Predictive Coding for Cognitive Mapping

Thesis by
James Andrew Gornet

In Partial Fulfillment of the Requirements for the
degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2027
Defended July 15, 2026

James Andrew Gornet
ORCID: 0000-0002-5431-7340

In Memory of My Father

Michael K. Gornet

December 12, 1959 – March 10, 2016

Dedicated in love and gratitude.

ACKNOWLEDGEMENTS

First and foremost, I express my deepest gratitude to my advisor, Matt Thomson, for his mentorship, unwavering support, and for fostering an environment where daring questions can be tackled.

A special thank you to my undergraduate advisor, Uygar Sumbul, for building my scientific foundation and setting me on the path that ultimately led to this dissertation. I am also incredibly grateful to my committee members—Thanos Siapas, Erik Winfree, and Matilde Marcolli. Moreover, Matilde Marcolli especially helped identify new avenues I had overlooked.

Finally, navigating graduate school would have been impossible without the camaraderie, late-night discussions, and intellect of my brilliant cohort and colleagues. To my fellow lab members and Computation and Neural Systems colleagues David Brown, Tatyana Dobreva, Shichen Liu, Arjuna Subramanian, Pranav Bhamidipati, Zach Martinez, Alec Lourenco, Jeremy Bernstein, Guru Raghavan, Aman Bhargava, Alexander Detkov, Meera Prasad, Surya Hari, and Michael Zellinger: thank you for the invaluable friendship and moral support that made this journey so rewarding.

ABSTRACT

Paths within a given space are constrained by the space's geometry. Any data generated by a path must then reflect the path's base space. A path's data constrained by the base space's geometry—which we called the *lifting property*—is the fundamental relation this thesis explores. Specifically, this thesis examines how mathematical functions—and statistical algorithms—that generate a path's data recover the base space's geometry—termed *predictive coding for cognitive mapping*.

It is shown that mathematical functions that generate a path's data recover the base space's geometry. We show that statistical estimators that generate a path's data can also recover the base space's geometry in a simulated natural environment. We also shown that mathematical functions recover a dynamical system if the function generates observations on a dynamical system's trajectory. We demonstrate applications of this principle to a bipedal robot's kinematics. Finally, we extend predictive coding for mapping—predicting future observations from past observations to build an environmental map—to plan and explore within the environment.

PUBLISHED CONTENT AND CONTRIBUTIONS

1. Gornet, J. & Thomson, M. Automated Construction of Cognitive Maps with Visual Predictive Coding. *Nature Machine Intelligence* **6.** 820–833. doi:10.1038/s42256-024-00863-1 (July 2024).
   J.G. conceived the project with M.T., developed the framework, implemented the models, performed all experiments and analysis, and wrote the manuscript. M.T. supervised the project and edited the manuscript.

2. Gornet, J. & Thomson, M. *Neural Networks Learn an Environment's Geometry in Latent Space by Performing Predictive Coding on Visual Scenes* in. NeurIPS 2022 Workshop on Information-Theoretic Principles in Cognitive Systems (Nov. 21, 2022).
   J.G. conceived the project with M.T., developed the framework, implemented the models, performed all experiments and analysis, and wrote the manuscript. M.T. supervised the project and edited the manuscript.

3. Huang, Y. *et al. Neural Networks with Recurrent Generative Feedback* in *Advances in Neural Information Processing Systems* **33** (Curran Associates, Inc., 2020), 535–545.
   J.G. helped implement the theory, performed the experiments and analysis, and participated in the writing of the manuscript.

4. Bagherian, D. *et al. Fine-Grained System Identification of Nonlinear Neural Circuits* in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Association for Computing Machinery, New York, NY, USA, Aug. 14, 2021), 14–24. doi:10.1145/3447548.3467402.
   J.G. contributed the theoretical work, proving that a sign constraint on the weights is a necessary condition for system recovery, and participated in the writing of the manuscript.

CONTENTS

LIST OF FIGURES

NOTATION

$\gamma : \mathbb{R} \to \mathcal{S}$	path
$\mathcal{S}$	state space
$\mathcal{A}$	action space
$\mathcal{O}$	observation space
$\phi : \mathcal{S} \to \mathcal{O}$	observation function
$\delta : \mathcal{A} \times \mathcal{S} \to \mathcal{S}$	transition function
$\hat{\mathcal{S}}$	latent space
$\hat{\phi} : \hat{\mathcal{S}} \to \mathcal{O}$	decoder
$p_\theta(x) \equiv p(x;\theta)$	the (probability) density function of a statistical estimator with parameters θ
$p_\theta(o_t \| o_{<t})$	the predictive coder
$p_{\pi_\theta}(a \| s)$	policy density
$V : \mathbb{S} \to \mathbb{R}$	value function
$Q : \mathbb{A} \times \mathbb{S} \to \mathbb{R}$	Q-function
$\pi_1(X, x_0)$	the fundamental group of the topological space X with the base point x_0
$\mathbb{E}_{x \sim p(x)}[x]$	the expectation of x given the probability density function $p(x)$
$\mathcal{T}$	the time space
$\Phi : \mathcal{T} \times X \to X$	the evolution operator
$q_\omega(z \| \tau)$	the discriminator model

PREFACE

A space's geometry constrains the possible paths within such space. Any data generated by a path is then restricted by the path's base space. This work studies the converse question: if a function can generate a path's data, then can it recover the path's base space? This area of research is commonly called *predictive coding for spatial mapping*. The question underlies many—seemingly unrelated—disciplines such as reinforcement learning, neuroscience, and robotics and has recently become an active area of research. Our purpose in this book is twofold: to provide an introduction to the subject and to present the central results of our work from a self-contained perspective.

Chapter I is directed at the general reader interested in predictive coding for spatial mapping. The knowledge of calculus, linear algebra, and probability theory will suffice for most of the chapter. Although some mention of more sophisticated topics is made from time to time, the passages containing such material are inessential and can be glossed over without loss of continuity. The use of mathematics, statistics, and neuroscience is homogeneous. Using various theory of path mapping, we have attempted to explain the development of the ideas involved in predictive coding.

Chapter II presents the key mathematical results for predictive coding in spatial mapping and describes statistical implementations of predictive coding. The style is more formal, and some of the material may require familiarity with algebraic topology and reinforcement learning.

Chapters III and IV are practically independent of one another. Chapter III is mainly concerned with the extension of predictive coding for more general spaces. The main theme is the unique path lifting theorem and its extension to predictive coding.

Chapter IV is devoted to the application of predictive coding for efficient exploration and mapping. Key results in this section will be a culmination of the mathematical ideas presented in Chapter IV.

Chapter 1

INTRODUCTION

Humans naturally build mental representations of their surroundings using direct sensory information, rather than relying on precise coordinates or formal measurements[1,2,3]. While machine learning—specifically in the field of simultaneous localization and mapping[4,5,6,7,8]—uses complex, specialized calculations to process visual and movement data, the human brain appears to operate differently. The versatility of human "cognitive maps" implies a single, universal underlying strategy that works across all senses, allowing humans to map the world through sound[9], touch[10], and even concepts[11,12,13,14] and memories[15]. Empirical evidence suggest that the brain uses common cognitive mapping strategies for spatial and non-spatial sensory information so that common mapping algorithms might exist that can map and navigate over not only visual but also semantic information and logical rules inferred from experience[16]. In such a paradigm reasoning itself could be implemented as a form of navigation within a cognitive map of concepts, facts, and ideas[17,18,19].

This internal mapping capacity is not a passive reflection of the environment but an active, predictive process. By integrating multimodal sensory streams[20], the brain constructs a coherent model that facilitates navigation, object manipulation, and even abstract reasoning. Unlike hand-crafted geometric representations, these biological maps are inherently flexible, adapting to novel environments and sensory modalities without the need for explicit reprogramming.

Current artificial algorithms[21,22,23] struggle in complex natural environments due to high-dimensional state spaces and the semantic gap between raw data and interaction. While classical methods like simultaneous localization and mapping[5,6,7,7] excel in structured settings, they lack the ability to identify high-level affordances in chaotic environments. In addition, these classical methods operate strictly in visual-spatial mapping and cannot extend to general interactive environments such as large language models[24], computer use agents[25,26,27], and underactuated robots[28]. Machine learning models that operate on interactive environments require a general internal representation of

sensory information.

Recently, large language models[29,30] have shown promise in structured, discrete data, but emulating this success in the physical world remains difficult. Visual information is continuous and dense, making the tokenization of scenes significantly more complex than text. The lack of sparsity in visual streams leads to a *curse of dimensionality*, in which models struggle to maintain physical consistency amidst infinite pixel configurations and low signal-to-noise ratios.

In addition, a significant hurdle is the *interface gap*: large language models are easily steered via natural language prompts, whereas models on physical environments often require brittle reward engineering or target-image matching[31,32,33]. Bridging this gap requires models to align visual information with conceptual context, creating a multimodal interface where high-level concepts—like "wash the dishes"—can be translated into specific adjustments in predictive simulation and motor control.

Effective action also requires a sophisticated internal representation of the agent's own body. Traditional hand-coded kinematic models are fragile and fail when mechanical properties change, sensors drift, or the agent interacts with heavy tools. An adaptive model must treat the agent's kinematics as a dynamic, learned component of the environment, integrating proprioception to allow for seamless and robust interaction[34].

As robotic systems gain more degrees of freedom, their configuration spaces become intractable for traditional inverse kinematics. Biological systems manage this complexity through synergies—coupled movements that reduce effective dimensionality[35,36]. To achieve human-like fluid motion, world models must learn task-relevant manifolds that capture the essential structures of kinematic interaction within the broader environment, moving beyond exhaustive search in configuration space.

This thesis proposes a framework for creating such internal representations of both the environment's space and agent's actions and is divided into two main parts. The first deals with the theory for how predictive coding—predicting future observations from the past—builds an internal representation for an environment. The second part shows applications regarding predictive coding's internal representation such as exploration and planning.

1.1 The principle of path-dependent persistence

Any movement an agent makes is fundamentally limited by the topological properties of its environment. These geometric constraints dictate which transitions are possible and how distant two points are within a space. One of the most powerful consequences of continuous movement is that temporal proximity serves as a proxy for spatial proximity. If an agent moves at a finite velocity, then two observations made closely together in time must correspond to states that are close together in the environment's geometry.

This property—that a space's geometry restricts the paths with its space—we call the *principle of path-dependent persistence*, which posits that the structure of information generated by a path is inextricably linked to the sequence of states it occupies. Unlike independent and identically distributed samples, the data encountered by a navigating agent possesses a temporal continuity that reflects the underlying topology of the environment. This persistence implies that the history of an agent's path contains information about the global structure of the space, even if the agent only has access to local observations.

The Grid World and Cylinder World

In this chapter, we will develop an intuition for how predictive coding—predicting future observations from past observations—implicitly builds a spatial map of an environment by studying the discrete case on a two-dimensional lattice. While the discrete case can appear to be an oversimplification, many insights in the discrete case correspond to the general case, topological spaces. Indeed, many fundamental theorems in reinforcement learning are stated only in the discrete case.

We will study two simplified environments to develop an intuition of how different geometries affect their respective paths and data-generated from these paths. First, we consider the Grid World. In this environment, an agent moves between adjacent cells in a two-dimensional lattice. Second, we consider the Cylinder World. In this environment, the agent moves between adjacent cells in a cylinder lattice.

In the Grid World, the environment is defined as discrete state space $\mathcal{S} \subset \mathbb{Z}^2$ where each state $s \in \mathcal{S}$ is represented by an integer coordinate (x, y). The topology of the environment is determined by the adjacency of cells in the

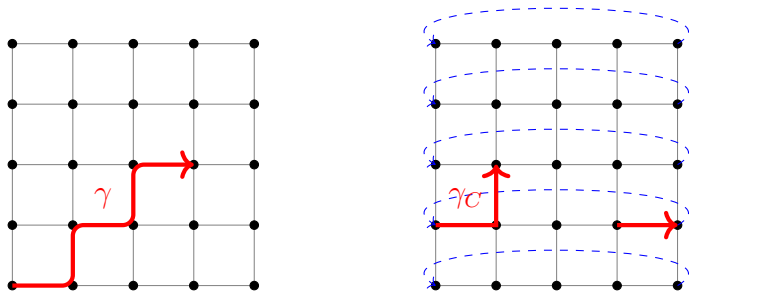


Figure 1.1: Visual representation of the state spaces and connectivity for the Grid World (left) and Cylinder World (right), with example paths γ and γ_C shown in red.

lattice, typically defined by a von Neumann neighborhood:

$$N((x, y)) = \{(x', y') \in \mathcal{S} : |x - x'| + |y - y'| = 1\} \tag{1.1}$$

We define the agent's action space $\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}\}$, where each action $a \in \mathcal{A}$ corresponds to a transition between adjacent states in the lattice. A path γ of length T is defined as a sequence of positions $\mathbf{s} = (s_1, s_2, \ldots, s_T)$, where each $s_t \in \mathcal{S}$ and the transition from s_t to s_{t+1} is mediated by an action $a_t \in \mathcal{A}$. Furthermore, we define an observation function $\phi : \mathcal{S} \to \mathcal{O}$ that maps each state $s \in \mathcal{S}$ to an observation $o = \phi(s)$ in the observation space $\mathcal{O}$. The sequence of observations generated by a path is then given by $\mathbf{o} = (\phi(s_1), \phi(s_2), \ldots, \phi(s_T))$.

The Cylinder World introduces a non-trivial topology by wrapping one dimension of a grid. While locally it resembles the flat Grid World, its global structure is cylindrical, meaning that an agent moving in one direction will eventually return to its starting point. This periodic boundary condition serves as a critical test case for an agent's ability to distinguish between local, flat geometry and global, curved topology.

We define the Cylinder World as a discrete state space $\mathcal{S}_C = \mathbb{Z}_w \times \mathbb{Z}$, where w is the width of the cylinder. Each state $s \in \mathcal{S}_C$ is represented by coordinates (x, y) where $x \in \{0, 1, \ldots, w - 1\}$ and $y \in \mathbb{Z}$. The adjacency neighborhood $N_C((x, y))$ incorporates the periodic boundary condition:

$$N_C((x, y)) = \{(x', y') \in \mathcal{S}_C : |(x - x') \pmod w| + |y - y'| = 1\} \tag{1.2}$$

The agent's action space $\mathcal{A}$, paths γ, and observation function $\phi : \mathcal{S}_C \to \mathcal{O}$ are

defined analogously to the Grid World, with the transition dynamics governed by the cylindrical neighborhood N_C.

Time as an implicit metric

Consider the state s_t of the agent—its exact coordinates in the lattice—as not directly accessible. Instead, the agent is restricted to a stream of observations $o_t = \phi(s_t)$, where the underlying state is a latent variable that must be inferred from the history of observations and actions. The challenge for the agent is to construct an internal representation that captures the essential geometric information of the environment despite this fundamental unobservability.

Although the true states s_t are hidden, the temporal structure of the observation stream $\mathbf{o} = (o_1, \ldots, o_T)$ provides an implicit metric for the environment's geometry. Because the agent moves through the state space $\mathcal{S}$ with a finite velocity—taking at most one step per unit time—temporal proximity in the sequence of observations serves as a reliable proxy for spatial proximity in the latent space.

Formally, let $d_{\mathcal{S}}(s_i, s_j)$ be the shortest-path distance between two states in the environment. For any path $\mathbf{s} = (s_1, s_2, \ldots, s_T)$, the transition between consecutive states is constrained such that $d_{\mathcal{S}}(s_t, s_{t+1}) \leq 1$. By the triangle inequality, for any two time steps t and $t + \Delta t$, the spatial distance between the corresponding states is bounded by the elapsed time:

$$d_{\mathcal{S}}(s_t, s_{t+\Delta t}) \leq \sum_{k=0}^{\Delta t-1} d_{\mathcal{S}}(s_{t+k}, s_{t+k+1}) \leq \Delta t \tag{1.3}$$

This inequality demonstrates that within a bounded time interval Δt, the agent's movement is restricted to a local neighborhood of radius Δt in the state space. Consequently, observations o_t and $o_{t+\Delta t}$ that are close in time are guaranteed to be generated from states that are close in distance. Conversely, observations o_t and $o_{t'}$ that are far in time are not guaranteed to be far in distance—as is the case for loops. This implicit temporal metric provides the structural property that allows a predictive model to organize its latent representations geometrically, effectively "stitching" together local observations into a globally consistent manifold.

Statement 1. *In the Grid World and Cylinder World, for a path γ and any two time steps t and $t + \Delta t$, the spatial distance between the corresponding*

states is bounded by the elapsed time:

$$d_{\mathcal{S}}(s_t, s_{t+\Delta t}) \leq \sum_{k=0}^{\Delta t-1} d_{\mathcal{S}}(s_{t+k}, s_{t+k+1}) \leq \Delta t$$

Any function that generates the path's data must recover the space's geometry

The previous section established that time acts as an implicit metric for an agent's experience; within a bounded interval, the temporal continuity of observations provides a structural prior for local spatial proximity. However, for a model to recover the global geometry and topology of its environment, it must not only understand local distance but also distinguish between distinct physical states.

We will show that if the observation sequence is injective—that is, every path in the environment corresponds to a unique sequence of observations—then the sequence of data implicitly encodes the entire topology of the space. Notice that two different states can have the same observation; the only requirement is the observation *sequences* be unique. Under this condition, the temporal transitions between observations provide a complete map of the traversable edges in the environment's graph. By requiring a model to predict these sequences, we effectively force it to find an internal representation that preserves this injective mapping, thereby ensuring that the latent topology is isomorphic to the physical world.

For simplicity, we will first tackle the special case where every observation corresponds to a unique state. We will then prove the general case: if every path (of duration T) is unique, then space S' of latent model $\hat{\phi}$ is isomorphic to the physical space S.

Formally, let $\gamma = (s_1, s_2, \ldots, s_T)$ be a path in the state space $\mathcal{S}$, and let $\mathbf{o} = \phi(\gamma) = (\phi(s_1), \ldots, \phi(s_T))$ be the corresponding observation sequence. Suppose we have a latent model consisting of a latent path $\gamma' = (\hat{s}_1, \ldots, \hat{s}_T)$ in a latent space $\hat{\mathcal{S}}$ and a decoding function $\hat{\phi} : \hat{\mathcal{S}} \to \mathcal{O}$ such that the generated data matches the oracle $\phi(\gamma)$:

$$\hat{\phi}(\hat{s}_t) = \phi(s_t) \quad \text{for all } t \in \{1, \ldots, T\} \tag{1.4}$$

Assuming that the observation function ϕ is injective, there exists a unique mapping $f : \mathcal{S} \to \hat{\mathcal{S}}$ defined by $f = \hat{\phi}^{-1} \circ \phi$ such that $\hat{s}_t = \psi(s_t)$.

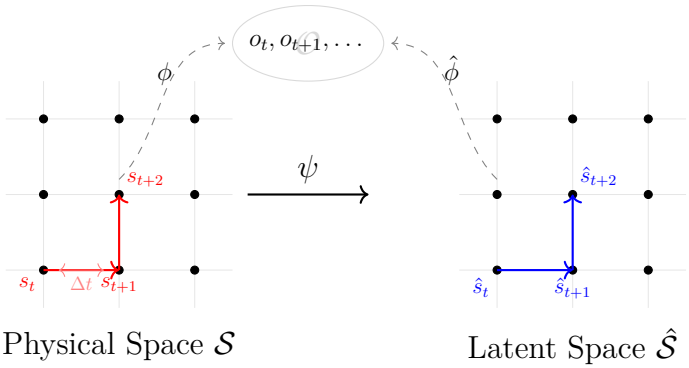


Figure 1.2: **Demonstration of the implicit metric of time and the isomorphism ψ between physical and latent spaces.** Temporal continuity in $\mathcal{S}$ ensures local spatial proximity, while injective mappings ϕ and $\hat{\phi}$ to a shared observation space $\mathcal{O}$ force the latent representation $\hat{\mathcal{S}}$ to recover the environment's topology.

