Furthermore, if we "block" the path we get another solution that is also optimal in terms of Djikstra's algorithm (Figure 3c). Even with partial knowledge of a novel room, an agent can vicariously evaluate the number and types of actions to be taken to reach a destination by reusing CHMMs transition graph from a familiar room.

When the transition matrix from the old room is reused, the new room is learned very quickly even when the agent explores using a random walk: the new room is learned fully when all the locations in the room are visited at least once (Figure 3d-f). The plots show the proportion of the room explored and the average accuracy of predicting the next symbol as a function of the number of random-walk steps.

Representation of paths and temporal order

CHMMs learn paths and represent temporal order when the observed statistics demand it, for example when the observations correspond to an animal repeatedly traveling prototypical routes. In rats, "splitter cells" that represent paths rather than locations emerge when they repeatedly traverse the same sequential routes as opposed to random walks [26]. Figure 4a shows a maze in which the agent can traverse two different routes to reach the same destination. Both these routes have regions in which the exact path that the agent follows is stochastic, as denoted by the arrows that indicate the possible movements from each cell. As before, the same observation can be sensed in many parts of the maze. Additionally, the two routes intersect and share a common segment. CHMMs trained on these paths are able to represent both routes by using different clones for each of the routes, analogous to the route dependency exhibited by place cells in similar experiments. We observe that disjoint subsets of clones will activate when traversing each of the routes. Figure 4c shows that when conditioning on the starting state, sampling in the learned CHMM will always produce paths that are consistent with the two routes. By visualizing the graph defined by the CHMM transition matrix, we see that the two routes are represented with two different chains (Figure 4b). With a first-order model, when the shared segment is reached, all context about the previous segments will be lost and the model will make incorrect predictions about the future path. CHMMs, on the other hand, are able to capture the history of the path and therefore properly model the routes and their distinct start states.

Learning higher-order sequences can also explain observed phenomena like chunking cells and lap-counting cells found by [25]. Figure 4d shows a setting similar to the experiment in [25] where a rat ran multiple laps in a looping rectangular track and a reward was received at the end of the third lap. A CHMM that is exposed to the same sequence learns to distinguish between the different laps. The graph learned by the CHMM, shown in Figure 4d, has unwrapped the loop into 3 repeated sequences. Figure 4f shows that the activity of the clones corresponds to the lap counting cells and the chunking cells, since certain clones of

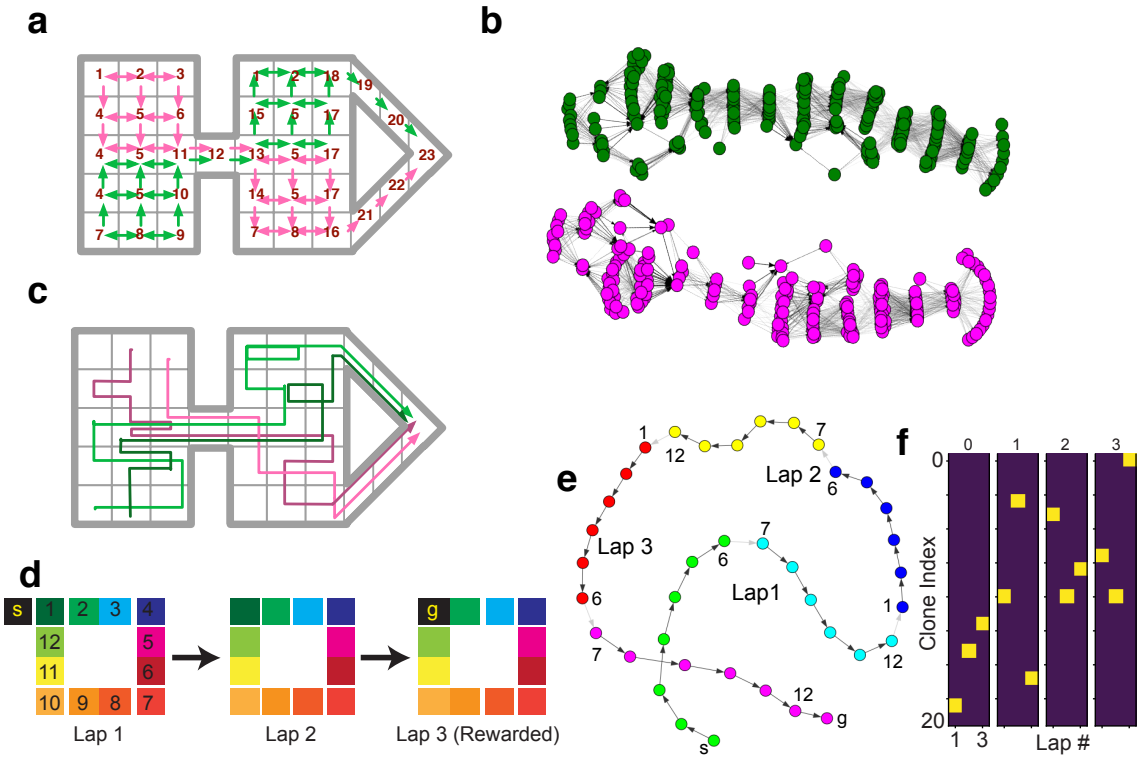


Figure 4: Structural generalization and path modeling. (a) A complex maze in which the agent takes two stochastic paths indicated in magenta and green. (b) Graph defined by the CHMM transition matrix shows 2 different chains are learned for the 2 routes (c) Paths sampled from the CHMM after it was trained on sequences from b. Although seven locations overlap between the two paths, CHMM learned different clones to represent the two paths. (d-f) A CHMM was trained on observations from three laps around a room (e). Community detection on the transition matrix again shows that each observation, although repeated on each lap, are represented by different clones (f). Thus, chunking cells reported by [25] in the hippocampus are easily explained by learning a CHMM on the sensory observations.

each observation will only activate on specific laps.

CHMMs ability to learn flexible higher-order sequences is independent of the modality. The inputs can correspond to spatial observations, odors, sequences of characters, or observations from any other phenomenon [18]. CHMM will learn an approximation of the graph underlying the generative process, in close correspondence with the role for cognitive maps envisaged by [2].