Theorem 1.1.1. *Let $G = (V, E)$ be a bounded lattice graph where $V \subset \mathbb{Z}^2$ and the edge set is defined by strict adjacency, $E = \{\{u, v\} : |u - v| = 1\}$. Suppose γ and γ' are two valid paths in G. If there exist injective embeddings ϕ and $\hat{\phi}$ into $\mathbb{R}^N$ such that their image sets satisfy $\phi(\gamma) = \hat{\phi}(\gamma')$, then the transformation Γ mapping γ to γ' is a graph isomorphism.*

Proof sketch. Let $X = \phi(\gamma) = \hat{\phi}(\gamma')$ represent the shared image set in $\mathbb{R}^N$. Because both ϕ and $\hat{\phi}$ are strictly injective functions evaluated over discrete sets, they behave as bijections onto X. This allows us to define the spatial composition $f = \hat{\phi}^{-1} \circ \phi$, which serves as a well-defined bijection from the vertices of γ directly to the vertices of γ'.

To account for temporal alignment, let τ act as an index reparameterization function. We define the global transformation operator Γ such that it aligns both the spatial coordinates and the path sequencing:

$$\gamma'(t) = \Gamma(\gamma(t)) = f(\gamma(\tau(t)))$$

Because γ and γ' are valid paths within the lattice, their consecutive states inherently satisfy the adjacency rule $|s_i - s_{i+1}| = 1$. The transformation Γ, acting spatially through the bijection f, perfectly maps the sequence of edges in γ onto the sequence of edges in γ'. Consequently,

Γ preserves the ℓ_1 adjacency relation of the lattice, guaranteeing that for any two vertices x and y on the path:

$$|x - y| = 1 \implies |\Gamma(x) - \Gamma(y)| = 1$$

Because Γ is bijective on the vertices and strictly preserves edge connectivity, it is a graph isomorphism. ■

The proof sketch provided above relies on the assumption of point-wise injectivity—that every individual state produces a unique observation. While this provides a clear intuition for how latent states map to physical ones, it is a restrictive condition that does not account for environments where multiple locations produce the same visual observation.

The fundamental condition for geometric recovery is not point-wise injectivity, but rather the injectivity of the sequence generating function. By shifting our focus from individual points to "path objects"—sequences of states and their corresponding observations—we can generalize the proof to cases where individual states may be indistinguishable, yet are uniquely identified by their temporal context. In this general framework, we demonstrate that as long as the mapping from paths to observation sequences is injective, any model capable of accurately predicting these sequences must necessarily recover an internal representation that is isomorphic to the environment's underlying graph structure.

Theorem 1.1.2 (Predictive Coding Mapping Theorem (discrete case))**.** *Let $S = (V, E)$ and $S' = (V', E')$ be graphs, and let $\mathcal{P}_N(S)$ and $\mathcal{P}_N(S')$ denote their respective sets of valid paths of length N. Let $\phi : V \to \mathbb{R}^M$ and $\phi' : V' \to \mathbb{R}^M$ be local observation functions. Define the sequence generators $\Phi : \mathcal{P}_N(S) \to \mathbb{R}^{N \times M}$ and $\hat{\Phi} : \mathcal{P}_N(S') \to \mathbb{R}^{N \times M}$ compositionally, such that for any path $p = (v_1, \ldots, v_N)$, the generator concatenates the vertex labels:*

$$\Phi(p) = [\phi(v_1), \ldots, \phi(v_N)]^T$$

Assume that the spaces (S, ϕ) and (S', ϕ') satisfy the following conditions with respect to paths of length N:

1. *Complete coverage: Every vertex and edge in the graph is traversed by at least one valid path in $\mathcal{P}_N$.*

2. *Local Unambiguity: No two paths in $\mathcal{P}_N$ originate from distinct vertices sharing the same initial label under ϕ. Furthermore, for any vertex v, all directed edges originating from v must lead to successor vertices with strictly unique labels.*

3. *Distinguishability: For any vertex v, let $L(v)$ denote its future sequence space—the set of all suffix label sequences generated by successfully completing a length-N path from v. For any distinct vertices $u, w \in V$, their future spaces must differ: $L(u) \neq L(w)$.*

If the global sequence spaces generated by both graphs are identical, namely $\Phi(\mathcal{P}_N(S)) = \hat{\Phi}(\mathcal{P}_N(S'))$, then the graphs S and S' are isomorphic.

> ***Proof sketch.*** First, the local unambiguity condition structurally guarantees that a sequence of local observations corresponds to at most one valid topological path, rendering both sequence generators Φ and $\hat{\Phi}$ strictly injective. Specifically, the initial observation uniquely identifies the starting vertex, and each subsequent observation unambiguously resolves the next traversed vertex. Because the global sequence spaces are identical and both mappings are strictly path-injective, there exists a sequence-preserving bijection between the valid paths of S and S', given by $\Gamma = \hat{\Phi}^{-1} \circ \Phi : \mathcal{P}_N(S) \to \mathcal{P}_N(S')$. For every path $\gamma \in \mathcal{P}_N(S)$, its mapped path $\gamma' = \Gamma(\gamma)$ strictly preserves the observation sequence: $\hat{\Phi}(\gamma') = \Phi(\gamma)$.
>
> We construct a vertex mapping $f : V \to V'$. By the complete coverage condition, every vertex $u \in V$ is traversed by at least one valid path $\gamma \in \mathcal{P}_N(S)$ at some step index k. We define $f(u) = u'$, where u' is the k-th vertex of the correspondingly mapped path $\gamma' = \Gamma(\gamma)$.
>
> To establish that f is strictly well-defined, we must prove it is independent of the choice of path and index. Suppose another path $\rho \in \mathcal{P}_N(S)$ traverses u at step index j. The set of all valid observation sequences that can extend paths from u to successfully complete a path in $\mathcal{P}_N$ exactly defines the future sequence space $L(u)$. Because the path bi-

jection Γ strictly preserves full observation sequences step-by-step, the corresponding mapped vertices in S'—namely, the k-th vertex of γ' and the j-th vertex of ρ'—must admit this exact same set of valid future sequences to successfully reproduce the identical global sequence space. Consequently, both vertices in S' possess a future sequence space identically equal to $L(u)$. By the distinguishability condition on S', no two distinct vertices can share the same future sequence space. Therefore, these two mapped vertices must be identical, ensuring $f(u)$ is uniquely determined.

By symmetric application of these conditions via Γ^{-1}, the mapping f has a well-defined inverse, establishing it as a bijection between V and V'.

Finally, we show that f preserves structural adjacency. Let $(u, v) \in E$ be a directed edge in S. By the trimness condition, there exists at least one path $\gamma \in \mathcal{P}_N(S)$ that traverses the edge (u, v) at consecutive indices k and $k+1$. Under the path bijection Γ, the corresponding path γ' in S' maintains the exact same sequence of observations and must therefore consecutively traverse the mapped vertices $f(u)$ and $f(v)$ at indices k and $k+1$. This guarantees the existence of the directed edge $(f(u), f(v)) \in E'$. Because f is a well-defined bijection that strictly preserves edge connectivity and local observations, it constitutes a graph isomorphism, yielding $S \cong S'$. ■

While we have shown that a model can recover the environment's geometry, the extent of this recovery is fundamentally limited by the information content of the observations. The observation function ϕ acts as a lens through which the agent perceives the world; if this lens collapses distinct physical states into identical sensory representations, the agent will naturally recover a geometry that reflects this degeneracy.

Consider the pathological case where every state in the environment produces the same observation: $\phi(s) = c$ for all $s \in \mathcal{S}$. In this scenario, the observation sequence conveys no information about movement or location. Any arbitrary graph structure would be consistent with this data, and thus no specific geometry can be recovered. The environment appears as a single point, or a structureless void, because the observations fail to provide the necessary

sensory information to distinguish between positions.

A more constructive example is found when the observations in a flat Grid World exhibit periodicity. Suppose $\phi(x, y) = \phi(x \pmod w, y)$, where w is some fixed width. Even though the physical environment is an infinite Euclidean plane, an agent traversing this space will perceive a repeating sequence of observations in the x-direction. For a model to accurately predict these observations, it will naturally find a latent representation that is cylindrical. The model wraps the x-dimension to match the observed periodicity, effectively recovering a Cylinder World. This demonstrates that a predictive model does not recover the true physical space, but rather the most efficient quotient space induced by the observation function—it recovers the environment's geometry up to an isomorphism in the observation space. In both cases, the observation sequences are not unique, which leads to the observed degeneracy.

For arbitrary graphs, these pathological cases arise under several conditions. First, if vertices or edges are unreachable by paths within the environment, then the observation sequences cannot recover the environment's geometry within unreachable spaces. Second, if there exist local ambiguity such as pathological cases where all observations are the same, then the environment's geometry does not adequately constrain a path's observation sequence. Lastly, two different locations cannot generate the same set of observation sequences. In the periodic Grid World and Cylinder World example, the Grid World contains locations (x, y) and $(x + w, y)$ that generate the same observation sequences.

To eliminate these pathological cases, the observation sequences must satisfy these conditions. First, the paths must completely cover the entire graph: every vertex and directed edge in the graph is traversed by at least one valid path in $\mathcal{P}_N$. Second, no location can contain local ambiguity. Specifically, no two paths in $\mathcal{P}_N$ begin at distinct starting vertices that share the same feature label—if $p = (u_1, \dots)$ and $q = (w_1, \dots)$ are in $\mathcal{P}_N$, then $\phi(u_1) = \phi(w_1) \implies u_1 = w_1$, and for any vertex v, no two distinct outgoing edges (v, u_1) and (v, u_2) lead to successor vertices with the same feature label

$$\phi(u_1) = \phi(u_2) \implies u_1 = u_2.$$

Lastly, each vertex must be distinguishable by its set of observation sequences. For any vertex v, its future sequence space $L(v)$ is the set of all suffix feature

sequences generated by paths originating from v that successfully complete a length-N path in $\mathcal{P}_N$. For any $u \neq w$, their valid futures must differ: $L(u) \neq L(w)$. Otherwise, graphs within the equivalence class that generate the same observation sequences are not guaranteed to be graph isomorphic.

By satisfying these conditions, a graph behaves as a minimal deterministic finite automaton—or more precisely, a minimal directed acyclic word graph. This formal equivalence provides a rigorous mathematical framework for understanding what it means for an agent to "learn" an environment. By viewing the environment as an automaton, we can map the components of spatial navigation directly onto the foundational constructs of formal language theory. Let the set of all possible unique local observations serve as an alphabet, $\Sigma = \{\phi(s) \mid s \in \mathcal{S}\}$. An observation sequence generated by a path of length N is therefore a word $w \in \Sigma^N$, and the global sequence space—the set of all such valid sequences—constitutes a formal language, $\mathcal{L}_N = \Phi(\mathcal{P}_N)$.

Because the path length is strictly bounded to N, the sequence space is strictly finite. In formal language theory, any finite set of words is trivially a regular language. Consequently, the mechanisms governing how the agent internalizes the environment's geometry are entirely dictated by the rules of regular languages and their corresponding acceptors, deterministic finite automata (DFA).

The conditions outlined above—complete coverage, local unambiguity, and distinguishable futures—are not arbitrary; they are the exact topological requirements of the Myhill-Nerode theorem[37,38] applied to a spatial graph. The theorem establishes that any regular language is defined by a unique right-congruence relation. Two prefix strings are considered equivalent if and only if they share the exact same set of valid appending suffixes.

In the context of our spatial graph, this suffix set is identical to the future sequence space $L(v)$. If the environment permitted $L(u) = L(w)$ for spatially distinct vertices u and w, these two locations would be Nerode-equivalent. From the perspective of the language $\mathcal{L}_N$, they are computationally identical. The agent's observation function would be unable to disentangle them, forcing a predictive model to merge these states in its latent representation to achieve optimal efficiency. This merging is precisely what occurs in the Cylinder World degeneracy, where periodic states share identical valid futures and collapse into a single latent equivalence class.

By explicitly mandating $L(u) \neq L(w)$ for all $u \neq w$, we enforce that every physical location occupies a distinct algebraic equivalence class. The mathematical consequence of this is profound: the Myhill-Nerode theorem guarantees that there exists exactly one strictly minimal DFA (up to state renaming) that recognizes a given regular language.

Because the physical environment graph is constrained to behave as this unique minimal DFA, any predictive model that perfectly learns to generate the observation language $\mathcal{L}_N$ must internalize a latent transition structure that is structurally isomorphic to the true physical graph. The agent is no longer simply modeling the extensional outputs of the space. The structural constraints mathematically guarantee that the extensional equivalence of the observation language dictates an intensional equivalence of the learned geometry.

1.2 Recovering the data-generating function using predictive coding

The mathematical foundations established in the previous section demonstrate a existence result: if a data-generating function accurately reproduces the observation sequences of an environment, its latent space must be isomorphic to the physical state space. However, this proof remains nonconstructive, as it does not specify a mechanism for how such a function can be recovered from observation data.

In this section, we propose predictive coding as the fundamental mechanism for recovering this data-generating function. Predictive coding operates as an iterative, unsupervised framework where the agent continuously generates predictions of its future observations based on its past observations. By minimizing the prediction error between these expected observations and the actual observations encountered along a path, the model progressively refines its internal representations. Because the process of error minimization recovers the data-generating function, predictive coding effectively recovers the environment's geometric and topological constraints into the model's parameters.

We have formulate the data-generating function $\hat{\phi}$ from a probabilistic view. In other words, $\hat{\phi}(\gamma')$ is a random variable rather than a sequence. As a random variable, the data-generating function has

1. a sample space Ω

2. probability distribution $\mathbb{P} : \Omega \to [0, 1]$

For the sample space Ω, data-generating function has the observation space $\mathcal{O}$, and for the probability distribution $\mathbb{P}$, the data-generating function has

$$\mathbb{P}[\hat{\phi}(\gamma')] = \mathbb{P}[\hat{\phi}(s'_1, \dots,)].$$

To formalize the predictive coding framework, we start with the joint probability of a sequence of observations up to time t, denoted as $p(o_1, o_2, \dots, o_t)$. Using the chain rule of probability, this joint distribution can be factorized as:

$$p(o_1, \dots, o_t) = p(o_t|o_{<t})p(o_{<t}) = \prod_{k=1}^{t} p(o_k|o_{<k}) \tag{1.5}$$

The core objective of predictive coding is to maximize this probability, or equivalently, to minimize the negative log-likelihood of the observations. To recover the environment's geometry, we introduce a latent state $s_t \in \hat{\mathcal{S}}$ that summarizes the history of observations. By assuming a Markovian structure where s_t depends only on s_{t-1} and the current observation o_t depends only on s_t, we can expand the conditional probability $p(o_t|o_{<t})$ by marginalizing over the latent states:

$$p(o_t|o_{<t}) = \sum_{\hat{\mathcal{S}}} p(o_t|s_t, o_{<t})p(s_t|o_{<t})ds_t \tag{1.6}$$

Under the standard model assumptions, o_t is conditionally independent of previous observations given the current state s_t, so $p(o_t|s_t, o_{<t}) = p(o_t|s_t)$. Furthermore, the latent state s_t is predicted from the previous state distribution $p(s_{t-1}|o_{<t-1})$. Thus, we obtain the recursive predictive coding equation:

$$p(o_t|o_{<t}) = \sum_{\hat{\mathcal{S}}} p(o_t|s_t) \left(\sum_{\hat{\mathcal{S}}} p(s_t|s_{t-1})p(s_{t-1}|o_{<t-1})ds_{t-1} \right) ds_t \tag{1.7}$$

This formulation demonstrates that predicting the next observation o_t requires two fundamental components: a transition model $p(s_t|s_{t-1})$ that captures the latent dynamics and an observation model $p(o_t|s_t)$ that maps latent states to sensory data. By iteratively maximizing this probability across the entire path, the predictive coding equation encodes that probabilistic formulation of the data-generating function.

1.3 Estimating the predictive coding distribution using maximum-likelihood estimation

The objective of predictive coding is to estimate a model $p_\theta(o_1, \ldots, o_T)$ that maximizes the likelihood of the observation sequence. Using the chain rule, we can factorize the joint probability as a product of conditional distributions:

$$p_\theta(o_1, \ldots, o_T) = \prod_{t=1}^{T} p_\theta(o_t | o_{<t}) \tag{1.8}$$

By maximizing the log-likelihood, we obtain the predictive coding objective:

$$\log p_\theta(o_1, \ldots, o_T) = \sum_{t=1}^{T} \log p_\theta(o_t | o_{<t}) \tag{1.9}$$

To evaluate each term $p_\theta(o_t|o_{<t})$, we marginalize over the latent state s_t:

$$p_\theta(o_t | o_{<t}) = \sum_{s_t \in \hat{\mathcal{S}}} p_\theta(o_t|s_t) p_\theta(s_t|o_{<t}) \tag{1.10}$$

where $p_\theta(s_t|o_{<t})$ represents the agent's belief about its current state given the history of observations. This belief is updated recursively according to the transition dynamics $p_\theta(s_t|s_{t-1})$:

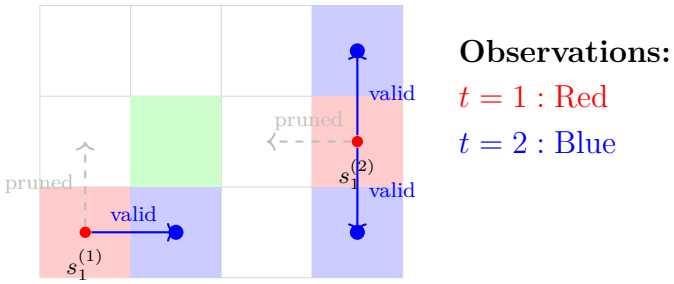
$$p_\theta(s_t | o_{<t}) = \sum_{s_{t-1} \in \hat{\mathcal{S}}} p_\theta(s_t|s_{t-1}) p_\theta(s_{t-1}|o_{<t-1}) \tag{1.11}$$

By iteratively minimizing the prediction error (maximizing the likelihood), the model recovers the transition graph and observation mapping that best explain the sensory data.

In the Grid World environment, observations are deterministic—that is, each latent state x maps to a unique observation $o = \phi(x)$—the observation model is simply $p(o_t|s_t) = \mathbb{1}(\phi(s_t) = o_t)$. Under this assumption, the predictive coding task of finding the most likely sequence of latent states $\hat{x}_{1:T}$ can be formulated using maximum likelihood estimation. We need to find the state sequence that maximizes the joint probability:

$$\hat{s}_{1:T} = \arg\max_{s_{1:T}} p(s_{1:T}, o_{1:T}) = \arg\max_{s_{1:T}} p(s_1) p(o_1|s_1) \prod_{t=2}^{T} p(s_t|s_{t-1}) p(o_t|s_t) \tag{1.12}$$

With deterministic observations, this reduces to finding the most probable path $\prod p(s_t|s_{t-1})$ that is strictly consistent with the observed sequence, such



Path Filtering in Grid World

Figure 1.3: Visualizing the pruning mechanism. At $t = 1$, the observation 'Red' identifies two potential latent states. As the agent moves at $t = 2$ and observes 'Blue', the algorithm extends paths to all adjacent cells but prunes those (dashed) that do not match the new observation. This recursive filtering ensures that only trajectories consistent with the entire sensory history are maintained.

that $\phi(s_t) = o_t$ for all t. This provides an efficient mechanism for the agent to infer its trajectory and the environment's structure from sensory feedback.

Consider an agent navigating a Grid World where each cell has a unique color or sensory signature. If the agent moves from a "red" cell to a "blue" cell, we can maintain a set of candidate paths that are consistent with this sequence. At each step, it discards any path that does not end in a cell matching the current observation. Among the surviving paths, it keeps track of the one with the highest cumulative transition probability.

In the Grid World, if we assume a uniform transition prior where an agent is equally likely to move to any adjacent cell, we can effectively find the shortest path through the latent graph that visits the observed colors in the correct order. If the observations are sufficiently distinct, the set of possible trajectories quickly collapses down to a single unique path, effectively localizing the agent in the latent space. This process demonstrates how the temporal structure of predictive coding allows the agent to unsupervisedly disentangle its movement from the sensory stream and recover the underlying topological map.