Retrieval and remapping

CHMM has the capacity to learn many disjoint maps from similar sequential observations, and then use contextual similarity to retrieve the appropriate maps, similar to global remapping phenomena observed in hippocampus. In Figure 5a, we show 5 different 10×10 mazes in which the agent can only observe the

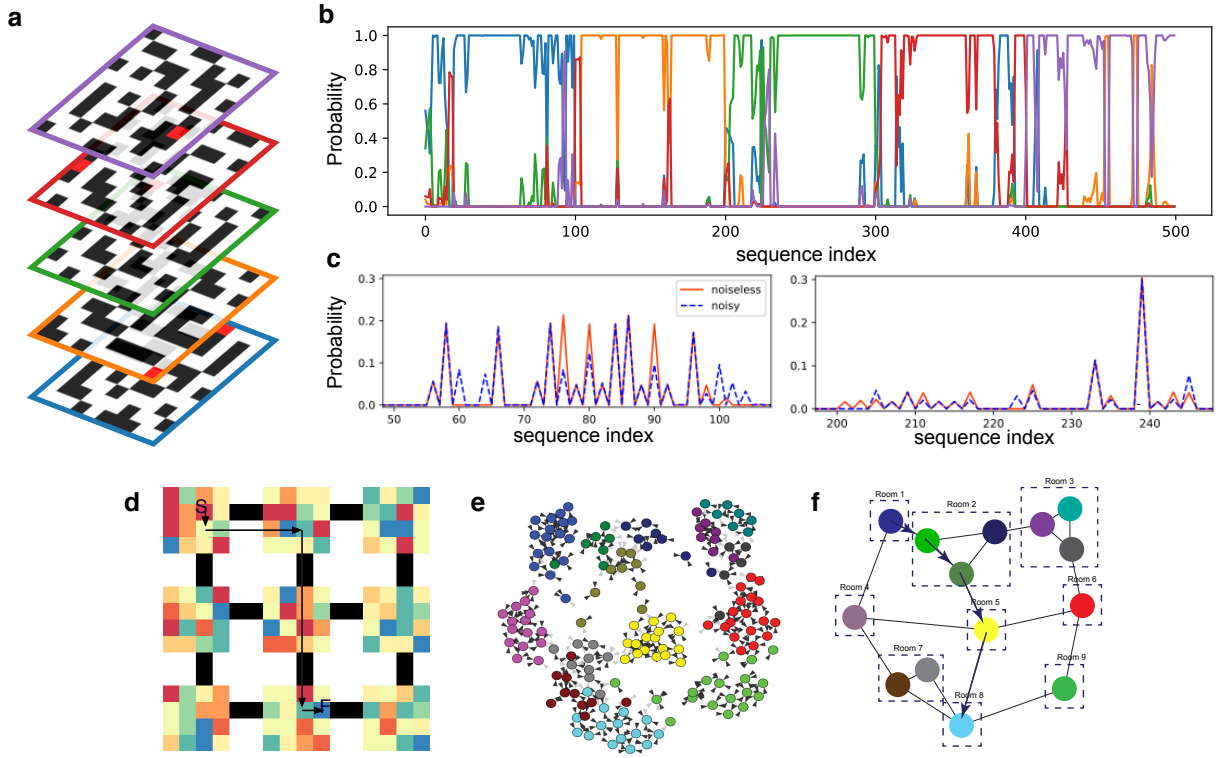


Figure 5: Remapping and hierarchical navigation. (a) Five different mazes used for training. Observations are not only aliased within a maze but also aliased with the other mazes. (b) Inferred probability of being in each of the 5 mazes, on observing consecutive 100-long test sequences from bottom maze to the top maze in that order. (c) Activations of a neuron when doing inference with noisy and noiseless observations. (d-f) A CHMM was learned by taking random walks in a maze (d) that was hierarchically organized. As before, this maze had sensory observations that were aliased both within rooms and across rooms. The black pixels denote "bridges" between the rooms. Once the maze was fully learned by the CHMM, community detection was performed (e) resulting in a clustering of the CHMM transition matrix into rooms (f). Planning a path (black arrows) between two rooms (denoted as (S)tart and (F)inish in (d)) was achieved by first planning which rooms to navigate and then finding the shortest path within each room.

occupancy of a 3×3 region surrounding their current location as they move around the maze. We learn a single CHMM from sequences of random walks in each of these mazes. Note that we do not provide any supervision about which maze the agent is in or if the agent has switched mazes. We then aggregate the activation of clones in each of the mazes to get a clone-to-maze mapping. After learning, when the agent begins exploring the mazes again (100 steps in each maze), we see that it activates the clones from the correct maze (Figure 5b), which allows it to make consistently correct predictions. It activates clones from the correct maze 92, 93, 92, 82, and 89 percent of the time, for each maze respectively. As discussed in [27], the CHMM demonstrates how inference of hidden states induces a remapping effect after context changes. In particular, it is because of this remapping that the CHMM can make accurate predictions in all mazes and

quickly recover from random switches between mazes.

CHMM can handle uncertainty and noise in observations and still navigate and plan properly. We simulate noise by corrupting 20% of observations and replacing them with random observations. The CHMM is quickly able to recover from corrupted symbols and begin making accurate predictions again. Further, through MAP inference given these uncertain observations, the CHMM can correct many of these errors. On a length 5000 sequence (1000 steps in 5 mazes), online MAP inference[1] corrected 493 of 996 corrupted symbols, while only corrupting 39 of the 4004 uncorrupted symbols. The native ability of CHMMs to deal with uncertainty can explain the rate remapping phenomena observed in biology. We compare the activation of clones on a noisy and noiseless version of the same sequence. Figure 5c show that the strength of the response diminishes when the observations are uncertain, similar to the rate remapping of place cells [12].