1.4 Maps from predictive coding can be used for planning

The recovered latent space $\hat{\mathcal{S}}$—isomorphic to the environment's physical geometry—provides a natural foundation for planning and decision-making. Once an agent has learned the transition dynamics $p(s_t|s_{t-1}, a_{t-1})$ from estimating predictive coding, it can propagate reward values through its latent space to find

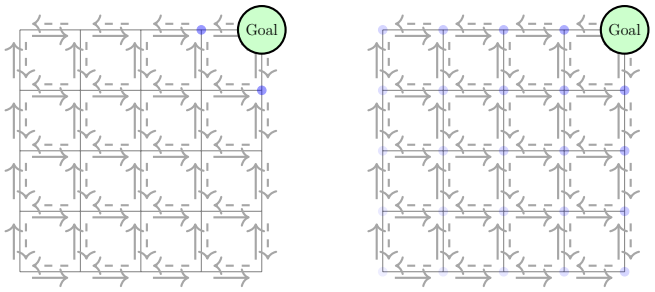


Figure 1.4: **Illustration of reward diffusion through value iteration.** A reward at the goal state (top-right) propagates across the lattice according to the Bellman update, creating a numeric value gradient and a corresponding policy (gray arrows) that leads the agent to the goal.

the optimal paths to a goal.

To enable goal-directed behavior, the predictive coding framework is extended by augmenting the state-observation space with a reward signal $r_t \in \mathbb{R}$. If we do not have access to the reward signal explicitly, a model can be trained to predict not only the next sensory observation but also the scalar reward associated with each state: $p(o_t, r_t|s_t)$. Within this augmented model, planning is formulated as the task of finding a sequence of actions $\mathbf{a}_{t:t+H}$ that maximizes the expected cumulative reward $\mathbb{E}[\sum_{k=t}^{t+H} \gamma^{k-t} r_k]$ over a future horizon H. Because the latent states s_t preserve the topological connectivity of the environment, any trajectory optimized in the latent space corresponds to a traversable path in physical space, transforming the problem of navigation into one of reward maximization within a learned geometric coordinate system.

A common method for propagating these rewards across the latent space is value iteration, which can be understood as a form of reward diffusion over the environment's topology[39,40]. In the context of the Grid World, if a significant reward is placed at a specific cell s_{goal}, the value iteration process iteratively "spreads" this value to neighboring cells. Formally, this is achieved through the update rule:

$$V(s) = \max_{a \in \mathcal{A}} \left[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V(s') \right] \tag{1.13}$$

Each iteration of this update allows the reward information to diffuse one step further into the surrounding lattice. As the value $V(s)$ stabilizes, it creates a global potential field, or value gradient, across the environment. By following

the gradient of this field—always choosing the action a that leads to the state s' with the highest $V(s')$—the agent can navigate from any point in the Grid World toward the goal. This diffusion process is only possible because the model has recovered the correct adjacency structure of the space; if the latent graph were incorrect, the reward would diffuse into unreachable shortcuts or fail to reach distant states, leading to suboptimal or impossible plans.

1.5 Exploration can be formulated as a wavefront propagation

Efficiently mapping an environment can be formalized as a graph traversal task where the objective is to visit every node in a hidden graph. This process can be understood through the lens of propagating a wavefront, a method analogous to the *fast marching method* used in numerical analysis. In this framework, the agent acts as the source of a propagating front that expands through the known subregion of the space, identifying the boundaries between explored and unexplored territory. By prioritizing transitions toward this frontier, the agent systematically pushes the wavefront forward, effectively filling the topological map.

The primary challenge, however, is that the agent lacks prior knowledge of the environment's underlying geometry; the underlying position in the Grid World are only perceive through ambiguous observations. However, as the agent traverses the environment, it builds a set of observation-action sequences. By performing predictive coding on these observation-action sequences, the predictive code can recover a subgraph of the Grid World. Within the latent space, the exploration wavefront can be propagated to generate novel observation-action sequences. Exploration acts as iterative loop: performing predictive coding to recover a subregion map and generating novel observation-action sequences from the subregion map. By performing this iterative loop, the observation sequences will converge to the Grid World's observation sequences: the latent space will converge to the Grid's World lattice.

The exploration algorithm can be formalized as an iterative four-step procedure:

1. **Sequence Generation:** The agent executes a exploratory policy to collect a set of observation-action sequences $\mathcal{D}_k = \{(o_1, a_1, \ldots, o_T)\}$. Initially, this policy may be random, but it progressively becomes more targeted as the internal map grows.

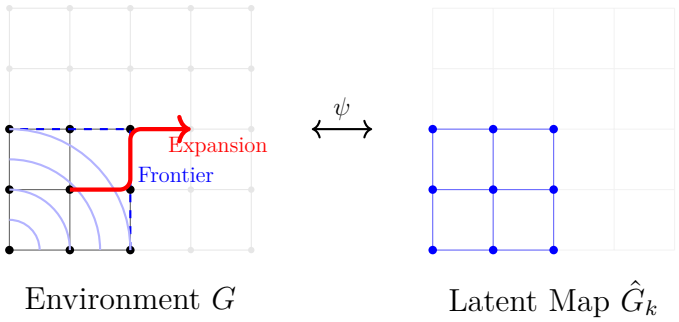


Figure 1.5: **Wavefront propagation for autonomous exploration.** (Left) The agent propagates a wavefront (blue arcs) through its explored subregion to identify the frontier of unknown space. (Right) The predictive coding algorithm recovers a latent subgraph $\hat{G}_k$ that is topologically isomorphic to the environment. Targeted expansion (red path) generates new data to extend the map iteratively.

2. **Latent Subgraph Recovery:** By applying the predictive coding framework to $\mathcal{D}_k$, the agent recovers a latent subgraph $\hat{G}_k = (\hat{V}_k, \hat{E}_k)$. According to the **predictive coding mapping theorem** (Theorem 1), this latent structure is a graph isomorphism of the physically traversed subgraph $G_k \subseteq G$, ensuring that the internal representation faithfully mirrors the environment's connectivity.

3. **Wavefront Propagation:** A wavefront is propagated across the latent subgraph $\hat{G}_k$ to identify frontier nodes—latent states that lie at the boundary of known space. Similar to the fast marching method, this propagation calculates the "time of arrival" for the explorer's front at every point in the latent manifold.

4. **Targeted Expansion:** New observation-action sequences are generated by planning trajectories that lead the agent toward these frontiers. These new experiences provide the data necessary to "un-hide" adjacent nodes and edges in the environment.

This cycle repeats, with each iteration expanding the latent subgraph $\hat{G}_{k+1}$ until the wavefront has fully propagated across the entire accessible space. The iterative loop—where the wavefront drives discovery and predictive coding refines the geometric coordinate system—allows the agent to autonomously construct a globally consistent representation of any complex space from scratch. By the conclusion of this process, the latent space converges to the Grid

World's lattice, and the internal model provides a complete, topologically accurate map for subsequent navigation and planning.

Proof of exploration and mapping

Does the wavefront propagation algorithm guarantee exploration of an environment's state space, and does the algorithm recover the environment's state space? We will show that the algorithm guarantees exploration and mapping in the discrete case.

To show exploration and mapping, let the physical environment be a deterministic, strongly connected graph $G = (V, E)$. We assume G is finite, such that the total number of edges $|E| = M < \infty$. Let $G_k = (V_k, E_k) \subseteq G$ denote the physically explored subgraph at iteration k, and let $\mathcal{D}_k$ be the set of unique observation-action sequences collected by traversing G_k. In addition, let $\hat{G}_k = (\hat{V}_k, \hat{E}_k)$ denote the recovered latent representation constructed by the predictive coding framework to reproduce the sequences in $\mathcal{D}_k$.

To show the exploration algorithm converges, we will use the discrete case of the predictive coding mapping theorem:

Theorem 1.1.2 (Predictive Coding Mapping Theorem (discrete case))**.** *Let $S = (V, E)$ and $S' = (V', E')$ be graphs, and let $\mathcal{P}_N(S)$ and $\mathcal{P}_N(S')$ denote their respective sets of valid paths of length N. Let $\phi : V \to \mathbb{R}^M$ and $\phi' : V' \to \mathbb{R}^M$ be local observation functions. Define the sequence generators $\Phi : \mathcal{P}_N(S) \to \mathbb{R}^{N\times M}$ and $\hat{\Phi} : \mathcal{P}_N(S') \to \mathbb{R}^{N\times M}$ compositionally, such that for any path $p = (v_1, \ldots, v_N)$, the generator concatenates the vertex labels:*

$$\Phi(p) = [\phi(v_1), \ldots, \phi(v_N)]^T$$

Assume that the spaces (S, ϕ) and (S', ϕ') satisfy the following conditions with respect to paths of length N:

1. *Complete coverage: Every vertex and edge in the graph is traversed by at least one valid path in $\mathcal{P}_N$.*
2. *Local Unambiguity: No two paths in $\mathcal{P}_N$ originate from distinct vertices sharing the same initial label under ϕ. Furthermore, for any vertex v, all directed edges originating from v must lead to successor vertices with strictly unique labels.*

3. *Distinguishability: For any vertex v, let $L(v)$ denote its future sequence space—the set of all suffix label sequences generated by successfully completing a length-N path from v. For any distinct vertices $u, w \in V$, their future spaces must differ: $L(u) \neq L(w)$.*

If the global sequence spaces generated by both graphs are identical, namely $\Phi(\mathcal{P}_N(S)) = \hat{\Phi}(\mathcal{P}_N(S'))$, then the graphs S and S' are isomorphic.

Because the predictive coding framework is designed to construct $\hat{G}_k$ such that it perfectly predicts the unique collected data $\mathcal{D}_k$, this condition is satisfied at each iteration, guaranteeing $\hat{G}_k \cong G_k$ during exploration.

Finally, define $F_k \subseteq V_k$ as the set of frontier nodes—explored states possessing at least one untraversed outgoing edge. We will first show that the wavefront propagation algorithm converges in $O(|E|)$ time.

Proposition 1 (Proof of convergence)**.** *Given a bounded graph G, the targeted expansion algorithm converges in at most $|E| = M$ iterations.*

> ***Proof sketch.*** At any iteration k prior to convergence, the set of physical frontier nodes is non-empty ($F_k \neq \emptyset$). By construction, the latent graph $\hat{G}_k$ reproduces the unique observation sequences $\mathcal{D}_k$ of the explored space. By Theorem 1.1.2, this identity of sequences guarantees the isomorphism $\hat{G}_k \cong G_k$.
>
> During wavefront propagation, the agent identifies the latent frontier $\hat{F}_k$ and plans a targeted trajectory to a node within this set. Because $\hat{G}_k \cong G_k$ and G is strongly connected, a valid path to the corresponding physical frontier is guaranteed.
>
> By executing an untraversed action from this frontier, the agent collects new observation sequences and adds at least one previously unexplored edge to the physical subgraph, forming G_{k+1}. Therefore, the number of explored edges strictly monotonically increases:
>
> $$|E_k| < |E_{k+1}|$$
>
> Since the underlying graph is finite, the total number of edges is bounded above by M. The strictly increasing integer sequence $|E_0|, |E_1|, \ldots$ must

reach its supremum in $K \leq M$ steps. At this point, no frontiers remain, and the algorithm terminates. ■

Thus, the wavefront propagation algorithm will converge in $O(|E|)$ time. In the following theorem, we will prove that the wavefront propagation algorithm recovers the underlying state space at convergence.

Proposition 2. *Upon convergence, the recovered latent graph is isomorphic to the true underlying graph ($\hat{G}_K \cong G$).*

Proof sketch. Let K be the iteration at which the algorithm converges. By definition of termination, the frontier set is exhausted, meaning $F_K = \emptyset$.

Assume, for the sake of contradiction, that the explored subgraph is strictly smaller than the true graph ($G_K \subset G$). Because G is strongly connected, there must exist at least one unexplored edge $e = (u, v) \in E \backslash E_K$ originating from a node that has already been explored ($u \in V_K$).

However, possessing an unexplored outgoing edge from a known node exactly satisfies the definition of a frontier node, implying $u \in F_K$. This contradicts the termination condition $F_K = \emptyset$. Therefore, our assumption is false, and the explored subgraph must be exactly the true graph:

$$G_K = G$$

At this convergence step K, the collected data $\mathcal{D}_K$ contains the unique observation sequences of the entire graph G_K. The predictive coding framework generates a final latent graph $\hat{G}_K$ that identically reproduces these unique sequences.

Because the observation sequences of $\hat{G}_K$ and G_K are unique and identical, the condition for Theorem 1.1.2 is met, yielding the isomorphism:

$$\hat{G}_K \cong G_K$$

Substituting $G_K = G$ into this relation directly yields:

$$\hat{G}_K \cong G$$

This demonstrates that, upon convergence, the algorithm guarantees the latent graph is a perfect structural isomorphism of the complete bounded physical environment. ■

References

1. Sivak, D. A. & Thomson, M. Environmental statistics and optimal regulation. *PLoS computational biology* **10.** e1003826 (2014).
2. Wang, Z. J. & Thomson, M. Localization of signaling receptors maximizes cellular information acquisition in spatially structured natural environments. *Cell Systems* **13.** 530–546 (2022).
3. Epstein, R. A., Patai, E. Z., Julian, J. B. & Spiers, H. J. The Cognitive Map in Humans: Spatial Navigation and Beyond. *Nature Neuroscience* **20.** 1504–1513 (Nov. 2017).
4. Lynen, S. *et al. Get Out of My Lab: Large-scale, Real-Time Visual-Inertial Localization* (July 2015).
5. Mourikis, A. I. & Roumeliotis, S. I. *A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation* in *Proceedings 2007 IEEE International Conference on Robotics and Automation* (Apr. 2007), 3565–3572.
6. Mur-Artal, R. & Tardós, J. D. Visual-Inertial Monocular SLAM With Map Reuse. *IEEE Robotics and Automation Letters* **2** (Apr. 2017).
7. Thrun, S. & Montemerlo, M. The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *The International Journal of Robotics Research* **25.** 403–429 (May 2006).
8. Thrun, S., Burgard, W. & Fox, D. *Probabilistic Robotics* 647 pp. (MIT Press, Cambridge, Mass, 2005).
9. Aronov, D., Nevers, R. & Tank, D. W. Mapping of a Non-Spatial Dimension by the Hippocampal–Entorhinal Circuit. *Nature* **543.** 719–722 (Mar. 2017).
10. Rosenthal, I. A. *et al.* S1 Represents Multisensory Contexts and Somatotopic Locations within and Outside the Bounds of the Cortical Homunculus. *Cell Reports* **42.** 112312 (Apr. 2023).
11. Huth, A. G., de Heer, W. A., Griffiths, T. L., Theunissen, F. E. & Gallant, J. L. Natural Speech Reveals the Semantic Maps That Tile Human Cerebral Cortex. *Nature* **532.** 453–458 (Apr. 2016).

12. Garvert, M. M., Dolan, R. J. & Behrens, T. E. A Map of Abstract Relational Knowledge in the Human Hippocampal–Entorhinal Cortex. *eLife* **6** (ed Davachi, L.) e17086 (Apr. 2017).

13. Constantinescu, A. O., O'Reilly, J. X. & Behrens, T. E. J. Organizing Conceptual Knowledge in Humans with a Gridlike Code. *Science* **352.** 1464–1468 (June 2016).

14. Nieh, E. H. *et al.* Geometry of Abstract Learned Knowledge in the Hippocampus. *Nature* **595.** 80–84 (July 2021).

15. Corkin, S. Lasting Consequences of Bilateral Medial Temporal Lobectomy: Clinical Course and Experimental Findings in H.M. *Seminars in Neurology* **4.** 249–259 (June 1984).

16. Behrens, T. E. J. *et al.* What Is a Cognitive Map? Organizing Knowledge for Flexible Behavior. *Neuron* **100.** 490–509 (Oct. 2018).

17. Whittington, J. C., McCaffary, D., Bakermans, J. J. & Behrens, T. E. How to build a cognitive map. *Nature neuroscience* **25.** 1257–1272 (2022).

18. Rescorla, M. Cognitive maps and the language of thought. *The British Journal for the Philosophy of Science* (2009).

19. Anderson, J. *Cognitive Psychology and Its Implications* Ninth edition (Worth Publishers, New York City, Jan. 2020).

20. Gornet, J. & Thomson, M. Automated Construction of Cognitive Maps with Visual Predictive Coding. *Nature Machine Intelligence* **6.** 820–833 (July 2024).

21. Duan, Y. *et al.* RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning (Nov. 2016).

22. Mirowski, P. *et al. Learning to Navigate in Cities Without a Map* in *Advances in Neural Information Processing Systems* **31** (Curran Associates, Inc., 2018).

23. Gupta, S., Davidson, J., Levine, S., Sukthankar, R. & Malik, J. Cognitive Mapping and Planning for Visual Navigation. 10.

24. Yao, S. *et al. ReAct: Synergizing Reasoning and Acting in Language Models* (2026). Pre-published.

25. Wang, L. *et al. Large Action Models: From Inception to Implementation* (2026). Pre-published.

26. Zhang, C. *et al. AppAgent: Multimodal Agents as Smartphone Users* (2026). Pre-published.

27. Nguyen, D. *et al. GUI Agents: A Survey* (2026). Pre-published.

28. Zhang, X., Li, G., Shokouhi, S. & Thein, M.-W. L. Modern Control Meets Machine Learning: A Review and Taxonomy of Synergistic Approaches for Robotics Applications. *Actuators* **15.** 235 (May 2026).

29. Brown, T. B. *et al.* Language Models Are Few-Shot Learners. *arXiv:2005.14165 [cs]* (July 2020).

30. Vaswani, A. *et al. Attention Is All You Need* Aug. 2023.

31. Felicia, G. *et al. From Perception to Action: Spatial AI Agents and World Models* (2026). Pre-published.

32. Maes, L., Lidec, Q. L., Scieur, D., LeCun, Y. & Balestriero, R. *LeWorld-Model: Stable End-to-End Joint-Embedding Predictive Architecture from Pixels* (2026). Pre-published.

33. Yang, S. *World Models as an Intermediary between Agents and the Real World* (2026). Pre-published.

34. Xu, M. *et al. Kinema4D: Kinematic 4D World Modeling for Spatiotemporal Embodied Simulation* (2026). Pre-published.

35. Markowitz, J. E. *et al.* Spontaneous Behaviour Is Structured by Reinforcement without Explicit Reward. *Nature* **614.** 108–117 (Feb. 2, 2023).

36. Lindsey, J. W., Markowitz, J., Gillis, W. F., Datta, S. R. & Litwin-Kumar, A. Dynamics of Striatal Action Selection and Reinforcement Learning. *eLife* **13.** RP101747 (May 8, 2025).

37. Rabin, M. O. & Scott, D. Finite Automata and Their Decision Problems. *IBM Journal of Research and Development* **3.** 114–125 (Apr. 1959).

38. Nerode, A. Linear Automaton Transformations. *Proceedings of the American Mathematical Society* **9.** 541–544 (1958).

39. Crane, K., Weischedel, C. & Wardetzky, M. The Heat Method for Distance Computation. *Communications of the ACM* **60.** 90–99 (Oct. 24, 2017).

40. Sutton, R. S. & Barto, A. G. *Reinforcement Learning, Second Edition: An Introduction* 552 pp. (Bradford Books, Cambridge, Massachusetts, 2018).

Chapter 2

THEORY

The previous chapter established a connection between sensory prediction and the recovery of spatial structure. Specifically, we demonstrated that in a discrete lattice setting, the principle of path-dependent persistence ensures that if every observation sequence is unique for a path of length T, then predictive coding—predicting future observations from past observations—is sufficient to recover the underlying geometry of the space. By minimizing the discrepancy between predicted and actual observations, an agent implicitly constructs an internal representation that is a graph isomorphism of its environment.

However, this initial result is constrained to the discrete lattice setting, where states and transitions are clearly defined on a grid. While this provides a motivating example, the environments encountered by biological and artificial agents are often continuous and topologically complex. In this chapter, we extend our theoretical framework to the general case of any topological space. We move beyond the discrete graph setting to show that the requirements of predictive coding naturally recover the manifold structure of the world, even in the presence of continuous dynamics and sensory manifolds.

The chapter is organized as follows. First, we provide a brief background on algebraic topology, introducing the concepts of the fundamental group and covering spaces that form the backbone of our theoretical extension. Second, we formally extend the discrete path principle to any topological space, proving that the uniqueness of path-generated data sequences necessitates a homeomorphic recovery of the base manifold. Third, we provide a statistical implementation of predictive coding through maximum-likelihood estimation, bridging the gap between abstract topology and practical inference. Finally, we demonstrate empirically that a predictive coding neural network trained on high-dimensional sensory streams can indeed recover the underlying spatial metric and topology of its environment, validating our theory through computational experiments.