Community detection and hierarchical planning

CHMMs, combined with community detection, can learn hierarchical graphs that help with efficient planning [28]. If the sequential observations come from a process that has an underlying hierarchy, the graphs learned by the CHMM will reflect that hierarchical structure. We tested CHMMs for their ability to learn hierarchical graphs by simulating the movement of an agent in a complex maze as shown in Figure 5d. The maze has a total of nine rooms arranged as a 3×3 grid. Each room has aliased observations, and are connected by corridors. The aliasing is global: instantaneous observations do not identify the room, corridor, or location within a room. As in the earlier examples, training a CHMM on random walk sequences learned a perfect model of the maze. We then used community detection to cluster the transition matrix of the CHMM (Figure 5e). This clustering revealed a hierarchical grouping of the clones (Figure 5f), and a connectivity graph between the discovered communities. The communities respected room boundaries: although some rooms were split into two or three communities, no community straddled rooms. To navigate to a particular final destination F from a starting location S, first a path can be planned in the community graph between the source community and the destination community, and using that information to reduce the search-space in the original lower-level graph. This would be more efficient than planning directly in the original graph. We implemented this form of hierarchical planning and found that we were always able to recover the shortest path between randomly selected start and end positions.

It is important to note that hierarchical planning is significantly more efficient. A representative example of the reduction in complexity can be given as follows. Assume that we have V communities, E inter-community edges and M nodes inside each community. Further, assume that the nodes inside each community are fully connected, and between any two communities, there is at most one edge. Then with hierarchical planning the complexity of running Djikstra's algorithm is $O(E + V \log V + N_t(M(M-1)/2 + M \log M))$, where N_t is the number of top-level nodes to traverse in the second planning stage. In contrast, on the full graph, the complexity is $O(E + MV \log MV + V(M(M-1)/2))$. From these equations it is easy to see that hierarchical planning is more efficient because $N_t \leq V$ in all graphs.

[1] We are correcting symbols after observing them and prior to observing any future symbols, so the process is online.

Learning higher-order sequences that encode temporal contexts appropriately is crucial for the extraction of the hierarchy using community detection algorithms. Approaches that learn first-order connectivity on the observations, for example successor representations [7], will not be able to form the right representations because the observations are severely aliased. Although [6] showed the promise of community detection for discovering hierarchies, it relied on unrealistic assumptions about the nature of observed sequences.

Discussion

We pursued the strong hypothesis that hippocampus performs a singular algorithm that learns a sequential, relational, content-agnostic structure to store, abstract, and access sensory experience [29], and demonstrated evidence for its validity. Realizing this core idea required several interrelated advancements: (1) a learning mechanism to extract higher-order graphs from sequential observations, (2) a storage and representational structure that supports transitivity, (3) efficient context-sensitive and probabilistic retrieval, (4) and learning of hierarchies that support efficient planning – techniques we developed in this paper. In contrast to recent work that has explored similar ideas [30, 31], tractable inference of CHMMs allows it to handle uncertainty effectively, and its graphical representation allows to form abstraction hierarchies.

We were also motivated by recognizing some fundamental shortcomings of the successor representations (SR) framework [7, 8, 32]. SR represents the current state of an agent by aggregating distributions over its future locations for a given policy. This limits the SR: First, the temporal aggregation means that the ordering of time is lost. SRs do not allow separate access to the current location and future locations, and mix the order of future locations [33]. In contrast, CHMMs provide separate access to the present and the predicted future, all while preserving the ordering. Second, SRs are a function of the policy. It is emphasized [8] that the value function can be easily recomputed when the reward changes, without recomputing the SR. However, what really needs to change when the reward changes is the policy. But this forces recomputing the SR. There is no value in recomputing the value function only in response to a reward change. SRs are strictly less informative than CHMMs, which capture the dynamics of the world instead. CHMMs can update the policy on the fly (and provide a mechanism to do so) since they learn the dynamics of an environment, which is constant for both reward and policy changes. The observation of grid-cell like properties in the eigenvectors of SR could be a property of all methods that employ a transition matrix (see Supplementary Results), and we suspect that this property in itself might not have any behavioral relevance. SR can be used to find communities, but it requires the world to be fully observable without latent states. In contrast, CHMMs have the ability to split aliased observations into different contexts to discover latent graphs and communities.

CHMMs have intriguing connections to schema networks [34], and the observation of schema cells in the hippocampus [35]. Like schema networks, CHMMs encode relational knowledge. Creating different clones for different temporal contexts is similar to idea of *synthetic items* used to address state aliasing [36]. We intend to explore these connections in future work.

In concordance with [29], CHMMs represent sequences of content-free pointers: each pointer can be refer-

ring to a conjunction of sensory events from different modalities. The output from grid cells, path integration signals, is treated as just another sensory modality. Grid cell outputs provide a periodic tiling of uniform space, which is advantageous for learning and navigating maps when other sensory cues are absent. Similarly encoding snapshots from a graphical model for vision [37] as the input to this sequencer might enable the learning of visuo-spatial concepts and visual routines [38], and model the bi-directional influence hippocampus has on the visual cortex. We believe these ideas are promising paths for future exploration.

Although beyond the scope of the current work, hippocampal replay [39] is a phenomenon that could potentially be explained using CHMMs. Related work [40] has shown that an algorithm that rapidly memorizes and gradually generalizes is possible for learning CHMM representation. Learning from rest time replay of sequences can help such an algorithm consolidate and generalize better. Inference time replay might be explained as the searching of trajectories to multiple goals and their vicarious evaluation.

Elucidating how cognitive maps are represented in the hippocampus, how they are acquired from a stream of experiences, and how to utilize them for prediction and planning is not only crucial to understand the inner workings of the brain, but also offers key insights into developing agents with artificial general intelligence. The CHMM model, which we introduce in this paper, provides a plausible answer to each of these questions. We expect this model to be beneficial in both neuroscience and artificial intelligence as a way to produce explicit representations that are easy to interpret and manipulate from multimodal sequential data.

Methods

Since the CHMM can be considered as a sparse restriction of an overcomplete HMM [19], the same learning and inference methods that have been successful on HMMs can be transferred to CHMMs. In contrast with HMMs, however, learning in CHMMs is easier (less learning iterations) because the emission matrix is known and fixed a priori, and cheaper (less cost per iteration for a given number of hidden states) due to the sparsity of the emission (and potentially transition) matrices.

Expectation-Maximization learning of CHMMs

The standard algorithm to train HMMs is the expectation-maximization (EM) algorithm [41] which in this is context is known as the Baum-Welch algorithm. CHMM equations require a few simple modifications with respect the HMM equations: the sparsity of the emission matrix can be exploited to only use small blocks of the transition matrix both in the E and M steps and the actions, if present, should be grouped with the next hidden state (see Figure 1d), to remove the loops and create a chain that is amenable to exact inference.