2.1 Preliminaries

In this section, we will cover the algebraic topology concepts necessary to prove the *predictive coding mapping theorem*. We want to provide the minimum amount of background required. As such, many proofs will be given an informal explanation or provided with references to the formal proof.

Topological spaces

A topology on a set $\mathcal{S}$ is a collection $\mathcal{T}$ of subsets of $\mathcal{S}$ that satisfies the following three axioms:

1. The empty set $\emptyset$ and the entire set $\mathcal{S}$ are in $\mathcal{T}$.
2. The union of any arbitrary subcollection of sets in $\mathcal{T}$ is also in $\mathcal{T}$.
3. The intersection of any finite subcollection of sets in $\mathcal{T}$ is also in $\mathcal{T}$.

A set $\mathcal{S}$ equipped with a topology $\mathcal{T}$ is called a topological space, $(\mathcal{S}, \mathcal{T})$. The elements of the collection $\mathcal{T}$ are referred to as the open sets of the space. A function $f : X \to Y$ is continuous if it takes nearby points to nearby points. Formally, $f^{-1}(U)$ is open if U is open.

Intuitively, a topology serves as a formal axiomatization of a neighborhood, providing a qualitative framework for defining closeness or proximity without the need for a numerical distance metric. While metric spaces rely on a distance function to define balls of radius ϵ, a topological space generalizes this by explicitly declaring which sets are "open." Any open set $U \in \mathcal{T}$ containing a point $s \in \mathcal{S}$ can be thought of as a neighborhood of s, encompassing all points that are "locally near" s in a way that respects the space's structure. This abstraction is powerful because it allows us to define continuous functions, convergence, and connectivity in environments that may not have a natural Euclidean distance, such as discrete state graphs or complex latent manifolds.

Paths and homotopy

In the context of navigation and movement, we are particularly interested in how an agent moves from one point to another. We model this as a *path*, which is simply a continuous mapping from a standardized time interval into our space. This represents a trajectory where the agent's position changes without any sudden jumps.

Definition 1. *A* path *in a topological space X is a continuous map $\gamma : [0,1] \to X$. The points $\gamma(0)$ and $\gamma(1)$ are called the* start point *and* end point *of the path, respectively.*

Finally, we need a way to determine if two paths or maps are essentially the same. This is where the concept of *homotopy* arises. Intuitively, two maps are homotopic if one can be continuously deformed into the other without breaking it or leaving the space. This is like imagining a rubber band being stretched or moved from one configuration to another; if you can slide it smoothly between two states, those states are homotopic.

Definition 2. *A* homotopy *between two continuous maps $f, g : X \to Y$ is a continuous map $H : X \times [0,1] \to Y$ such that $H(x,0) = f(x)$ and $H(x,1) = g(x)$ for all $x \in X$. If such a map exists, we say f and g are* homotopic*, denoted $f \simeq g$.*

For paths, we often require the deformation to keep the endpoints fixed. This allows us to group paths into "homotopy classes," which represent different ways of navigating between the same two locations that are topologically equivalent (e.g., they don't wrap around different obstacles).

Definition 3. *Two paths $\gamma_0, \gamma_1 : [0,1] \to X$ with the same endpoints ($x_0 = \gamma_0(0) = \gamma_1(0)$ and $x_1 = \gamma_0(1) = \gamma_1(1)$) are* homotopic relative to endpoints *if there exists a homotopy $H : [0,1] \times [0,1] \to X$ such that:*

1. *$H(s,0) = \gamma_0(s)$ and $H(s,1) = \gamma_1(s)$ for all $s \in [0,1]$.*
2. *$H(0,t) = x_0$ and $H(1,t) = x_1$ for all $t \in [0,1]$.*

Homotopy categories and homotopy equivalences

In the previous section, we introduced homotopy as a way to continuously deform one map into another. If we decide that two homotopic maps are essentially the same for our purposes, we can group them into sets called *homotopy classes*. This allows us to abstract away the specific details of a continuous function and focus on its topological behavior. When we organize topological spaces and these homotopy classes into a algebraic structure, we obtain the *homotopy category*.

Definition 4. *Let X and Y be topological spaces. The set of* homotopy classes *of continuous maps from X to Y is denoted by $[X, Y]$. An element of $[X, Y]$ is an equivalence class $[f]$ containing all maps homotopic to f.*

In a standard category of topological spaces, the morphisms are continuous maps. However, in the homotopy category, we treat all maps within the same homotopy class as a single morphism. The quotienting by the homotopy relation simplifies the study of spaces by identifying maps that can be continuously warped into one another.

Definition 5. *The* homotopy category*, denoted* hTop*, is the category where:*

1. *The* objects *are topological spaces.*
2. *The* morphisms *between X and Y are the homotopy classes of continuous maps, i.e., $Hom_{\mathsf{hTop}}(X, Y) = [X, Y]$.*
3. *The* composition *of morphisms $[f] \in [X, Y]$ and $[g] \in [Y, Z]$ is defined by $[g \circ f]$, which is well-defined because if $f \simeq f'$ and $g \simeq g'$, then $g \circ f \simeq g' \circ f'$.*

With this category in hand, we can define a more flexible notion of sameness between spaces than homeomorphism. While a homeomorphism requires a perfect, reversible mapping between every point of two spaces (like a square and a circle), a *homotopy equivalence* only requires that the spaces can be deformed into one another. For example, a solid disk and a single point are homotopy equivalent because the disk can be "shrunk" to the point, even though they have different dimensions.

Definition 6. *Two topological spaces X and Y are* homotopy equivalent*, denoted $X \simeq Y$, if there exist continuous maps $f : X \to Y$ and $g : Y \to X$ such that:*

1. *$g \circ f \simeq id_X$ (the composition is homotopic to the identity on X).*
2. *$f \circ g \simeq id_Y$ (the composition is homotopic to the identity on Y).*

The map f is called a homotopy equivalence*, and g is its* homotopy inverse.

This concept is powerful because it allows us to simplify complex environments into their topological skeletons. A space that can be shrunk down to a single point is called *contractible*, representing the simplest possible topology where every loop and path can be collapsed.

Definition 7. *A topological space X is* contractible *if it is homotopy equivalent to a point space $\{*\}$. This is equivalent to saying that the identity map id_X is homotopic to a constant map.*

The fundamental group

While homotopy gives us a way to deform paths, the *fundamental group* provides a way to count and categorize "holes" in a space. To do this, we focus on paths that start and end at the same point, which we call *loops*. If a space has a hole, some loops will be able to shrink to a point, while others will be stuck around the hole. The fundamental group collects these loops into an algebraic structure that allows us to perform arithmetic with the ways we can navigate around obstacles.

Definition 8. *A* loop *in a topological space X based at $x_0 \in X$ is a path $\gamma : [0,1] \to X$ such that $\gamma(0) = \gamma(1) = x_0$. The point x_0 is called the* base point*.*

To turn these loops into a group, we need an operation to combine them. The natural choice is *path concatenation*: if you follow one loop and then another, you have completed a larger loop. When we perform this operation on homotopy classes of loops, we find that it satisfies the rules of a mathematical group.

Definition 9. *The* fundamental group *of a topological space X with base point x_0, denoted $\pi_1(X, x_0)$, is the set of all homotopy classes of loops based at x_0. The group operation is defined by the concatenation of classes:*

$$[\gamma] \cdot [\eta] = [\gamma * \eta]$$

where $(\gamma * \eta)(s)$ is the path that traverses γ for $s \in [0, 1/2]$ and η for $s \in [1/2, 1]$.

For this to be a group, it must satisfy three key properties:

1. *Associativity:* Following $(\gamma * \eta)$ then ω is homotopic to following γ then $(\eta * \omega)$.

2. *Identity:* There exists a identity loop (the constant map c_{x_0}) that doesn't change any other loop when concatenated.

3. *Inverse:* For every loop, there is a reverse loop (traversing the same path in the opposite direction) such that their combination can be shrunk back to the base point.

Theorem 2.1.1. *The set $\pi_1(X, x_0)$ with the operation of path concatenation forms a group.*

> ***Proof.*** see Dieck (2008) p. 42[41] ■

One might wonder if the choice of base point x_0 fundamentally changes the group. In a space where you can travel from any point to any other point (a path-connected space), the flavor of the loops is the same regardless of where you start. Any two base points will yield isomorphic groups, meaning the fundamental group is a property of the space's topology rather than its specific coordinates.

Proposition 3. *If X is a path-connected topological space, then for any two points $x_0, x_1 \in X$, the groups $\pi_1(X, x_0)$ and $\pi_1(X, x_1)$ are isomorphic.*

> ***Proof.*** see Dieck (2008) p. 44[41] ■

This allows us to speak of *the* fundamental group of a path-connected space, denoted simply as $\pi_1(X)$, which acts as a powerful invariant for distinguishing between different types of environments. For example, the fundamental group of a simple room is trivial (all loops shrink), while the fundamental group of a circular corridor is the set of integers $\mathbb{Z}$, representing how many times a loop wraps around the circle.

Covering spaces

Sometimes, to understand the global structure of a space, it is easier to focus on a local neighborhood: tracking the local neighborhood as we move around the space. In the context of a physical environment, we can think about the observations within a neighborhood and track those observations as we move around the space. By focusing on a local neighborhood, we avoid two distant positions giving the same observations—as we only track the local

neighborhood rather than the global space. The observation space can be seen as a *covering space* with the (local) inverse observation map $I^{-1}(U)$ as the *covering map*.

Definition 10. *A* covering space *of a topological space X is a space $\tilde{X}$ together with a continuous map $p : \tilde{X} \to X$, called the* covering map*, such that every point $x \in X$ has an open neighborhood U (an* evenly covered neighborhood*) for which the preimage $p^{-1}(U)$ is a disjoint union of open sets in $\tilde{X}$, each of which is mapped homeomorphically onto U by p.*

The most famous example of a covering space is the real line $\mathbb{R}$ covering the unit circle S^1. If you imagine the real line as an infinite piece of string, you can wrap it infinitely many times around the circle. Locally, a small segment of the string looks exactly like a small arc of the circle. However, globally, the real line is flat and has no holes, while the circle wraps back on itself.

Example 1. *The map $p : \mathbb{R} \to S^1$ defined by $p(t) = (\cos(2\pi t), \sin(2\pi t))$ is a covering map. Each point on the circle is covered by infinitely many points on the real line, corresponding to each "lap" around the circle $(t, t+1, t+2, \dots)$.*

Because a covering map is a *local homeomorphism*, it preserves all the local properties of the space—like adjacency and connectivity—but it unfolds the global topological obstacles. This makes covering spaces an essential tool for paths: if an agent knows how to navigate in the unrolled covering space, it can effectively navigate the more complex base space by following the projection of its trajectory.

Proposition 4. *A covering map $p : \tilde{X} \to X$ is a local homeomorphism. For every $\tilde{x} \in \tilde{X}$, there exists a neighborhood V of $\tilde{x}$ such that the restriction $p|_V : V \to p(V)$ is a homeomorphism.*

> ***Proof.*** see Dieck (2008) p. 64[41] ■

The unique path lifting property

The power of covering spaces lies in their ability to track movement from the base space. Imagine walking along a path on a circle while a shadow of you walks on the real line directly above you. If you know exactly where your shadow starts, there is only one possible path your shadow can take to stay

synchronized with your movement on the circle. This process of moving from the base space to the unrolled space is called *lifting*.

Definition 11. *Let $p : \tilde{X} \to X$ be a covering map. A* lift *of a map $f : Y \to X$ is a continuous map $\tilde{f} : Y \to \tilde{X}$ such that $p \circ \tilde{f} = f$. In our context, we are primarily interested in the case where $Y = [0, 1]$, representing the lifting of a path.*

The *Unique Path Lifting Theorem* is the topological guarantee that paths are well-defined. It tells us that for any trajectory an agent takes in the physical world, there is exactly one corresponding trajectory in the covering space, provided we know the starting point. This uniqueness is what allows an agent to maintain a consistent internal estimate of its position.

Theorem 2.1.2 (Unique Path Lifting Theorem)**.** *Let $p : \tilde{X} \to X$ be a covering map, and let $\gamma : [0, 1] \to X$ be a path starting at $x_0 \in X$. For any point $\tilde{x}_0 \in \tilde{X}$ such that $p(\tilde{x}_0) = x_0$, there exists a unique path $\tilde{\gamma} : [0, 1] \to \tilde{X}$ such that $\tilde{\gamma}(0) = \tilde{x}_0$ and $\tilde{\gamma}$ is a lift of γ.*

> ***Proof.*** see Dieck (2008) p. 63[41], May (1999) p. 22[42], and Hatcher (2002) p. 60[43]. ■

This property extends beyond single paths to entire deformations. If you can smoothly warp one path into another in the base space, the unique path lifting ensures that their corresponding lifts warp into each other just as smoothly. This is known as the *Homotopy Lifting Property*. It implies that topologically equivalent paths in the environment will always result in equivalent movements in the internal representation.

Theorem 2.1.3 (Homotopy Lifting Theorem)**.** *Let $p : \tilde{X} \to X$ be a covering map. Given a homotopy $H : [0, 1] \times [0, 1] \to X$ and a lift $\tilde{\gamma}_0$ of the initial path $\gamma_0(s) = H(s, 0)$, there exists a unique homotopy $\tilde{H} : [0, 1] \times [0, 1] \to \tilde{X}$ such that $p \circ \tilde{H} = H$ and $\tilde{H}(s, 0) = \tilde{\gamma}_0(s)$.*

> ***Proof.*** see Dieck (2008) p. 66[41]. ■

In the framework of predictive coding, the observation space acts as a covering space for the environment. The unique path lifting property ensures that an

agent can uniquely resolve visual ambiguity—where different locations look the same—by leveraging the history of its movement. As long as the agent's path is uniquely lifted, its internal map remains a globally consistent representation of the world, effectively stitching together local sensory experiences into a coherent topological manifold.

2.2 Environments as a state-transition system

The Agent-Environment Interface

The foundation of predictive coding is the closed-loop interaction between an agent and its external environment. This interaction is formalized as a sequence of discrete time steps $t \in \mathbb{R}^+$. Crucially, the agent does not have direct access to the environment's true state $s_t \in \mathcal{S}$; rather, the underlying state is a unobserved (or latent) variable. At each time step t, the agent receives a sensory observation $o_t \in \mathcal{O}$ that provides a partial and often ambiguous representation of the environment. On the basis of its history of observations, the agent selects an action $a_t \in \mathcal{A}$. As a consequence of its action, the environment transitions to a new hidden state $s_{t'}$ and generates a new observation $o_{t'}$.

This framework is universal: the agent represents the decision-maker, while the environment represents everything outside the agent's direct control, including its own physical body in robotic applications. The boundary between agent and environment is defined by the limit of the agent's absolute control, not by physical constraints.

Locality condition

A critical assumption in this framework is that the environment's dynamics are governed by a deterministic transition function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. This function maps the current hidden state s_t and the agent's action a_t to the subsequent state $s_{t'}$:

$$s_{t+1} = f(s_t, a_t) \tag{2.1}$$

This deterministic mapping implies that the current state s_t encapsulates all the necessary information to determine the next state, assuming the action is known. In the context of spatial navigation, this transition function reflects the physical laws and geometric constraints of the environment, where a movement action results in a predictable change in the agent's position within the topological space.

An environments is a partially observable deterministic state-transition system

A partially observable deterministic state-transition system provides a precise mathematical definition of an environment. A partially observable deterministic state-transition system is defined as the 5-tuple $(\mathcal{S}, \mathcal{A}, f, \mathcal{O}, I)$:

1. $\mathcal{S}$ is the topology of hidden states.
2. $\mathcal{A}$ is the topology of actions.
3. $f : \mathcal{A} \times \mathcal{S} \to \mathcal{S}$ is the transition function, $f(a, s) = s'$.
4. $\mathcal{O}$ is the topology of observations.
5. $I : \mathcal{S} \to \mathcal{O}$ is the observation function $\phi(s) = o$.

To ground these abstract definitions, consider a Grid World environment. The state space $\mathcal{S}$ consists of 16 cells in a 4×4 grid, indexed from $(0, 0)$ to $(3, 3)$. The action space $\mathcal{A}$ contains four discrete movements: {North, South, East, West}. Certain cells are "Walls" that the agent cannot enter. If the agent takes an action that would lead into a wall, the transition probability $f(a, s) = s$, meaning the agent stays in its current cell. The agent observe the visual scene at each state. However, some states may have the same visual scene, which introduces ambiguity in the agent's current state.

2.3 The predictive coding mapping theorem

In this section, we will prove the *predictive coding mapping theorem*, which states that if two spaces generated the same set of unique observation sequences then the two spaces are homeomorphic. Before we prove the theorem, let us provide some motivation for the theorem. In Chapter I, we showed that if the observation sequences of two different path spaces were unique and identical then the underlying grid lattice must be the same, i.e., a graph isomorphism. While any particular observation may not be bijective, the observation sequences were bijective we could label any position by taking the terminal endpoints of each observation sequence. To prove the general case, the idea is quite similar: we use the terminal endpoints to label the position within a topological space. The condition that "the observation sequences are the same" translates to the loops $p_M(\pi_1(M, m_0))$ and $p_N(\pi_1(N, n_0))$ in both

spaces M and N generate the same lifted paths: the induced subgroups in the fundamental group of X are identical

$$(p_M)_*(\pi_1(M, m_0)) = (p_N)_*(\pi_1(N, n_0)).$$

From this translated condition, we can prove the homeomorphism in a similar manner to the graph isomorphism.

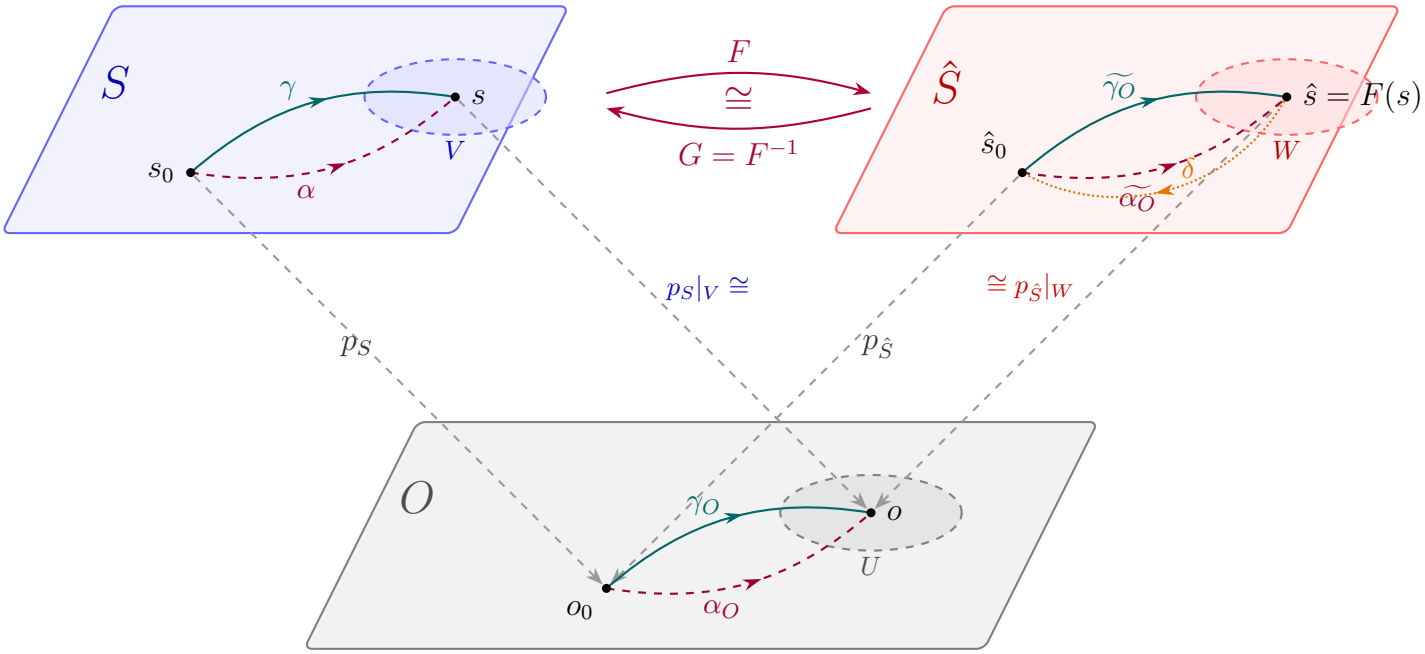


Figure 2.1: **Proof of the predictive coding mapping theorem** By hypothesis, $(p_S)_*\pi_1(S, s_0) = (p_{\hat{S}})_*\pi_1(\hat{S}, \hat{s}_0)$, so this loop lifts to a **closed loop** in $\hat{S}$. Thus, the unique path lifts $\widetilde{\gamma_O}$ and $\widetilde{\alpha_O}$ must end at the exact same point: $\widetilde{\alpha_O}(1) = \widetilde{\gamma_O}(1) = F(s)$, making F unconditionally well-defined. Locally, $F|_V = (p_{\hat{S}}|_W)^{-1} \circ (p_S|_V)$ establishes continuity, making F a homeomorphism.

Theorem 2.3.1 (Predictive coding mapping theorem)**.** *Let O be connected, locally path-connected topological spaces, and let $S, \hat{S}$ be connected topological spaces. Let $p_S : S \to O$ and $p_{\hat{S}} : \hat{S} \to O$ be covering spaces with basepoints $s_0 \in S$, $\hat{s}_0 \in \hat{S}$, and $o_0 \in O$ such that $p_S(s_0) = p_{\hat{S}}(\hat{s}_0) = o_0$. If the induced subgroups in the fundamental group of O are identical, such that:*

$$(p_S)_*(\pi_1(S, s_0)) = (p_{\hat{S}})_*(\pi_1(\hat{S}, \hat{s}_0))$$

then the spaces S and $\hat{S}$ are homeomorphic.

Proof. For $s \in S$, choose a path $\gamma : [0,1] \to S$ from s_0 to s. Let $\gamma_O = p_S \circ \gamma$, a path in O from o_0. Since $p_{\hat{S}}$ is a covering map, γ_O has a unique lift $\widetilde{\gamma_O} : [0,1] \to \hat{S}$ with $\widetilde{\gamma_O}(0) = \hat{s}_0$. Define

$$F(s) = \widetilde{\gamma_O}(1).$$

Let $\alpha : [0,1] \to S$ be another path from s_0 to s, with $\alpha_O = p_S \circ \alpha$ and lift $\widetilde{\alpha_O}$ starting at $\hat{s}_0$. We must show $\widetilde{\gamma_O}(1) = \widetilde{\alpha_O}(1)$.

The concatenation $\gamma * \overline{\alpha}$—where $\overline{\alpha}$ is α reversed—is a loop at s_0, so $[\gamma * \overline{\alpha}] \in \pi_1(S, s_0)$. Applying p_S,

$$[\gamma_O * \overline{\alpha_O}] = (p_S)_*[\gamma * \overline{\alpha}] \in (p_S)_*\pi_1(S, s_0) = (p_{\hat{S}})_*\pi_1(\hat{S}, \hat{s}_0).$$

By the lifting criterion, a loop in O based at o_0 lifts to a loop in $\hat{S}$ based at $\hat{s}_0$ if and only if its class lies in $(p_{\hat{S}})_*\pi_1(\hat{S}, \hat{s}_0)$. Hence the lift $\widetilde{\gamma_O * \overline{\alpha_O}}$ of $\gamma_O * \overline{\alpha_O}$ starting at $\hat{s}_0$ is closed.

By uniqueness of lifts, this lift restricted to the first half $[0, \frac{1}{2}]$ is exactly $\widetilde{\gamma_O}$, ending at $\widetilde{\gamma_O}(1)$. The second half is the lift of $\overline{\alpha_O}$ beginning at $\widetilde{\gamma_O}(1)$; call it δ. Because the whole lift is closed, $\delta(1) = \hat{s}_0$.

Now consider $\overline{\delta}$, the reversal of δ: it is a lift of $\overline{\overline{\alpha_O}} = \alpha_O$ starting at $\overline{\delta}(0) = \delta(1) = \hat{s}_0$ and ending at $\overline{\delta}(1) = \delta(0) = \widetilde{\gamma_O}(1)$. But $\widetilde{\alpha_O}$ is the lift of α_O starting at $\hat{s}_0$, so by uniqueness $\overline{\delta} = \widetilde{\alpha_O}$. By construction, $p_{\hat{S}} \circ F = p_S$. Because $p_{\hat{S}}(\widetilde{\gamma_O}(1)) = \gamma_O(1) = p_S(s)$, F is a map over O. Therefore

$$\widetilde{\alpha_O}(1) = \overline{\delta}(1) = \widetilde{\gamma_O}(1),$$

proving F is well-defined.

By symmetry, because $(p_{\hat{S}})_*\pi_1(\hat{S}, \hat{s}_0) = (p_S)_*\pi_1(S, s_0)$, $\hat{S}$ is path-connected, and S locally path-connected, the same construction yields a well-defined map $G : \hat{S} \to S$ over O with $G(\hat{s}_0) = s_0$.

We will now show that $G \circ F = \mathrm{id}_S$. Fix s and a path γ from s_0 to s. By construction $F(s) = \widetilde{\gamma_O}(1)$, and $\widetilde{\gamma_O}$ is a path in $\hat{S}$ from $\hat{s}_0$ to $F(s)$ with $p_{\hat{S}} \circ \widetilde{\gamma_O} = \gamma_O = p_S \circ \gamma$. To compute $G(F(s))$ we may use this path $\widetilde{\gamma_O}$:

its O-projection is γ_O, whose lift to S starting at s_0 is, by uniqueness, γ itself (since $p_S \circ \gamma = \gamma_O$ and $\gamma(0) = s_0$). Hence $G(F(s)) = \gamma(1) = s$. Symmetrically $F \circ G = \mathrm{id}_{\hat{S}}$. Thus F is a bijection with $F^{-1} = G$.

We will now show that F is continuous. Let $s \in S$ and put $o = p_S(s)$, $\hat{s} = F(s)$. Since O is locally path-connected and p_S, $p_{\hat{S}}$ are covering maps, choose a path-connected open neighborhood $U \subseteq O$ of o that is evenly covered by both p_S and $p_{\hat{S}}$. Let $V \subseteq S$ be the sheet over U containing s and $W \subseteq \hat{S}$ the sheet over U containing $\hat{s}$; then $p_S|_V : V \to U$ and $p_{\hat{S}}|_W : W \to U$ are homeomorphisms. For any $s' \in V$, form a path to s' by following a fixed path γ to s and then a path inside V from s to s'; projecting and lifting, the lift stays in the sheet W (by uniqueness and path-connectedness of U). Consequently

$$F|_V = (p_{\hat{S}}|_W)^{-1} \circ (p_S|_V),$$

a composition of homeomorphisms, hence continuous on V. Since such V cover S, F is continuous. The same argument applied to G shows G is continuous.

F is a continuous bijection with continuous inverse G, and $p_{\hat{S}} \circ F = p_S$. Therefore F is a homeomorphism $S \xrightarrow{\sim} \hat{S}$ commuting with the projections to O, an isomorphism of covering spaces. ■

2.4 Recovering the underlying space using predictive coding

Throughout this chapter, we have referred to *predictive coding* without definition. In this section, we will define predictive coding and show that predictive coding generates the observation sequences if the observation sequences are generated from an environment (2.2).

Although the observation sequences are deterministic, it is often convenient to view them as stochastic. An analogous example is information theory, in which we treat messages as stochastic despite no messages ever being random[44]. The underlying motivation is that we can study the *distributions* of sequences rather than a single sequence.

Formally, an observation sequence $O = (o_1, \ldots, o_T)$ is a sequence of random

variables such that

$$\begin{aligned} p(O) &= p(o_1, \ldots, o_T) \\ &= \prod_{t=1}^{T} p(o_t | o_{<t}) \end{aligned}$$

where $p(o_t)$ is the probability density for the random variable o_t. We define the conditional probability $p(o_t|o_{<t})$ as the *predictive code*. We define the estimation of the conditional probability $p(o_t|o_{<t})$ as *predictive coding*.

In the context of an environment (2.2), the conditional probability is given by

$$p(o_{k+1}|o_1, \ldots, o_k) \tag{2.2}$$

$$= \int_{\Omega} \mathbf{dx} \underbrace{p(x_0, \ldots, s_k|o_0, \ldots, o_k)}_{\text{encoding}} \underbrace{p(s_{k+1}|s_k)}_{\text{transition probability}} \underbrace{p(o_{k+1}|s_{k+1})}_{\text{decoding}} \tag{2.3}$$

where $\mathbf{dx} = dx_0 \ldots ds_k$. This formulation decomposes the predictive inference into three distinct stages:

1. *Encoding:* Estimating the likelihood of the agent's historical path given the observed image sequence.
2. *State Transition:* Predicting the agent's next position on its transition model within the environment.
3. *Decoding:* Mapping the inferred future state back into the observation space to generate a prediction of the next image.

We consider an agent positioned at x, where $x \in \mathcal{S}$ represents spatial coordinates. The agent's sensory experience is defined by a conditional probability distribution $p(O|x)$, which assigns a likelihood to observing a specific observation $O \in \mathcal{O}$ at a given state. This distribution captures the deterministic structure of the environment. From the deterministic perspective, $p(O|x)$ can be viewed as the (inverse) covering map on on the covering space where the base space is the configuration space of the agent and the covering space represents the space of possible observations $\mathcal{O}$.

As the agent explores the environment along a sequence of points $x_0, \ldots, s_k$ and collects a corresponding sequence of observations $O_0, \ldots, O_k$, the core predictive coding task is to estimate the next observation O_{k+1}. Theoretically, this is solved by maximizing the posterior distribution $p(O_{k+1}|O_0, \ldots, O_k)$.

2.5 Estimating the predictive coding distribution using maximum-likelihood estimation

In the predictive coding's stochastic formulation, we can estimate the predictive code using a statistical model. Let $p_\theta(o_t|o_{<t})$ be a statistical model with parameters θ for observations $o_1, \ldots, o_t$.

We can express the model distribution $p_\theta(o_t|o_{<t})$ as

$$\begin{aligned} p_\theta(o_t|o_{<t}) &= \int p_\theta(o_t, s_t, s_{<t}|o_{<t})\ ds_t ds_{<t} \\ &= \int p_\theta(o_t|s_t) p_\theta(s_t|s_{<t}) p_\theta(s_{<t}|o_{<t})\ ds_t ds_{<t} \\ &= \mathbb{E}_{x\sim p_\theta(s_t|o_{<t})}\left[p_\theta(o_t|s_t)\right] \end{aligned}$$

Performing maximum likelihood estimation,

$$\begin{aligned} &\arg\max_\theta \mathbb{E}_{o\sim p_{\text{data}}(o)} p_\theta(o_1, \ldots, o_T) \\ &\quad = \arg\max_\theta \sum_{t=1}^{T} \mathbb{E}_{o\sim p_{\text{data}}(o)} \left[\mathbb{E}_{s_t\sim p_\theta(s_t|o_t)} p_\theta(o_t|s_t)\right] \end{aligned}$$

As log is a monotonic, increasing function, we can take perform maximum *log-likelihood* estimation,

$$\begin{aligned} &\arg\max_\theta \mathbb{E}_{o\sim p_{\text{data}}(o)} p_\theta(o_1, \ldots, o_T) \\ &\quad = \arg\max_\theta \mathbb{E}_{o\sim p_{\text{data}}(o)} \log p_\theta(o_1, \ldots, o_T) \\ &\quad = \arg\max_\theta \sum_{t=1}^{T} \mathbb{E}_{o\sim p_{\text{data}}(o)} \mathbb{E}_{s_t\sim p_\theta(s_t|o_t)} \left[\log p_\theta(o_t|s_t)\right] \end{aligned}$$

Predictive coding, which is solving for $p_\theta(o_t|s_t)$ and $p_\theta(s_t|o_{<t})$, is equivalent to estimating the data-generating distribution $p_\theta(o_1, \ldots, o_T)$.

Because an environment is deterministic, the transition probability as

$$p(s_{k+1}|s_k) = \mathbb{1}[f(a, s_k) = s_k + 1]$$

for the environment's transition function $f : \mathcal{A} \times \mathcal{S} \to \mathcal{S}$ and

$$p(O_k|s_k) = \mathbb{1}[\phi(s_k) = o_k]$$

for the observation function $I : \mathcal{S} \to \mathcal{O}$ given the Dirac measure $\mathbb{1}[\cdot]$. We can then parameterize the model distributions

$$p_\theta(s_t|s_{<t}) = \mathbb{1}[f_\theta(s_{<t}) = s_t]$$

and

$$p_\theta(s_{<t}|o_{<t}) = \mathbb{1}[F_\theta(o_{<t}) = s_{<t}]$$

with functions $o_{<t} \overset{f_\theta}{\longmapsto} s_{<t}$ and $s_{<t} \overset{F_\theta}{\longmapsto} s_t$. Because every state has a single observation, we can parameterize the distribution $p_\theta(o_t|s_t)$ as

$$p_\theta(o_t|s_t) = \mathcal{N}(o_t; g_\theta(s_t), \omega^2\mathbf{I}),$$

then maximum likelihood estimation becomes

$$\begin{aligned}
&\arg\max_\theta \mathbb{E}_{o\sim p_{\text{data}}(o)} p_\theta(o_1, \ldots, o_T) \\
&= \arg\max_\theta \sum_{t=1}^{T} \mathbb{E}_{o\sim p_{\text{data}}(o)} \mathbb{E}_{s_t\sim p_\theta(s_t|o_t)} \left[\log p_\theta(o_t|s_t)\right] \\
&= \arg\max_\theta \sum_{t=1}^{T} \mathbb{E}_{o\sim p_{\text{data}}(o)} \left[\log p_\theta(o_t|s_t = F_\theta \circ f_\theta(o_{<t}))\right] \\
&= \arg\min_\theta \mathbb{E}_{o\sim p_{\text{data}}(o)} \sum_{t=1}^{T} \left\|o_t - g_\theta \circ F_\theta \circ f_\theta(o_{<t})\right\|_{\ell_2}^2 .
\end{aligned}$$

2.6 A neural network can recover an environment's map using predictive coding

The theoretical framework of predictive coding as a path integral over environmental trajectories can be directly implemented using deep neural networks. In this section, we will show that a neural network trained to predict future sensory observations can internally construct a representation that recovers the environment's spatial coordinate system and topology. The neural network implementation demonstrates empirically that predictive coding can recover an geometric structure of the environment's space.

Architecture and training of the predictive coder

To demonstrate that a neural network that performs predictive coding can implicitly construct a spatial map, we implemented predictive coding neural

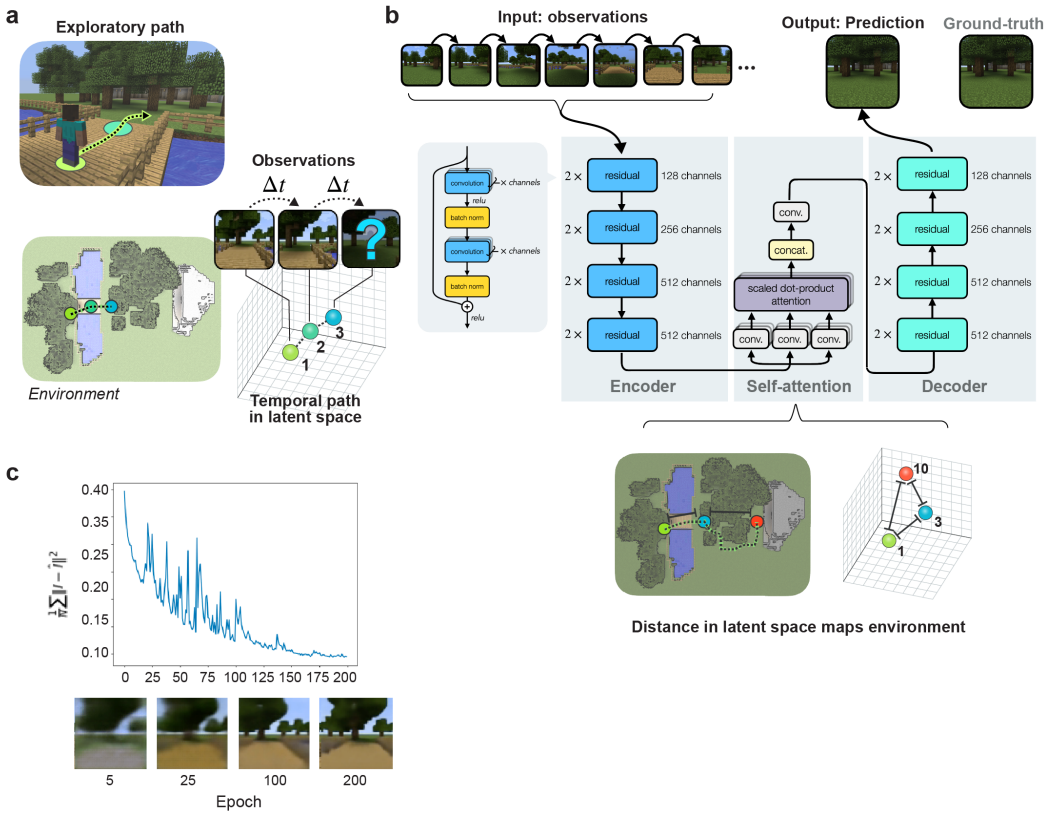


Figure 2.2: **A predictive coding neural network explores a virtual environment.** In predictive coding, a model predicts observations and updates its parameters using the prediction error. **a,** an agent's traverses its environment by taking the most direct path to random positions. **b,** a self-attention-based encoder-decoder neural network architecture learns to perform predictive coding. A ResNet-18 convolutional neural network acts as an encoder; self-attention is performed with 8 heads, and a corresponding ResNet-18 convolutional neural network performing decoding to the predicted image.**c,** the neural network learns to perform predictive coding effectively—with a mean-squared error of 0.094 between the actual and predicted images.

network and analyzed the spatial maps it acquired while navigating a virtual space. Our environment was built using the Minecraft Malmo platform[45], spanning a 40×65 lattice unit area. It incorporates three distinct visual elements: a cave acting as a global landmark, a forest introducing visual degeneracy, and a river with a bridge that restricts agent movement (Figure 2.2(**a**)). The agent traverses paths calculated via A^* search between stochastic locations, collecting visual data throughout its journey (Figure 2.3(**b-c**)).