Learning a CHMM requires optimizing the vector of prior probabilities π: $\pi_u = P(z_1 = u)$ and the transition matrix T: $T_{uv} = P(z_{n+1} = v | z_n = u)$. To this end, we assume the hidden states are indexed such that all the clones of the first emission appear first, all the clones of the second emission appear next, etc. Let E be the total number of emitted symbols. The transition matrix T can then be broken down into smaller submatrices $T(i,j), i,j \in 1 \dots E$. The submatrix $T(i,j)$ contains the transition probabilities $P(z_{n+1}|z_n)$ for $z_n \in \text{hid}(i)$ and $z_{n+1} \in \text{hid}(j)$ (where $\text{hid}(i)$ and $\text{hid}(j)$ respectively correspond to the hidden states (clones) of emissions i and j).

The standard Baum-Welch equations can then be expressed in a simpler form in the case of CHMM. The E-step recursively computes the forward and backward probabilities and then updates the posterior probabilities. The M-step updates the transition matrix via row normalization.

E-Step

$$\alpha(1) = \pi(x_1) \qquad \alpha(n+1)^\top = \alpha(n)^\top T(x_n, x_{n+1})$$
$$\beta(N) = 1(x_N) \qquad \beta(n) = T(x_n, x_{n+1})\beta(n+1)$$

$$\xi_{ij}(n) = \frac{\alpha(n) \circ T(i,j) \circ \beta(n+1)^\top}{\alpha(n)^\top T(i,j) \beta(n+1)}$$
$$\gamma(n) = \frac{\alpha(n) \circ \beta(n)}{\alpha(n)^\top \beta(n)}.$$

M-Step

$$\pi(x_1) = \gamma(1)$$
$$T(i,j) = \sum_{n=1}^{N} \xi_{ij}(n) \oslash \sum_{j=1}^{E} \sum_{n=1}^{N} \xi_{ij}(n).$$

where $\circ$ and $\oslash$ denote the elementwise product and division, respectively (with broadcasting where needed). All vectors are $M \times 1$ column vectors, where M is the number of clones per emission. We use a constant number of clones per emission for simplicity here, but the number of clones can be selected independently per emission.

Computational savings: For a standard HMM with H hidden states, the computational cost for running one EM step on a sequence of length N is $O(H^2N)$ and the required memory is $O(H^2+HN)$ (for the transition matrix and forward-backward messages). In contrast, a CHMM exploits the sparse emission matrix: with M clones per emission, the computational cost is $O(M^2N)$ and the memory requirement is $O(H^2+MN)$, in the worst case. Also, there will be additional savings for every pair of symbols that never appear consecutively in the training sequence (since the corresponding submatrix of the transition matrix does not need to be stored). Memory requirements can be improved further by using the online version of EM described in the Supplementary Materials.

Since $H = ME$, where E is the total number of symbols, an increase in alphabet size will increase the computation cost of HMMs, but will not affect the cost of CHMMs.

Intuitively, the computation advantage of CHMMs over HMMs comes from the sparse emission matrix structure. The sparsity pattern allows CHMMs to only consider a smaller submatrix of the transition matrix when performing training updates and inference, while HMMs must consider the entire transition matrix.

Action-augmented CHMMs

The action-augmented CHMMs are a minor extension of normal CHMMs in which an action happens at every timestep (conditional on the current hidden state) and the hidden state of the next timestep depends not only on the current hidden state, but also on the current action. The probability density function is

$$P(x_1,\ldots,x_N,a_1,\ldots,a_{N-1}) = \sum_{\{z_n \in \mathrm{hid}(x_n)\}_{n=1}^{N}} P(z_1)\prod_{n=1}^{N-1} P(z_{n+1},a_n|z_n) \tag{3}$$

and the standard CHMM can be recovered by integrating out the actions. All the previous considerations about CHMMs apply to action-augmented CHMMs and the EM equations for learning them are also very similar:

E-Step

$$\alpha(1) = \pi(x_1) \qquad \alpha(n+1)^\top = \alpha(n)^\top T(x_n,a_n,x_{n+1})$$
$$\beta(N) = 1(x_N) \qquad \beta(n) = T(x_n,a_n,x_{n+1})\beta(n+1)$$

$$\xi_{ikj}(n) = \frac{\alpha(n)\circ T(i,a_n,j)\circ \beta(n+1)^\top}{\alpha(n)^\top T(i,a_n,j)\beta(n+1)}$$
$$\gamma(n) = \frac{\alpha(n)\circ\beta(n)}{\alpha(n)^\top\beta(n)}.$$

M-Step

$$\pi(x_1) = \gamma(1)$$
$$T(i,k,j) = \sum_{n=1}^{N} \xi_{ikj}(n) \oslash \sum_{k=1}^{N_a} \sum_{j=1}^{E} \sum_{n=1}^{N} \xi_{ikj}(n).$$

where N_a is the number of actions and $T(i,k,j) = P(z_{n+1}, a_n = k | z_n)$ for $z_n \in \text{hid}(i)$ and $z_{n+1} \in \text{hid}(j)$, i.e., a portion of the action-augmented transition matrix.

We have observed that convergence can be improved by using a small pseudocount κ. A pseudocount is simply a small constant that is added to the accumulated counts statistic matrix $\sum_{n=1}^{N} \xi_{ikj}(n)$ and ensures that any transition under any action has non-zero probability. This ensures that at test time the model does not have zero probability for any observations stream. When the pseudocount is only used to improve convergence, one can run EM a second time with no pseudocount, warmstarting from the result of the EM with pseudocount. To use the pseudocount, we only need to change our transition matrix update to be $T(i,k,j) = (\kappa + \sum_{n=1}^{N} \xi_{ikj}(n)) \oslash \sum_{k=1}^{N_a} \sum_{j=1}^{E} (\kappa + \sum_{n=1}^{N} \xi_{ikj}(n))$. The pseudocount can be interpreted as the hyperparameter of a Laplacian prior that is set on the transition matrix, and EM as solving MAP inference for such hyperparameter. As any prior, the pseudocount has a regularization effect that helps generalization when the amount of training data is small in comparison with the capacity of the model.

Inference

Since the resulting model (with the action a_n and hidden state z_{n+1} collapsed in a single variable) forms a chain, inference on it using belief propagation (BP) is exact. When no evidence is available for a given variable, BP will simply integrate it out, so we can for instance train a model with actions and then, at test time, use it even if no actions are available. We can still ask the model which observation is the most likely in the next timestep, or even several timesteps ahead, and BP will produce the exact answer by analytically integrating over all possible past and future actions, and even over the unseen future observations when necessary.

The same model can be used to generate sequences (e.g., to generate plausible observations and actions that would correspond to wandering in a previously learned room) simply by applying ancestral sampling [42] to the conditionals that describe the model after learning (i.e., the transition and emission matrices).

A consequence of the above for spatial data is that an agent roaming the world can infer where in an environment it is located (z_n) and then predict which actions are feasible at that location, which is useful for navigation. One can even condition on a future location to discover which set of actions can take you there, and which observations you are expected to see on the way there, see e.g. Figures 3b and c. This is essentially planning as inference [20].

All of this flexible querying is performed by running a single algorithm (BP) on the same model (without retraining) and only changing the selection of which evidence is available and which probabilistic predictions are requested.

Supplementary materials

Supplementary results

Learning spatial representations

In order to learn the spatial representation of a room with a CHMM, we let an agent roam in a room and receive a stream of local visual cues paired with the executed actions. Note that there are two factors that complicate learning: on the one hand, the visual cues do not need to be unique to each location in the room. In fact, in an empty room, every location in the room away from the walls and the corners looks the same (see Figure 2c). On the other hand, even though the agent's proprioception lets it know the identifier of the executed action, there is no meaning associated to it. I.e., every time that the agent moves west, it knows it is doing the same thing, but it does not have any prior knowledge of what that thing is.

In the case of empty rooms, most of the observations received by the agent are the same (the empty observation, as it wanders through empty space) and it is hard for the agent to locate itself in the room, since only the walls and corners provide context. We have experimented with CHMMs learning in empty rooms of different sizes. We use 50000 steps[2] in a room of size $(4+d)\times(6+d)$, where d is a parameter controlling the room size. For rooms of size 6×8 and below, EM learning recovers exactly the structure of the room. For larger sizes, it starts to make some mistakes in its understanding of the room, slightly decreasing its predictive ability as the room grows, see Figure 6a. Figure 6b shows the learned transition matrix in graph form for a room of size 9×11. The graph looks almost perfect, but if we follow the path between observations "1" and "3", we should traverse seven observations of type "7", whereas there are only six. The CHMM has merged two physical locations in the room (that have a large neighborhood of identical sensory cues) into the same perceived location.

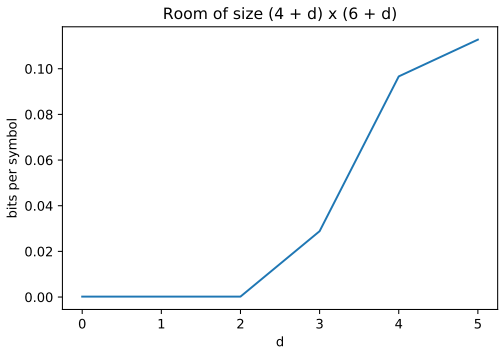
Learning the structure of a room would be trivial if each observation was unique to a single room location. In that case, a CHMM with only one clone would learn the correct solution in one EM step. We experiment with different numbers of unique symbols randomly placed in a room. Figure 6c shows that in a room of size 6×8 (depicted in Figure 2a) the performance degrades as the number of unique symbols decreases, with recovery being exact only when the number of unique symbols is 4 or more. The number of EM iterations required for convergence[3] is also affected, as shown in Figure 6d.

Successor representation of the CHMM transition matrix

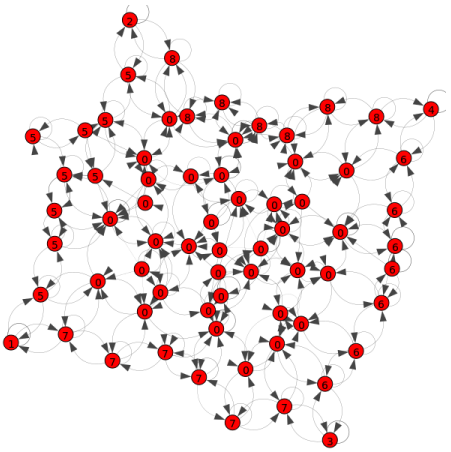
A CHMM can be thought of as a higher-order extension of the successor representation (SR) [8, 27]. However unlike the SR, which assumes full observation of the state, CHMM can handle partial or aliased observations. We take the CHMM learned from the aliased 6×8 room in Figure 2a and generate the SR from the CHMM transition matrix. Then we identify which clones correspond to which spatial locations by observing

[2] Other parameters for this experiment are the number of clones used (70, which is theoretically enough to exactly recover even the largest room), a pseudocount of $2\cdot 10^{-3}$ used in the EM procedure and a maximum number of EM iterations of 1000.

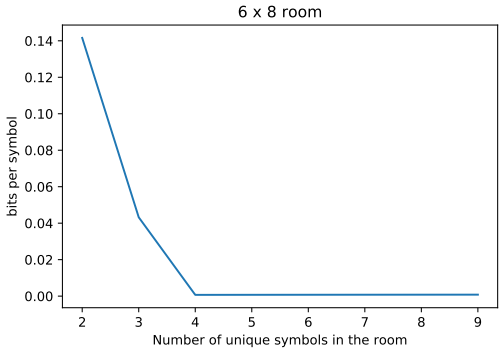
[3] The parameters for this experiment are: 10000 recorded steps, a CHMM with 30 clones (enough for exact recovery for all the numbers of unique symbols tested), a pseudocount of 10^{-2} in the EM procedure and a maximum number of EM iterations of 1000.