To implement predictive coding, we designed an encoder-decoder convolutional neural network (CNN) employing a ResNet-18 backbone[46] for the encoder and

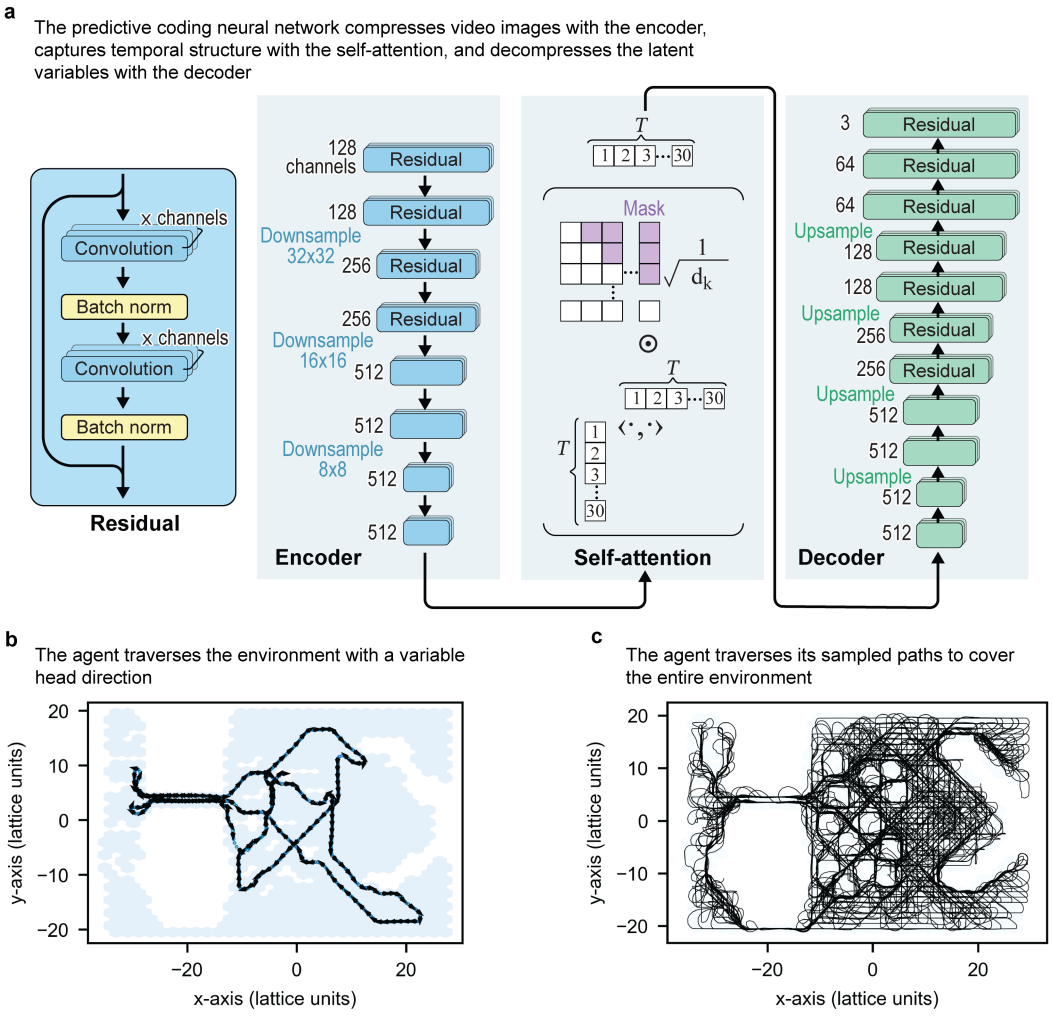


Figure 2.3: **Extended neural network architecture and training description. a,** the predictive coding neural network, or predictive coder, uses an encoder, self-attention, and decoder to perform predictive coding. The encoder is a convolutional neural network architecture called ResNet-18 that uses residuals to compress the high-dimensional video image. The self-attention module capture temporal dependencies from the low-dimensional encoded images. The self-attention's output gives the predictive coder's latent variables. The decoder is a convolutional neural network that upscales—rather than downscales—the predictive coder's latent variables to predicted images. **b,** an example path of the agent shows the agent traversing the environment with a variable head direction. **c,** the agent's paths traverse the space to cover the entire environment.

a symmetric ResNet-18 structure with transposed convolutions for the decoder (Figure 2.2(**b**)). This architecture integrates U-Net-style skip connections[47] to relay latent features to the decoder. A multi-headed attention mechanism[30] with $h = 8$ heads is used to process the sequence of latent units, capturing the history of visual inputs. For latent units of dimension $D = C \times H \times W$, the dimension of each attention head is set to $d = D/h$.

The model is optimized to minimize the mean-squared error (MSE) between actual and predicted sensory observations. Training was conducted on $82,630$ samples over 200 epochs using gradient descent with Nesterov momentum[48], a weight decay of 5×10^{-6}, and an initial learning rate of 10^{-1} managed by a OneCycle scheduler[49]. The resulting predictive coder achieved an MSE of 0.094, demonstrating high visual fidelity in its predictions (Figure 2.2(**c**)).

Predictive coding neural network constructs an implicit spatial map

To validate whether spatial maps are an emergent property of predictive coding, we evaluate whether the latent representations generated by the predictive coder inherently capture the geometric properties of the physical environment, specifically positional information and relative spatial distances.

We demonstrate that the predictive coder builds an implicit spatial map by showing it can reconstruct environmental positions and distances. We utilize the predictive coder's encoder to map image sequences into latent representations for analysis. To quantify the spatial information present, we train a decoder to estimate the agent's location from these latent units (Figure 2.2(**a**)). The prediction error

$$E(x, \hat{x}) = \|\hat{x} - x\|_{\ell_2}$$

provides an indirect assessment of the positional information. Baseline models are constructed for comparison. A lower bound on prediction error is established using true positions perturbed by Gaussian noise

$$\hat{x} = x + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma).$$

We compare the predictive coder's performance against these baselines using error histograms (Figure 2.4(**b**)).

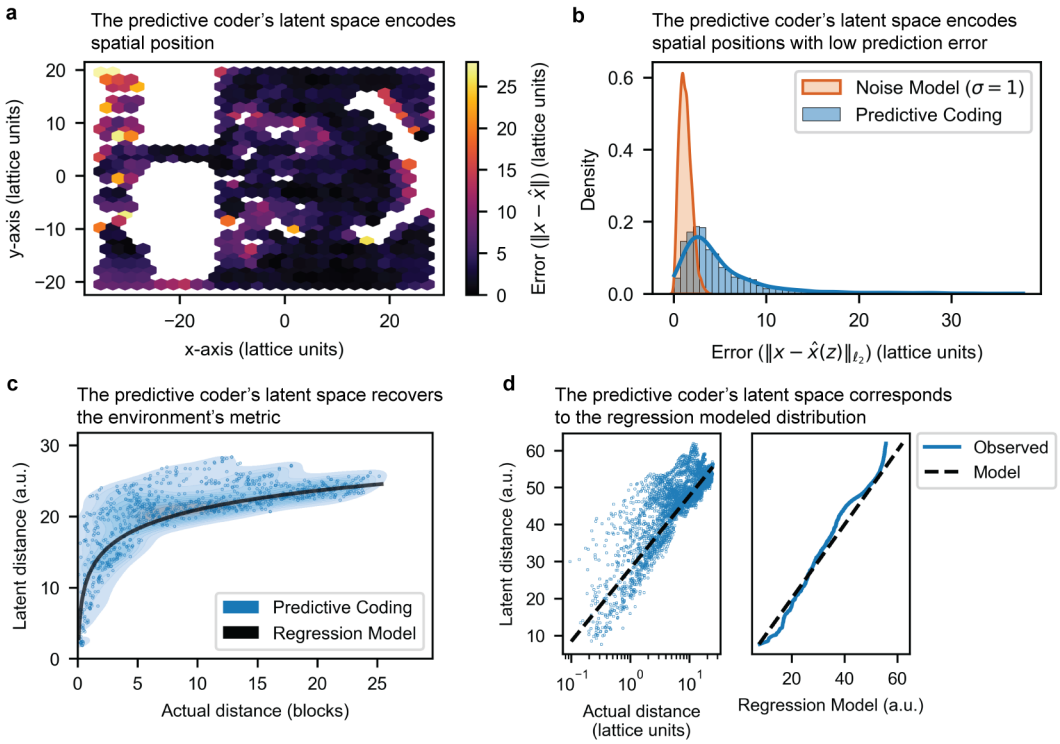


Figure 2.4: **Predictive coding neural network constructs an implicit spatial map.** **a-b,** The predictive coder's latent space encodes accurate spatial positions. A neural network predicts the spatial location from the predictive coding's latent space. **a,** a heatmap of the prediction errors between the actual position and the predictive coder's predicted positions show a low prediction error. **b,** The histogram of prediction errors of positions from the predictive coder's latent space show a low prediction error. As a baseline (Noise model ($\sigma = 1$ lattice unit)), actual positions with a small noise displacement gives an error model. **c,** predictive coding's latent distances recover the environment's spatial metric. Sequential visual images are mapped to the neural network's latent space, and the latent space distances (ℓ_2) are plotted with physical distances onto a joint density plot. An nonlinear regression model

$$\|z - z'\| = \alpha \log \|x - x'\| + \beta$$

is shown as a baseline. **d,** a correlation plot and a quantile-quantile plot show the overlap between the empirical and model distributions.

The predictive coder achieves high precision in encoding spatial location (Figure 2.4.d). It demonstrates a mean error of 5.04 lattice units, with over 80% of samples having an error below 7.3 units. For comparison, the $\sigma = 4$ Gaussian model has a mean error of 4.98 lattice units, with 80% of samples under 7.12 units.

Our analysis shows that the predictive coder's latent space preserves local environmental distances. For each trajectory, we compute pairwise physical and

latent distances within a 100-step window. To evaluate the correspondence, we calculate the joint density of these distances (Figure 2.4(**c**)). We model the latent distances by fitting physical distances (with added noise) to a logarithmic function

$$d(z, z') = \alpha \log(\|x - x' + \epsilon\|) + \beta, \epsilon \sim \mathcal{N}(0, \sigma).$$

This model aligns with the predictive coder's distribution (Figure 2.4(**d**)), yielding a Pearson correlation coefficient of 0.827 and a Kullback-Leibler divergence ($\mathbb{D}_{\mathrm{KL}}(p_{\mathrm{PC}} \| p_{\mathrm{model}})$) of 0.429 bits.

The predictive coding neural network recovers spatial proximity not image similarity

As established in the 2.6, predictive coding enables a neural network to develop an internal spatial representation within its latent space. Here, we argue that the prediction task itself is fundamental to spatial mapping, as it compels the network to prioritize spatial proximity over simple image similarity. Traditional frameworks like PCA, IsoMap[50], and standard autoencoders typically group images based on visual resemblance. While visual and spatial proximity often coincide, similar scenes can be spatially distant. In our virtual environment, for instance, two distinct 'forest' zones are separated by a lake; images from these two zones are more similar to each other than to images of the intervening lake, yet they are spatially further apart (Figure 2.2(**a**)).

To highlight the importance of prediction, we compared the latent representations of the predictive coder to those of an autoencoder. The autoencoder utilizes a similar architecture but processes only a *single* image observation at a time, reconstructs that same image, and thus learns an embedding based strictly on visual proximity. Like the predictive coder, the autoencoder (Figure 2.5(**a**)) was trained on $82,630$ samples over 200 epochs using gradient descent with Nesterov momentum, 5×10^{-6} weight decay, and a 10^{-1} learning rate with OneCycle scheduling. The autoencoder achieved an MSE of 0.039 and high reconstruction fidelity.

The predictive coder generates a more refined and accurate spatial map compared to the autoencoder. When we train an auxiliary network to predict the agent's location from the autoencoder's latent units (Figure 2.5(**b**)), we find that more than 80% of the points have a prediction error exceeding 13.1 lattice

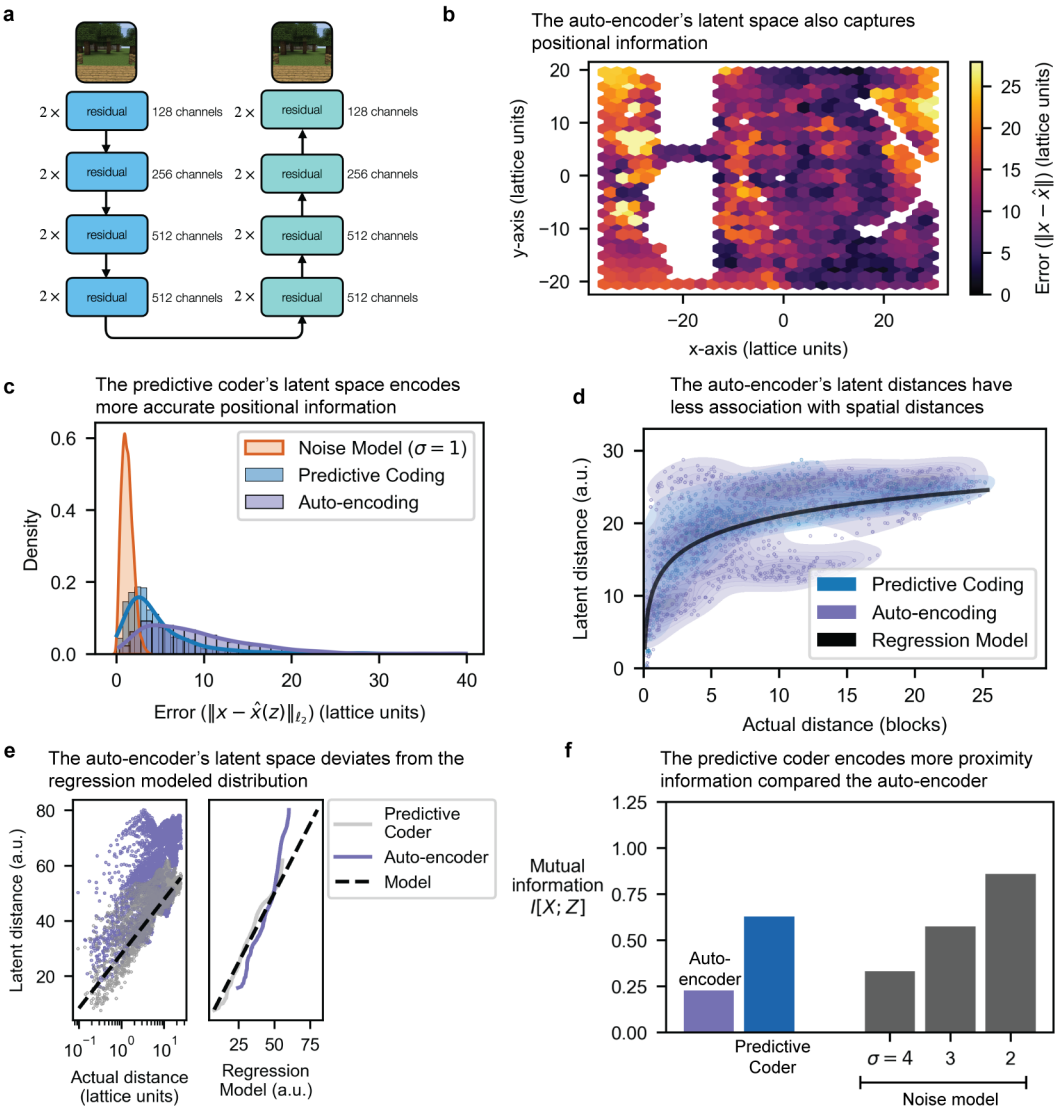


Figure 2.5: **Predictive coding network learns spatial proximity not image similarity** **a,** an autoencoding neural network compresses visual images into a low-dimensional latent vector and reconstructs the image from the latent space. Auto-encoder trains on visual images from the environment *without any sequential order*. **b-c,** auto-encoding encodes lower resolution in positional information. **b,** a neural network predicts the spatial location from the auto-encoding's latent space. A heatmap of the prediction errors between the actual position and the auto-encoder's predicted positions show a higher prediction error—compared to the predictive coder. **c,** auto-encoding captures less positional information compared to predictive coding. The histogram shows the prediction errors of positions from the latent space of both the auto-encoder and the predictive coder.

Figure 2.5: **(continued) d,** latent distances, however, show a weaker relationship with physical distances, as the joint histogram between physical and latent distances is less concentrated. **e,** a correlation plot and a quantile-quantile plot show a lower correlation and a lower density overlap between the empirical and model distributions. **f,** predictive coding's latent units communicate more fine-grained spatial distances whereas auto-encoding communicates broad spatial regions. Joint density plots show the association between latent distances and physical distances for both predictive coding and auto-encoding. Predictive coding's latent distances increase with spatial distances, with a higher concentration compared to auto-encoding.

units, whereas the predictive coder keeps 80% of its samples below 7.3 units (Figure 2.5(**c**)).

We further demonstrate that the predictive coder captures environmental distances with superior resolution. By analyzing the joint density of physical and latent distances (Figure 2.5(**d**)), we observe that while autoencoder latent distances do increase with physical distance, they show much higher dispersion and uncertainty compared to the predictive coder.

This dispersion can be quantified using mutual information (Figure 2.5(**e**)):

$$I[X;Z] = \mathbb{E}_{p(X,Z)}\left[\log \frac{p(X,Z)}{p(X)p(Z)}\right].$$

The autoencoder provides 0.227 bits of mutual information, whereas the predictive coder yields 0.627 bits. For context, a model using true positions with $\sigma = 2$ Gaussian noise gives 0.911 bits. The predictive coder thus offers 0.400 bits of additional distance information over the autoencoder, illustrating that temporal dependencies capture significantly more spatial structure than visual similarity alone.

The predictive coding neural network maps visually ambiguous environments whereas auto-encoding cannot

While sequential prediction is clearly beneficial for mapping accuracy and metric correspondence, we now investigate whether it is *necessary*. We demonstrate that autoencoders fundamentally fail to recover maps in certain environments—specifically those with visual degeneracy, where different locations look identical. We provide both empirical evidence from a circular corridor and a theoretical proof of this limitation.

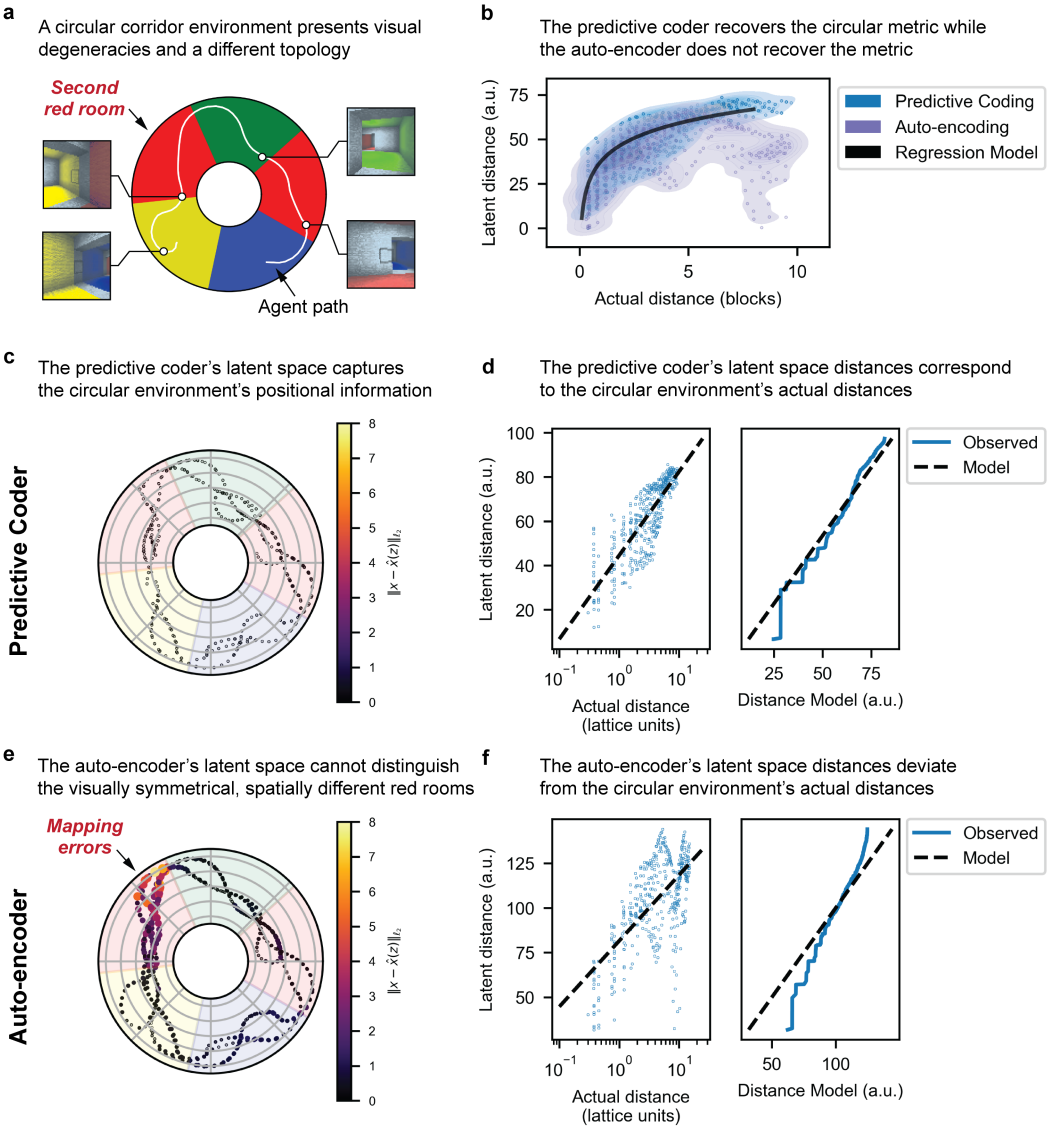


Figure 2.6: **Predictive coding network can learn a circular topology and distinguishes visually identical, spatially different locations** **a,** an agent traverses a circular environment with two visually identical red rooms, which provides visually similar yet spatially different locations. **b,** the predictive coder's latent distances show a correspondence with the circular environment's metric while the auto-encoder's latent distances show little correlation. **c,** similar to Figures 2 and 3, a different neural network measures the predictive coder's spatial information by predicting the agent's location from the predictive coder's latent space. The predictive coder's latent space demonstrates a low prediction error. **d,** similar to Figures 2 and 3, the nonlinear regression measures the correspondence between the latent distances $\|z - z'\|$ and the actual distances $\|x - x'\|$ with the model

$$\|z - z'\| = \alpha \log \|x - x'\| + \beta.$$

Figure 2.6: **(continued) d,** the correlation plot (left) with the nonlinear regression model show a strong correlation between the predictive coder's latent distances and the environment's actual distances ($r = 0.827$). The quantile-quantile plot (right) between the predictive coder's latent distances and the regression model show high overlap ($\mathbb{D}_{\mathsf{KL}}(p_{\mathsf{PC}} \| p_{\mathsf{model}}) = 0.250$). **e,** without any past information, the auto-encoder cannot distinguish the two different red rooms and produces a high prediction error in these locations. **f,** the correlation plot (left) with the nonlinear regression model show little correlation between the auto-encoder's latent distances and the environment's actual distances ($r = 0.288$). The quantile-quantile plot (right) between the auto-encoder's latent distances and the regression model show little overlap ($\mathbb{D}_{\mathsf{KL}}(p_{\mathsf{PC}} \| p_{\mathsf{model}}) = 3.806$).

We introduced a circular environment (Figure 2.6(**a**)) with a specific room sequence (red, green, red, blue, yellow) such that two rooms are *visually identical* but spatially distinct. This setup tests whether these models can handle visual symmetry and whether they learn a global geometric map or merely a relative sequence.

We trained both predictive coding and autoencoding networks on this circular corridor. The autoencoder fails in the presence of visual degeneracy, mapping both red rooms to the same latent location (Figure 2.6(**e**)). Its prediction errors are high in these degenerate regions, even though it performs well in visually distinct areas like the blue and yellow rooms (mean error $\|x - \hat{x}\|_{\ell_2} = 5.004$ lattice units). In contrast, the predictive coder successfully discriminates between the two red rooms (Figure 2.6(**b**)) and maintains low prediction error throughout the entire environment (mean error $\|x - \hat{x}\|_{\ell_2} = 0.071$ lattice units) (Figure 2.6(**c**)).

Quantitatively, we evaluated the relationship between latent and physical metrics using the regression model:

$$\|z - z'\| = \alpha \log \|x - x'\| + \beta.$$

The predictive coder's latent metric closely recovers the spatial metric, showing a high correlation ($r = 0.827$, Figure 2.6(**d**, left)) and high overlap with the model distribution ($\mathbb{D}_{\mathrm{KL}}(p_{\mathrm{PC}} \| p_{\mathrm{model}}) = 0.250$, Figure 2.6(**d**, right)). The autoencoder fails this metric recovery, showing low correlation ($r = 0.288$, Figure 2.6(**f**, left)) and minimal distribution overlap ($\mathbb{D}_{\mathrm{KL}}(p_{\mathrm{PC}} \| p_{\mathrm{model}}) = 3.806$, Figure 2.6(**f, right)**).

Our theoretical analysis confirms that no statistical estimator can recover a map from *stationary* observations in a visually degenerate environment. We present a proof sketch on a lattice environment $X \subset \mathbb{Z}^2$.

Theorem 2.6.1. *Consider an environment $X \subset \mathbb{Z}^2$ where a function $x \overset{f}{\mapsto} I$ maps each position $x \in X$ to an image $I_x = f(x) \in \mathbb{R}^D$. If the environment is degenerate such that $f(x_1) = f(x_2)$ for some $x_1 \neq x_2$, there exists no decoder $I \overset{d}{\mapsto} x$ such that $x = d \circ f(x)$ for all x.*

> ***Proof.*** A function has a left inverse if and only if it is injective. If $f(x_1) = f(x_2) = I$ for $x_1 \neq x_2$, then any decoder d would require $x_1 = d(I) = x_2$, which is a contradiction. ■

Since Theorem 2.6.1 shows that no decoder exists for stationary observations in degenerate environments, an autoencoder *cannot* recover such a map; it cannot discriminate between locations with identical visual inputs.

Corollary 1. *For an autoencoder $g = dec \circ enc$ compressing images into a latent space $z \in \mathbb{R}^H$, there exists no mapping $z \overset{h}{\mapsto} x$ such that $x = h \circ enc \circ f(x)$.*

> ***Proof.*** Defining a decoder $d = h \circ \text{enc} : I \to x$ would lead to a contradiction via Theorem 2.6.1. ■

Emergence of place fields and vector navigation

While the previous section established that predictive coding networks embed richer spatial information than autoencoders, we now examine the specific structure of this spatial code. We show that individual units in the latent space activate within discrete, localized physical zones, mirroring the place fields found in mammalian brains (Figure 2.7(**a**)). These fields are overlapping and collectively span the entire environment. Every physical point is uniquely identified by the specific set of latent units active at that location. This combinatorial mapping allows for the reconstruction of the agent's coordinates. Moreover, any two locations are distinguishable by their respective latent activation patterns. Vector navigation involves representing the heading vector from a starting point to a target goal[51]. We demonstrate that these overlapping fields provide the necessary information for such calculations. In particular, a linear decoder can compute the vector to a goal location by evalu-

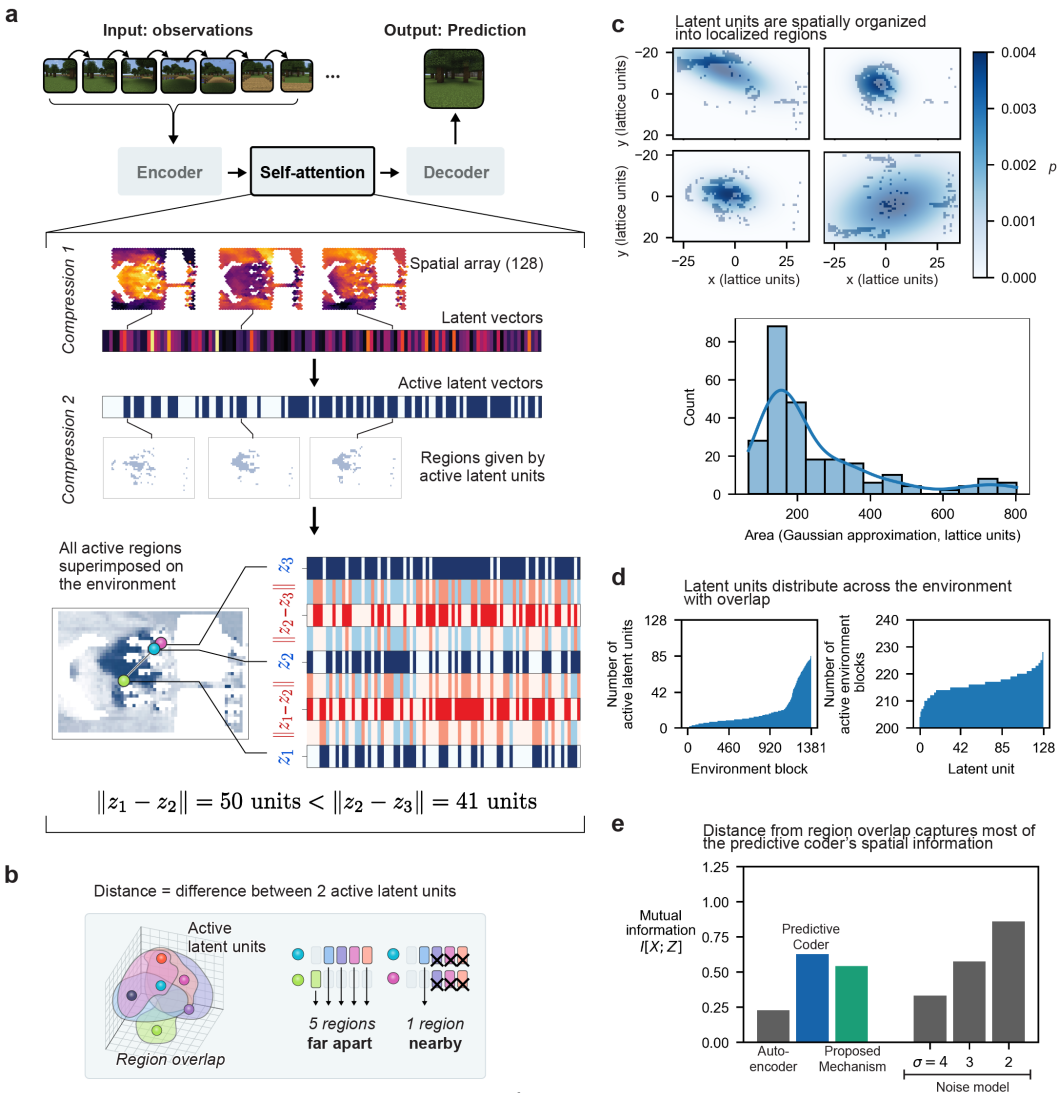


Figure 2.7: **The predictive coding network generates place fields that support vector based distance calculations a,** when encoding past images for predictive coding, the self-attention module generates latent vectors. Each continuous unit in these latent vectors activates in concentrated, localized regions in physical space. These continuous units can be thresholded to generate a binary vector determining whether each unit is active. Each latent unit covers a unique region, and each physical location gives a unique combination of these overlapping regions. As an agent moves away from its original location, the combination of overlapping regions gradually deviates from its original combinations. This deviation, as measured by Hamming distance, correlates with physical distance. **b,** distance is given by the difference in the latent units' overlapping regions. Two nearby locations have small deviations in overlap (right) while two distant locations have large deviations (middle).

Figure 2.7: **(continued) c,** latent units are spatially organized into localized regions. The active latent units are approximated by a two-dimensional Gaussian distribution to measure the latent unit's localization (top). The latent units' Gaussian approximations are highly localized with a mean area of 254.6 for densities $p \geq 0.0005$. **d,** latent units distributed across the environment. The number of latent units was calculated as each lattice block in the environment (left), and the number of lattice blocks were calculated for each active unit (right). The latent units provide a unique combination for 87.6% of the environment, and their aggregate covers the entire environment. **e,** distance from the region overlap captures most of the predictive coder's spatial information. We calculate the distance for every pair of active latent vectors and their respective physical Euclidean distances as a joint distribution. The proposed mechanism captures a majority of the predictive coder's spatial information—as the proposed mechanism's mutual information (0.542 bits) compares to the predictive coder's mutual information (0.627 bits).

ating the difference between place field activations, thereby supporting vector navigation (Figure 2.8)[1].

To validate this mechanism, we first confirm that the network produces localized place fields in the environment. We define latent unit activity by thresholding continuous values at their 90th percentile, ensuring stability across varying head directions. Localization is quantified by fitting the spatial distribution of each latent unit to a two-dimensional Gaussian (Figure 2.7(**c, top**):

$$p = \frac{1}{2\pi|\Sigma|} \exp\left[-\frac{1}{2}(x-\mu)^{\intercal}\Sigma^{-1}(x-\mu)\right]$$

We calculate the area of the resulting Gaussian ellipsoid for $p \geq 0.0005$ (Figure 2.7(**c**), bottom). Given an environment of $40 \times 65 = 2,600$ lattice units, these latent units exhibit significant focus. The mean approximation area is 254.6 units, with 80% of units falling below 352.6 units—covering roughly 9.79% and 13.6% of the total area, respectively.

Latent units provide a distinct combinatorial signature for every coordinate, and their collective reach spans the whole environment. Analyzing the number of active units per lattice block (Figure 2.7(**d**), left), we find unique counts in 87.6% of the environment. Every block activates at least one unit, confirming full spatial coverage. To verify robustness, we tested the stability of

[1]While studies like Bush *et al.* (2015) focus on grid cells, our implementation demonstrates vector navigation via place cells.

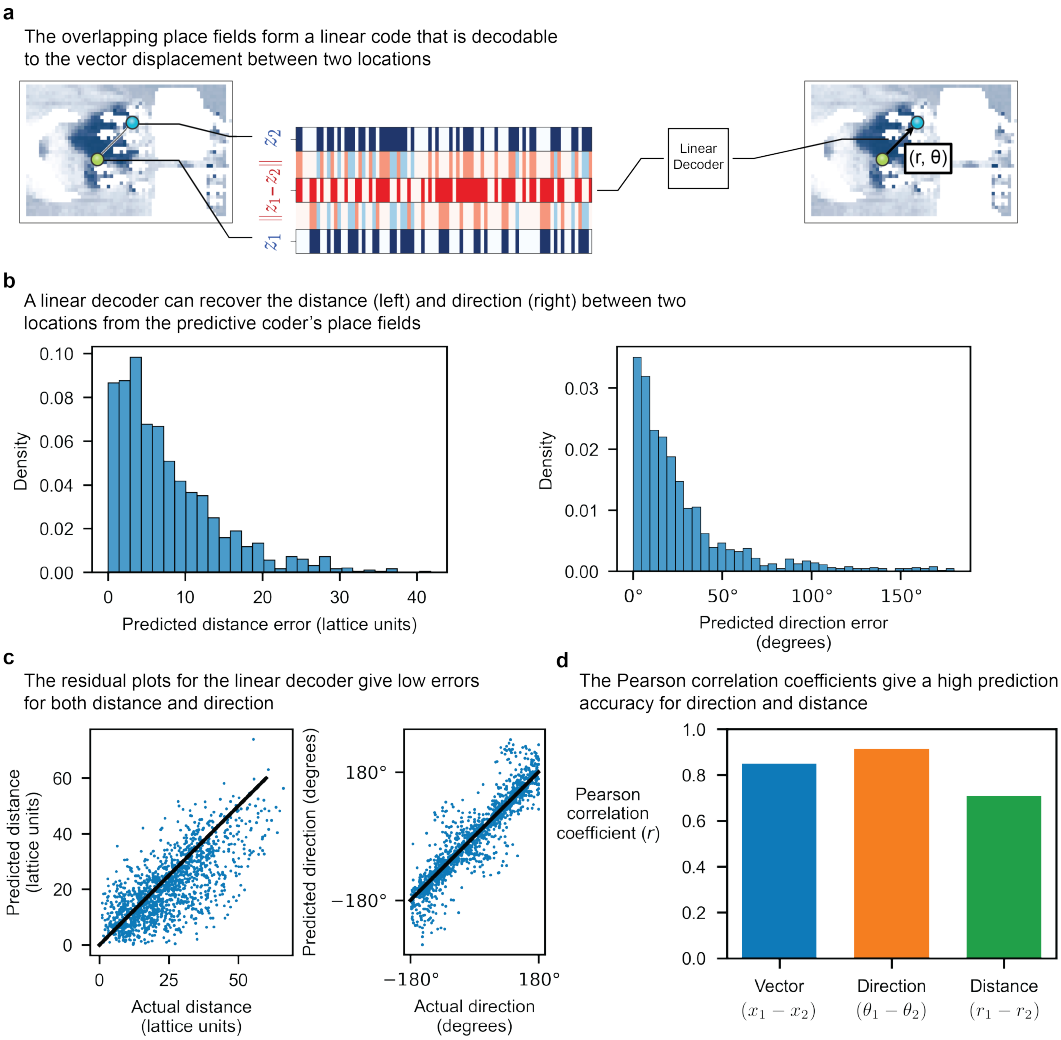


Figure 2.8: **The place field overlap between two locations is linearly decodable to a vector heading.** **a,** at different two locations x_1 and x_2, there exists a place field code z_1 and z_2, respectively. The bitwise different $z_1 - z_2$ gives the overlap between place fields at locations x_1 and x_2. We perform linear regression inputting the overlap codes $z_1 - z_2$ and predicting the vector displacement $x_1 - x_2$ between the two locations,

$$x_1 - x_2 = W[z_1 - z_2] + b.$$

The linear model forms a linear decoder from the place field code $z_1 - z_2$ to the vector displacement $x_1 - x_2$. **b,** the errors in predicted distance $\|r - \hat{r}\|$ (left) and predicted direction $\left\|\theta - \hat{\theta}\right\|$ (right) are decomposed from the predicted displacement $x_1 - x_2$. The linear decoder has a low prediction error for distance ($< 80\%$, 12.49 lattice units; mean, 7.89 lattice units) and direction ($< 80\%$, 48.04°; mean, $30.6°$). **c,** residual plots show both distance (left) and direction (right) are strongly correlated with the place field code. **d,** in addition, the place field code is strongly correlated with the vector heading with Pearson correlation coefficients of 0.861, 0.718, and 0.924 for vector displacement $(x_1 - x_2)$, distance $(r_1 - r_2)$, and direction $(\theta_1 - \theta_2)$, respectively.

these regions against landmark alterations by redistributing trees in the virtual world (Figure 2.3(**a-b**)). The fields remained stable despite these changes, maintaining a Jaccard index ($|S_{\text{new}} \cap S_{\text{old}}|/|S_{\text{new}} \cup S_{\text{old}}|$) of 0.828.

Finally, we show that comparing overlapping latent regions enables the network to compute physical distances and perform vector navigation. By thresholding latent values, we generate a 128-dimensional binary vector for each position. The difference $z_1 - z_2$ between these binary codes at two locations x_1 and x_2 is used to estimate the displacement $x_1 - x_2$ (Figure 2.8(**a**)). A linear decoder was fitted to this relationship:

$$x_1 - x_2 = W[z_1 - z_2] + b.$$

From the estimated displacement $\hat{x_1} - \hat{x_2}$, we derived distance and direction errors. The decoder achieved low errors in distance (mean 7.89 lattice units; 80% under 12.49 units) and orientation (mean 30.6°; 80% under 48.04°) (Figure 2.8(**b-c**)). The differential code $z_1 - z_2$ shows high Pearson correlation with direction (0.924) and distance (0.718) (Figure 2.8(**d**)). We also evaluated the alignment between bitwise distance[2] $|z_1 - z_2|$ and Euclidean physical distance $\|x_1 - x_2\|_{\ell_2}$. Analyzing the joint densities and mutual information, we found that the Hamming distance between binary vectors captures 0.542 bits of spatial information. This compares favorably with the full predictive coder's 0.627 bits and significantly exceeds the autoencoder's 0.227 bits (Figure 2.7(**e**)). This confirms that the overlapping region code preserves the vast majority of the network's spatial intelligence.

References

30. Vaswani, A. *et al. Attention Is All You Need* Aug. 2023.

41. Dieck, T. T. *Algebraic Topology* (American Mathematical Society, Zürich, 2008).

42. May, J. P. *A Concise Course in Algebraic Topology* (University of Chicago Press, Sept. 1999).

43. Hatcher, A. *Algebraic Topology* (Cambridge University Press, Cambridge ; New York, 2002).

[2]Unlike the Euclidean metrics used previously, bitwise distance involves thresholding latent units and calculating the L_1-norm difference.}

44. Shannon, C. E. A Mathematical Theory of Communication. *The Bell System Technical Journal* **27.** 379–423 (July 1948).

45. Johnson, M., Hofmann, K., Hutton, T. & Bignell, D. *The Malmo Platform for Artificial Intelligence Experimentation* in *International Joint Conference on Artificial Intelligence* (2016).

46. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]* (Dec. 2015).

47. Ronneberger, O., Fischer, P. & Brox, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation* May 2015.

48. Sutskever, I., Martens, J., Dahl, G. & Hinton, G. On the Importance of Initialization and Momentum in Deep Learning.

49. Smith, L. N. & Topin, N. *Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates* May 2018.

50. Tenenbaum, J. B., de Silva, V. & Langford, J. C. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science* **290.** 2319–2323 (Dec. 2000).

51. Bush, D., Barry, C., Manson, D. & Burgess, N. Using Grid Cells for Navigation. *Neuron* **87.** 507–520 (Aug. 5, 2015).

Chapter 3

PREDICTIVE CODING IN DYNAMICAL SYSTEMS

The theoretical results established in the previous chapter provided a rigorous foundation for why predictive coding recovers the geometry of an environment. We demonstrated that if a neural network's latent representation generates the same sequence of observations as the physical world, then that latent space must be topologically isomorphic to the underlying state space manifold. While our initial application focused on spatial navigation in discrete lattices and simple continuous arenas, the mathematical machinery of covering spaces and path-lifting is fundamentally indifferent to the physical nature of the state space.