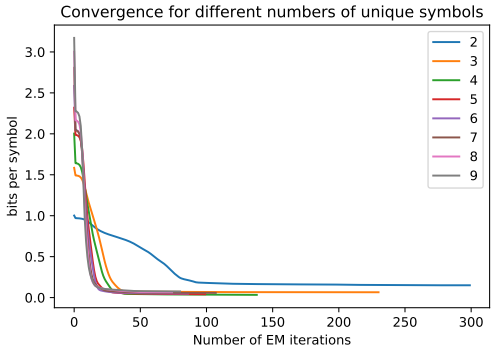
(a) Larger empty rooms result in increased bps



(b) Imperfect graph learned from a large empty room



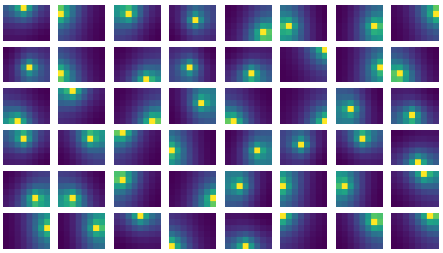
(c) Fewer unique observations result in increased bps



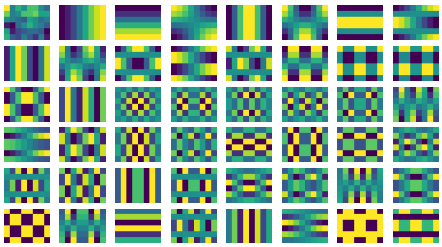
(d) Fewer unique observations make learning harder

Figure 6: The effect of varying room size and the number of unique observations in each room.

which clones activate in each location during inference. In Figure 7a we visualize the SR for each hidden state. We also compute the eigenvectors of the matrix containing the SR of each state. We visualize these eigenvectors in Figure 7b and we also observe various grid patters of different scales, similarly to [8].



(a) Successor Representations in 6x8 room



(b) Eigenvectors of Successor Representation

Figure 7: Successor representation and eigenvectors derived from the transition matrix of a CHMM. The CHMM was trained with data collected from a random walk in a rectangular room.

Supplementary experimental details

Emergence of spatial maps from aliased sequential observations

For this experiment we collected a stream of 50000 action-observations pairs. We learned a CHMM with 20 clones (a total of 360 states) with pseudocount $2 \cdot 10^{-3}$ and ran EM for 1000 iterations. This gets a result that is very close to the global minimum: when Viterbi decoded, only 48 distinct states are in use, which is the theoretical optimum on a 6×8 grid. Viterbi training [24] is used to refine the previous solution.

Transitive inference: disjoint experiences can be stitched together into a coherent whole

Figure 2e showcases the CHMM's ability to stitch together two disjoint room experiences when the rooms overlap. For this experiment, we randomly generate two square rooms of size 8×6 with 15 different observations each. We make both rooms share a 3×3 patch in their corners as shown in Figure 2e.

We sample a random walk of length 10000 of action-observation pairs on each room, always avoiding to take actions that would make the random walk move outside of the room. We use 20 clones, which is enough to fully recover both rooms separately, and use a pseudocount of 10^{-2}. We run EM (on both sequences simultaneously, as two independent observations of the same CHMM) for a maximum of 100 iterations. After EM convergence, we additionally use Viterbi decoding (with no pseudocount) to remove unused clones. The recovered transition matrix is plotted in Figure 2f, showing that the two rooms that were experienced separately have been stitched together. Predictive performance on the stitching of the two rooms is perfect (indicating that learning succeeded) after a few observations required for the agent to locate itself.

Notice that there is another patch in the first room that is identical to the merged patches, but was not merged. The model is using the sequential information to effectively identify patches that can be merged while respecting the observational data and context, and not simply looking for locally identical patches to merge.

Learned spatial maps form a reusable structure to explore similar environments

For this experiment, we train on a 6×8 room using 10000 action-observation pairs. We call this Room 1, see Figure 3a. There are only 20 unique symbols in the room, some of which are repeated. The pseudocount is set to 10^{-2} and we use 20 clones (in this case, only 7 clones are strictly required to memorize the room). The regularizing pressure of the pseudocount effectively removes redundant clones. Training is done using EM for a maximum of a 100 iterations. This results in an almost perfect discovery of the underlying graph. Then we set the pseudocount to 0 and continue the training using Viterbi training [24]. This results in perfect discovery of the underlying graph with no duplicate clones. Predictions become perfect after a few initial observations required to know where in the room we are.

When we try to partially learn Room 2 with a few samples from its periphery (see Figure 3a), we create a new CHMM with the transition matrix that we learned from Room 1 and keep it fixed. The emission matrix is initialized uniformly and learned using EM. The whole data for learning the emission matrix are only the

20 action-observation pairs seen in Figure 3a. At that point, we fix the model and query it for a return path plan, both with and without blockers in the path. The results are displayed in Figure 3b and Figure 3c.

In Figure 3d-f we showcase the increase in data efficiency when we transfer the learned topology to a new room with different observations. First we ignore the results from training on Room 1 and train on a new room, Room 2, from scratch following the same procedure outlined above. We train on the first N action-observation pairs and predict for the rest. We average (geometrically) the probability of getting the next observation right for the last 8000 samples of the 10000 available. This results in the graph in Figure 3e, where N is shown in the horizontal axis. Then we repeat the same procedure, but instead of training from scratch from a random transition matrix with fixed emissions, we fix the transition matrix that we got from training in Room 1 and we learn the emission matrix, which is initialized to uniform. EM for the emission matrix converges in a few iterations. Once all nodes have been observed (when the red curve achieves 1.0), this procedure converges to perfect predictions in one or two EM iterations. This results in the graph in Figure 3f, where again the horizontal axis shows N, the number of training action-observation pairs.

Representation of paths and temporal order

To learn the CHMM on the maze in Figure 3a, we sample 5000 paths along each of the stochastic routes that are shown. The number in each cell indicates the observation received at that cell, and the arrows indicate possible transitions. We consider both sequences as independently generated by the model, and run EM to optimize the probability of both simultaneously. We allocate 20 clones for each other observation. By inspecting the sum-product forward messages of belief propagation at each step as the rat navigates the two routes, we can see the distributions over clones. We observe they are over disjoint subsets of the clones. To generate the paths from the CHMM shown in Figure 3b (producing only paths that are consistent with the route), we sample an observation from the normalized messages from hidden state to observation during forward message passing. Finally, to extract the communities and generate the visualization in Figure 3c, we run the InfoMap algorithm [43] on the graph defined by CHMM transition matrix.

In Figure 4d-f, we replicate the experiments of Sun and colleagues [25] as follows. First, we learned a CHMM with 20 clones per observation on a sequence of observations sampled from three laps around the maze shown in Figure 4d. The start and end positions were unique observations. Training was terminated when perfect learning of the underlying graph was achieved. Community detection (explained below) revealed that each sensory observation was encoded by a unique clone, akin to the chunking cells found by Sun et al.

Retrieval and Remapping

We generate random walks (random actions out of up, right, down, left) of length 10000 in each of 5 randomly generated mazes. The observation is the 3×3 cell surrounding the current location (mapped to a unique integer). We learn a CHMM on these sequences. After learning, we sum the forward messages of sum-product belief propagation in each maze to get a distribution over hidden states for each maze. Now on a test sequence, we can use the forward messages and these clone distributions per maze to infer the

probability of being in each maze at each timestep. Figure 5b shows these predictions.

To correct errors in a corrupted observation sequence we modify the emission matrix to generate a random symbol with a small probability, thus modeling errors. Then we perform sum-product message passing on sequences with errors and find the most likely a posteriori value for each symbol. In our case, we only perform a forward pass, which provides an online estimation (based only on past data) of the MAP solution. We will use a corruption probability of 20% in our experiments, uniform over the incorrect symbols. Figure 5c shows the result of this online error correction.

Community detection and hierarchical planning

In all figures, custom Python scripts were used to convert the transition matrix of the CHMM into a directed graph. This graph was then visualized using built-in functions in `python-igraph` (`https://igraph.org/python/`). Similarly, community detection was performed using igraph's built-in `infomap` function.

In the hierarchical planning experiments shown in Figure 5d-f, we first generated each room by drawing a random integer between 1 and 12 with repetitions to serve as observations. The rooms were then connected via bridges (observation 13, colored black) and were tiled to form a maze as shown in Figure 5d. Next, we trained a CHMM with 40 clones per observation using 1000 random restarts as described above. The learned CHMM achieved perfect prediction accuracy, suggesting perfect learning of the underlying graph. Community detection on the learned CHMM was performed using `igraph`. To form the top-level graph shown in Figure 5f, we collapsed each distinct community into a single node. These communities roughly corresponded to each of the rooms in the maze. In some cases, certain rooms were partitioned into multiple communities.

To compute the shortest trajectory between two locations on the maze, we first computed the shortest path in the top-level graph using Dijkstra's algorithm, implemented in `networkx` (`https://networkx.github.io/`). This returned the sequence of rooms to be visited in order to reach the goal from a start point. Next, we pruned the community graph to include only clones corresponding to these rooms and then found the shortest path in this reduced graph, which gave the exact sequence of observations from the start position to the goal (denoted by the black arrow in Figure 5d. This hierarchical approach was consistently better than searching for the shortest path on the full maze itself with an average 25% fewer steps (n = 10 mazes). To determine to what extent the partitioning of the CHMM transition matrix into communities helped planning, we formed surrogate communities which no longer respected room boundaries. This resulted in a planned trajectory with an average 35% more steps than the hierarchical plan.

Adaptive and online EM variant of CHMMs

Although the sequences in the experiments of this work are not too large, we might want to be able deal with cases in which there is a very long incoming stream of observations, so long that we cannot even store it in its entirety. In order to handle this case, we can simply extend the previous EM algorithms to make them online.

The adaptive, online version of the EM algorithm in the **Methods** section is obtained by splitting the sequence in B batches $b = 1 \dots B$ and performing EM steps on each batch successively. This allows the model to adapt to changes in the statistics if those happen over time. The statistics $\xi_{ij}(n)$ of batch b are now computed from the E-step over that batch, using the transition matrix $T^{(b-1)}$ from the previous batch. After processing batch b, we store our running statistic in $A^{(b)}$ as:

$$A_{ij}^{(b)} = (1-\lambda) \sum_{k=1}^{b} \lambda^{b-k} \sum_{n \in \text{batch}(k)} \xi_{ij}(n)$$

and then compute the transition matrix $T^{(b)}$ as

$$T^{(b)}(i,j) = A_{ij}^{(b)} \oslash \sum_{j=1}^{E} A_{ij}^{(b)}$$

where $0 < \lambda < 1$ is a memory parameter and $n \in \text{batch}(k)$ refers to the time steps contained in batch k. For $\lambda \to 1$, $T^{(b)}(i,j)$ coincides with the transition matrix from the **Methods** section. For smaller values of λ, the expected counts are weighed using an exponential window,[4] thus giving more weight to the more recent counts.

To learn from arbitrarily long sequences, we consider an online formulation and express $A_{ij}^{(b)}$ recursively:

$$A_{ij}^{(b)} = \lambda A_{ij}^{(b-1)} + (1-\lambda) \sum_{n \in \text{batch}(b)} \xi_{ij}(n),$$

so that the expected counts of the last observed batch are incorporated into the running statistics.

[4] Normalization of the exponential window is unnecessary, since it will cancel when computing $T^{(b)}(i,j)$.

References

1. Redish, A. D. Vicarious trial and error. *Nature Reviews Neuroscience* **17,** 147 (2016).
2. Tolman, E. C. Cognitive maps in rats and men. *Psychological review* **55,** 189 (1948).
3. Niv, Y. Learning task-state representations. *Nature neuroscience* **22,** 1544–1553 (2019).
4. Moser, E. I., Kropff, E. & Moser, M.-B. Place cells, grid cells, and the brain's spatial representation system. *Annu. Rev. Neurosci.* **31,** 69–89 (2008).
5. Eichenbaum, H., Dudchenko, P., Wood, E., Shapiro, M. & Tanila, H. The hippocampus, memory, and place cells: is it spatial memory or a memory space? *Neuron* **23,** 209–226 (1999).
6. Schapiro, A. C., Turk-Browne, N. B., Norman, K. A. & Botvinick, M. M. Statistical learning of temporal community structure in the hippocampus. *Hippocampus* **26,** 3–8 (2016).
7. Dayan, P. Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation* **5,** 613–624. eprint: https://doi.org/10.1162/neco.1993.5.4.613. https://doi.org/10.1162/neco.1993.5.4.613 (1993).
8. Stachenfeld, K. L., Botvinick, M. M. & Gershman, S. J. The hippocampus as a predictive map. *Nature neuroscience* **20,** 1643 (2017).
9. Whittington, J., Muller, T., Mark, S., Barry, C. & Behrens, T. in *Advances in Neural Information Processing Systems 31* (eds Bengio, S. *et al.*) 8484–8495 (Curran Associates, Inc., 2018). http://papers.nips.cc/paper/8068-generalisation-of-structural-knowledge-in-the-hippocampal-entorhinal-system.pdf.
10. Frank, L. M., Brown, E. N. & Wilson, M. Trajectory encoding in the hippocampus and entorhinal cortex. *Neuron* **27,** 169–178 (2000).
11. Wood, E. R., Dudchenko, P. A., Robitsek, R. J. & Eichenbaum, H. Hippocampal neurons encode information about different types of memory episodes occurring in the same location. *Neuron* **27,** 623–633 (2000).
12. Colgin, L. L., Moser, E. I. & Moser, M.-B. Understanding memory through hippocampal remapping. *Trends in neurosciences* **31,** 469–477 (2008).
13. Duvelle, É. *et al.* Insensitivity of place cells to the value of spatial goals in a two-choice flexible navigation task. *Journal of Neuroscience* **39,** 2522–2541 (2019).
14. Cormack, G. V. & Horspool, R. N. S. Data compression using dynamic Markov modelling. *The Computer Journal* **30,** 541–550 (1987).
15. Hawkins, J., George, D. & Niemasik, J. Sequence memory for prediction, inference and behaviour. *Philosophical Transactions of the Royal Society B: Biological Sciences* **364,** 1203–1209. ISSN: 14712970. http://www.ncbi.nlm.nih.gov/pubmed/19528001%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2666719 (May 2009).
16. Xu, J., Wickramarathne, T. L. & Chawla, N. V. Representing higher-order dependencies in networks. *Science advances* **2,** e1600028 (2016).

17. Cui, Y., Ahmad, S. & Hawkins, J. Continuous online sequence learning with an unsupervised neural network model. *Neural Computation* **28,** 2474–2504. ISSN: 1530888X. arXiv: 1512.05463. http://arxiv.org/abs/1512.05463 (Dec. 2016).

18. Dedieu, A. *et al.* Learning higher-order sequential structure with cloned HMMs. arXiv: 1905.00507. http://arxiv.org/abs/1905.00507 (May 2019).

19. Sharan, V., Kakade, S. M., Liang, P. S. & Valiant, G. *Learning overcomplete hmms* in *Advances in Neural Information Processing Systems* (2017), 940–949.

20. Attias, H. *Planning by probabilistic inference.* in *AISTATS* (2003).

21. Morton, N. W., Sherrill, K. R. & Preston, A. R. Memory integration constructs maps of space, time, and concepts. *Current opinion in behavioral sciences* **17,** 161–168 (2017).

22. Behrens, T. E. *et al.* What is a cognitive map? Organizing knowledge for flexible behavior. *Neuron* **100,** 490–509 (2018).

23. Blum, K. I. & Abbott, L. A model of spatial map formation in the hippocampus of the rat. *Neural computation* **8,** 85–93 (1996).

24. Jelinek, F. Continuous speech recognition by statistical methods. *Proceedings of the IEEE* **64,** 532–556 (1976).

25. Sun, C., Yang, W., Martin, J. & Tonegawa, S. CA1 pyramidal cells organize an episode by segmented and ordered events (2019).

26. Grieves, R. M., Wood, E. R. & Dudchenko, P. A. Place cells on a maze encode routes rather than destinations. *Elife* **5,** e15986 (2016).

27. Gershman, S. J. The successor representation: its computational logic and neural substrates. *Journal of Neuroscience* **38,** 7193–7200 (2018).

28. Tomov, M. S., Yagati, S., Kumar, A., Yang, W. & Gershman, S. J. Discovery of Hierarchical Representations for Efficient Planning. *bioRxiv.* eprint: https://www.biorxiv.org/content/early/2019/03/28/499418.full.pdf. https://www.biorxiv.org/content/early/2019/03/28/499418 (2019).

29. Buzsáki, G. & Tingley, D. Space and time: The hippocampus as a sequence generator. *Trends in cognitive sciences* **22,** 853–869 (2018).

30. Whittington, J., Muller, T., Mark, S., Barry, C. & Behrens, T. *Generalisation of structural knowledge in the hippocampal-entorhinal system* in *Advances in neural information processing systems* (2018), 8484–8495.

31. Whittington, J. C. *et al.* The Tolman-Eichenbaum Machine: Unifying space and relational memory through generalisation in the hippocampal formation. *bioRxiv,* 770495 (2019).

32. Momennejad, I. *et al.* The successor representation in human reinforcement learning. *Nature Human Behaviour* **1,** 680 (2017).

33. Momennejad, I. & Howard, M. W. Predicting the future with multi-scale successor representations. *BioRxiv,* 449470 (2018).

34. Kansky, K. *et al. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics* in *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), 1809–1818.

35. Baraduc, P., Duhamel, J.-R. & Wirth, S. Schema cells in the macaque hippocampus. *Science* **363,** 635–639 (2019).

36. Holmes, M. P. *et al. Schema learning: Experience-based construction of predictive action models* in *Advances in Neural Information Processing Systems* (2005), 585–592.

37. George, D. *et al.* A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs. *Science* **358,** eaag2612 (2017).

38. Lázaro-Gredilla, M., Lin, D., Guntupalli, J. S. & George, D. Beyond imitation: Zero-shot task transfer on robots by learning concepts as cognitive programs. *arXiv preprint arXiv:1812.02788* (2018).

39. Skaggs, W. E. & McNaughton, B. L. Replay of neuronal firing sequences in rat hippocampus during sleep following spatial experience. *Science* **271,** 1870–1873 (1996).

40. Rikhye, R. V., Guntupalli, J. S., Gothoskar, N., Lázaro-Gredilla, M. & George, D. V. Memorize-Generalize: An online algorithm for learning higher-order sequential structure with cloned Hidden Markov Models. *bioRxiv,* 764456 (2019).

41. Wu, C. J. *et al.* On the convergence properties of the EM algorithm. *The Annals of statistics* **11,** 95–103 (1983).

42. Bishop, C. M. *Pattern recognition and machine learning* (springer, 2006).

43. Rosvall, M. & Bergstrom, C. T. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences* **105,** 1118–1123 (2008).