This leads to a useful generalization: the predictive coding mapping theorem can be extended to any dynamical system (X, T, Φ) where X is a topological space, T represents time, and Φ represents the system's dynamics. In this broader framework, the map being recovered is not necessarily a geographic one, but rather the structural manifold that governs the system's evolution. By applying predictive coding to the observation space of an arbitrary dynamical system, an agent can unsupervisedly recover the latent coordinates and the transition rules that generate the observed data stream, effectively learning the laws of physics for that specific domain.

A critical application of this generalization lies in the field of robotics, specifically in the management of kinematic configuration spaces. As the degrees of freedom of a robotic system increase—as seen in complex manipulators or humanoid walkers—the configuration space $\mathcal{Q}$ becomes high-dimensional and non-Euclidean. For these systems, traditional inverse kinematics, which requires mapping a desired task-space position back to specific joint angles, often becomes analytically intractable or computationally prohibitive[28,34]. The complexity arises from the non-linear relationship between the high-dimensional joint space and the lower-dimensional task space, often resulting in singular configurations or multiple redundant solutions that are difficult to manage with hand-coded heuristics.

In this chapter, we investigate how predictive coding can bypass these tradi-

tional bottlenecks by recovering kinematic latent maps directly from sensory and proprioceptive experience. By tasking a world model with predicting the future sensory consequences of motor commands, we force the network to learn a latent manifold that is homeomorphic to the agent's actual kinematic structure. We will focus our investigation on bipedal locomotion, a notoriously difficult control problem characterized by periodic dynamics and high-dimensional interactions with the environment. We aim to show that the same predictive coding algorithms that build spatial maps of a room can autonomously chart the kinematic manifold of a walking agent, providing a functional body schema that supports movement and planning.

3.1 The predictive coding mapping theorem generalizes to any dynamical system

We formally define a (topological) dynamical system[1] as a triple (X, T, Φ), which consists of:

1. A state space X, which is typically a compact topological space (e.g., the configuration space $\mathcal{Q}$ of a robot).
2. A time space T, which is a group—typically continuous ($T = \mathbb{R}$) or discrete ($T = \mathbb{Z}$).
3. An evolution operator $\Phi : T \times X \to X$ that describes how the state $s \in X$ changes over time.

The evolution operator must satisfy the following identity and composition axioms:

1. *Identity:* $\Phi(0, s) = s$ for all $s \in X$ (the initial state is preserved).
2. *Composition*: $\Phi(t, \Phi(\tau, s)) = \Phi(t + \tau, s)$ for all $t, \tau \in T$ and $s \in X$ (the law of evolution is consistent across time steps).

In other words, Φ must satisfy a group action. In the context of this chapter, we consider agents observing the output of such a system through an observation function $\phi : X \to \mathcal{O}$. The task of predictive coding is to recover the manifold X and the flow Φ from the temporal correlations in the observation stream.

[1] see Behrisch *et al.* [52, p. 25], Vries [53, p. 7], and Sakai [54, p. 12]

By performing predictive coding, we will show that the recovered dynamical system $((Y, \mathbb{R}, \hat{\Phi}))$ is topologically conjugate to the actual dynamical system $((X, \mathbb{R}, \Phi))$—that is, there is a homeomorphism h such that

$$h \circ \hat{\Phi} = \Phi \circ h.$$

To prove the *predictive coding mapping theorem* for dynamical systems, we define the *trajectory map* $S_X : X \to C(\mathbb{R}, X)$, where $C(\mathbb{R}, X)$ is the space of all continuous functions from $\mathbb{R}$ to X.

For an initial state $x \in X$, $S_X(x) \in C(\mathbb{R}, X)$ is defined as the function mapping any time $t \in (0, t')$ to the state of the system at that time

$$[S_X(x)](t) = \Phi^t(x)$$

where $C(\mathbb{R}, X)$ is the space of all continuous functions from $\mathbb{R}$ to X.

In this context, S_X takes a single point in the state space and outputs its continuous orbit as a singular mathematical object. We will first show that the trajectory map S_X is continuous. To prove this proposition, we will use the Tube Lemma.

Lemma 1 (Tube Lemma)**.** *Let X be a topological space and let Y be a compact topological space. For a point $x \in X$ and W open in $X \times Y$, if $\{x\} \times Y \subset W$, then there is a open neighborhood N of x such that $N \times Y$ around $\{x\} \times Y$ is contained in W,*

$$N \times Y \subset W$$

Proposition 5. *The trajectory map $S_X : X \to C(\mathbb{R}, X)$ is continuous.*

> ***Proof.*** To prove S_X is continuous, it suffices to show that the preimage of any subbasic open set in $C(\mathbb{R}, X)$ is open in X. Let $V(I, U) = \{\gamma \in C(\mathbb{R}, X) \mid \gamma(I) \subseteq U\}$ be such a set, where $I \subset \mathbb{R}$ is compact and $U \subset X$ is open.
>
> Let $x \in S_X^{-1}(V(I, U))$. By definition, the trajectory of x over the interval I lies in U, which implies $\Phi(I \times \{x\}) \subseteq U$. Because the flow Φ is continuous, the preimage $\Phi^{-1}(U)$ is open in the product space $\mathbb{R} \times X$.
>
> Since I is compact and $I \times \{x\} \subseteq \Phi^{-1}(U)$, the Tube Lemma guarantees the existence of an open neighborhood $W \subset X$ containing x such that

$I \times W \subseteq \Phi^{-1}(U)$. This means for all $w \in W$ and $t \in I$, $\Phi^t(w) \in U$, which implies $S_X(W) \subseteq V(I, U)$.

Because $W \subseteq S_X^{-1}(V(I, U))$ and W is open, x is an interior point. Thus, the preimage $S_X^{-1}(V(I, U))$ is open, establishing that the trajectory map S_X is continuous. ■

Theorem 3.1.1 (Predictive coding mapping theorem (dynamical systems))**.** *Let $(S, \mathbb{R}, \Phi)$ and $(\hat{S}, \mathbb{R}, \hat{\Phi})$ be compact continuous-time dynamical systems. Let O be a Hausdorff topological space, and suppose there exist continuous sequence maps $S_S, S_{\hat{S}}$ and continuous observation maps $\phi, \hat{\phi}$ such that the composite maps $F = \phi \circ S_S : S \to C(\mathbb{R}, O)$ and $G = \hat{\phi} \circ S_{\hat{S}} : \hat{S} \to C(\mathbb{R}, O)$ are injective.*

If the induced dynamics of the flows on $C(\mathbb{R}, O)$ are strictly equal—that is,

$$F \circ \Phi^t \circ F^{-1} = G \circ \hat{\Phi}^t \circ G^{-1}$$

for all $t \in \mathbb{R}$—then the dynamical systems $(S, \mathbb{R}, \Phi)$ and $(\hat{S}, \mathbb{R}, \hat{\Phi})$ are topologically conjugate. Specifically, the map $H = G^{-1} \circ F$ is a topological conjugacy between the flows.

Proof. Let $W_S = F(S)$ and $W_{\hat{S}} = G(\hat{S})$ denote the images of the state spaces in $C(\mathbb{R}, O)$.

By hypothesis, the induced operators $F \circ \Phi^t \circ F^{-1}$ and $G \circ \hat{\Phi}^t \circ G^{-1}$ are strictly equal for all t. For two maps to be equal, their domains must coincide. The domain of $F \circ \Phi^t \circ F^{-1}$ is exactly the image W_S, and the domain of $G \circ \hat{\Phi}^t \circ G^{-1}$ is $W_{\hat{S}}$. Therefore, the equality of the operators implies that the images are identical, so we may define

$$F(S) = G(\hat{S}) \equiv W.$$

Because the constituent sequence and observation maps are continuous, the composites F and G are continuous. Since F and G are injective, their corestrictions onto W are continuous bijections.

Because O is Hausdorff, the function space $C(\mathbb{R}, O)$ (under standard topologies such as the compact-open topology) is also Hausdorff. There-

fore, W is a Hausdorff subspace. Since S is compact and W is Hausdorff, the continuous bijection $F : S \to W$ is a homeomorphism; in particular, $F^{-1} : W \to S$ is continuous. By symmetry, for the compact space $\hat{S}$, G is a homeomorphism onto W and $G^{-1} : W \to \hat{S}$ is continuous.

We construct the transition map $H : S \to \hat{S}$ as the composition

$$H = G^{-1} \circ F.$$

As the composition of two homeomorphisms ($F : S \to W$ and $G^{-1} : W \to \hat{S}$), H is a homeomorphism.

It remains to verify the equivariance condition $H \circ \Phi^t = \hat{\Phi}^t \circ H$. We begin with the assumed equality of the induced flows on W:

$$F \circ \Phi^t \circ F^{-1} = G \circ \hat{\Phi}^t \circ G^{-1}.$$

Right-composing with F yields

$$F \circ \Phi^t \circ F^{-1} \circ F = G \circ \hat{\Phi}^t \circ G^{-1} \circ F.$$

Since $F^{-1} \circ F = \mathrm{id}_S$ and $G^{-1} \circ F = H$, this simplifies to

$$F \circ \Phi^t = G \circ \hat{\Phi}^t \circ H.$$

Left-composing with G^{-1} yields

$$G^{-1} \circ F \circ \Phi^t = G^{-1} \circ G \circ \hat{\Phi}^t \circ H.$$

Applying $G^{-1} \circ F = H$ and $G^{-1} \circ G = \mathrm{id}_{\hat{S}}$, we obtain the conjugacy relation

$$H \circ \Phi^t = \hat{\Phi}^t \circ H.$$

Since H is a homeomorphism satisfying this equivariance condition, $(S, \mathbb{R}, \Phi)$ and $(\hat{S}, \mathbb{R}, \hat{\Phi})$ are topologically conjugate dynamical systems. ■

3.2 Neural networks can map kinematic configuration spaces using predictive coding

Humans interact with the world despite their interactions requiring complex movements, or trajectories. The somatosensory system—which provides sensory data for body position, movement, and touch—builds an internal map of these complex configurations and enables the human body to perform movements such as walking and throwing. Although control systems and machine learning can emulate various human locomotion through hard-wired algorithms, the somatosensory system suggests a general map of the human body's configuration that generalizes spatial, Cartesian maps to the coordinate system imposed by the human's kinematic constraints. Here we will demonstrate that predictive coding provides a neural network algorithm for constructing these somatosensory maps. We introduce rigid-body simulations of biomechanical robots that simulate touch and proprioception sensory systems (Figure 3.1(a)). We train a predictive coding neural network to predict future tactile and proprioceptive sensory input from past observations using a self-attention-equipped neural network. While learning to perform sensory prediction, we will determine how the agent automatically constructs an internal map of its kinematic configuration and the proximity of tactile sensors. Specifically, we will isolate the internal map that allows the agent to represent high-dimensional movement trajectories in a low-dimensional space (Figure 3.1(b)). In addition, we will identify whether the internal map emphasizes its explored task-specific trajectories by increasing the region allocation. By identifying the low-dimensional internal map, this project generalizes cognitive map construction beyond visual-spatial mapping.

In this section, we investigate how predictive coding recovers the low-dimensional maps of locomotive grammars—the underlying structural patterns that govern complex movement. We employ the predictive coding architecture established in Chapter II, but adapt it for high-dimensional kinematic data by replacing the convolutional neural network with a multilayer perceptron for both the encoder and decoder. The agent, a humanoid robot simulated in the MuJoCo environment, performs a diverse repertoire of behaviors—including walking, running, and jumping—that have been pre-trained using reinforcement learning. The neural network is trained to perform predictive coding on the stream of kinematic coordinates (proprioceptive joint angles and velocities), with the critical exception of the humanoid's global Cartesian position.

By tasking the network with predicting the future kinematic state without explicit global coordinates, we force it to learn an intrinsic latent representation of the body's movement dynamics. Using principal component analysis to visualize two-dimensional projections of the latent space, we observe that the network autonomously recovers a structured manifold where distinct locomotive grammars are clearly embedded. The resulting projections reveal how the agent's internal representation clusters periodic movement patterns, effectively charting the body schema required for complex locomotion.

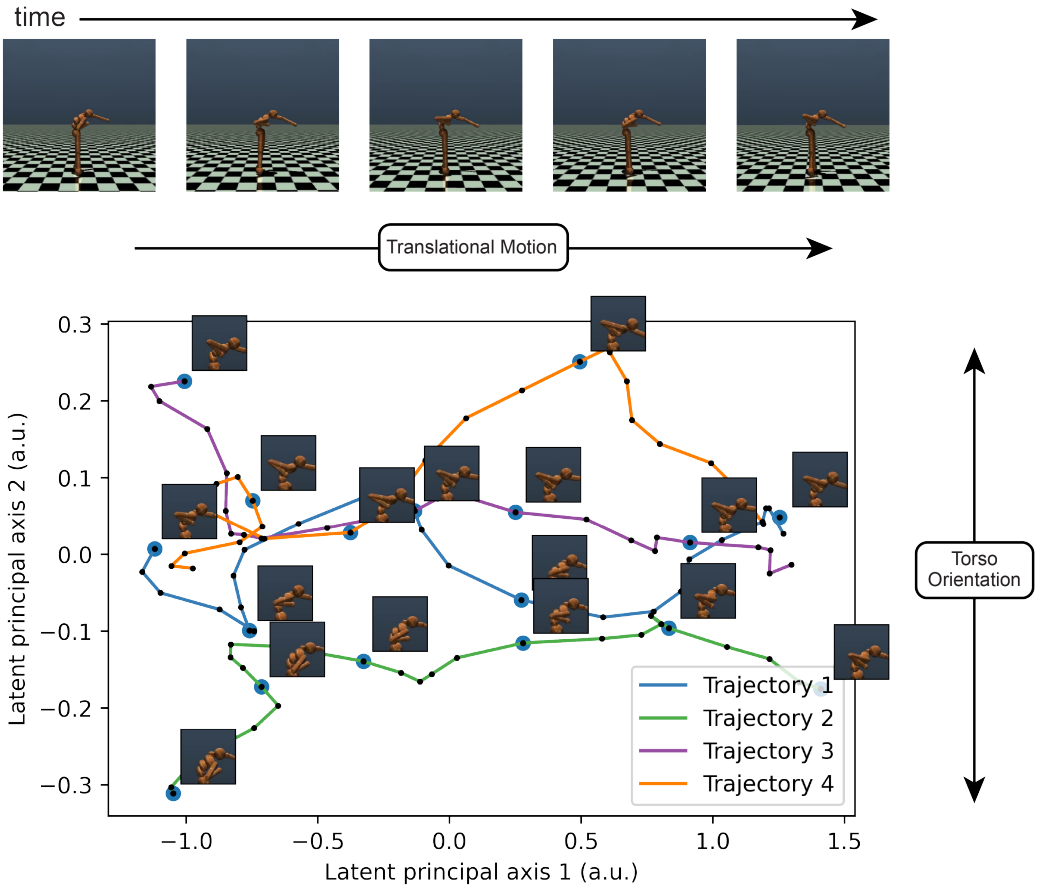


Figure 3.1: **A predictive coding neural network recovers a somatosensory map that captures different locomotion behaviors. a,** an agent performs different locomotion tasks (in this case, walking), and a neural network learns to predict future proprioceptive sensory inputs. **b,** different agent trials are recorded, and the proprioceptive information is passed into the predictive coding neural network. Within the neural network's latent space, the agent's translational motion and torso orientation are linearly encoded onto the latent space's principal axes.

Neural network can perform predictive coding on a robot's configuration space

To accommodate the kinematic nature of robotic state spaces, we utilize a predictive coding architecture where the high-dimensional sensory processing is handled by a multilayer perceptron rather than a convolutional network. The encoder f_θ maps the input kinematic vector $o_t \in \mathbb{R}^d$ to a latent representation z_t, while a self-attention mechanism integrates the history of these latent states to capture the temporal dependencies of movement. A corresponding MLP decoder g_ϕ then generates the predicted next observation $\hat{o}_{t+1}$ from the history-informed latent state. This architecture maintains the core predictive coding principle—minimizing the discrepancy between $\hat{o}_{t+1}$ and o_{t+1}—while specifically tailoring the feature extraction for proprioceptive data.

The experimental setup utilizes the MuJoCo physics engine to simulate a high-fidelity humanoid agent. This agent is pre-trained using reinforcement learning to master a variety of locomotor tasks, specifically walking, running, and jumping. The observation stream provided to the predictive coding network consists of the agent's full proprioceptive state—comprising joint angles and angular velocities—but critically excludes any information regarding the robot's global Cartesian position or orientation in the world. This exclusion ensures that any learned representations must be derived from the intrinsic kinematic constraints of the body's movement rather than external spatial coordinates. The network is then trained on these locomotor trajectories to predict the evolution of the kinematic state over time.

The neural network's latent space occupies a low-dimensional space that characterizes locomotive grammars

To investigate the topological structure of the learned representations, we apply principal component analysis to the high-dimensional latent vectors z_t produced by the predictive coder. By projecting the latent space onto its first two principal components, we can visualize the global organization of the movement manifold. This dimensionality reduction allows us to examine whether the network has recovered a coherent, low-dimensional coordinate system for locomotion, and how different types of movement are spatially distributed within this learned "body schema."

In the resulting principal component projections, we observe the emergence

of distinct, periodic structures that correspond to the "locomotive grammars" of walking, running, and jumping. Each behavior occupies a unique region of the latent manifold, with cyclic trajectories reflecting the repetitive nature of gait patterns. Notably, the transition regions between these behaviors suggest that the predictive coder has learned a continuous body schema where related movements are represented with topological proximity. This unsupervised recovery of the latent grammar of motion validates our hypothesis: predictive coding effectively charts the low-dimensional structural manifold of complex dynamical systems, even in the absence of explicit spatial or task-specific labels.

References

28. Zhang, X., Li, G., Shokouhi, S. & Thein, M.-W. L. Modern Control Meets Machine Learning: A Review and Taxonomy of Synergistic Approaches for Robotics Applications. *Actuators* **15.** 235 (May 2026).

34. Xu, M. *et al. Kinema4D: Kinematic 4D World Modeling for Spatiotemporal Embodied Simulation* (2026). Pre-published.

52. Behrisch, M., Kerkhoff, S., Pöschel, R., Schneider, F. M. & Siegmund, S. Dynamical Systems in Categories. *Applied Categorical Structures* **25.** 29–57 (Feb. 2017).

53. Vries, J. de. *Topological Dynamical Systems* 498 pp. (De Gruyter, Berlin ; Boston, 2014).

54. Sakai, S. *Operator Algebras in Dynamical Systems* 232 pp. (Cambridge University Press, Cambridge England New York, 1991).

Chapter 4

PREDICTIVE CODING FOR PLANNING AND EXPLORATION

Chapters II and III have established the theoretical and empirical foundations for spatial recovery through sensory prediction. Specifically, we have demonstrated that predictive coding—predicting future observations from past observations—provides a system to recover the underlying geometry and topology of arbitrary environments, from discrete lattices to topological spaces. By minimizing the discrepancy between predicted and actual sensory observations, an agent unsupervisedly constructs an internal representation that is homeomorphic to its physical surroundings, effectively lifting local observations into a globally consistent cognitive map.

In this chapter, we transition from the theoretical recovery of spatial structure to the practical application of these learned representations. The maps generated from predictive coding are not only spatial structure; they can be adapted for complex tasks. We will demonstrate how these internal coordinate systems provide the necessary geometric grounding for two fundamental cognitive tasks: planning and exploration. By leveraging the learned transition dynamics and topological connectivity of the latent space, an agent can perform goal-directed navigation and systematic discovery of novel environments with significantly greater efficiency than methods lacking such structural priors.

The chapter is organized as follows. First, we provide a review of methods in reinforcement learning, establishing the mathematical framework of Markov decision processes and value functions that we will employ. Second, we formulate planning as a reinforcement learning problem and provide statistical methods for planning, demonstrating how value iteration and policy gradients can be effectively performed within the predictive coding latent space to achieve goal-directed movement. Finally, we formulate exploration as a reinforcement learning problem and provide statistical methods for exploration in predictive coding, extending the wavefront propagation method in Chapter I (1.5).

4.1 Preliminaries

The Agent-Environment Interface

The foundation of reinforcement learning is the closed-loop interaction between an autonomous *agent* and its external *environment*. This interaction is formalized as a sequence of discrete time steps $t = 0, 1, 2, \ldots$. At each time step t, the agent receives a representation of the environment's *state*, $S_t \in \mathcal{S}$, and on that basis selects an *action*, $A_t \in \mathcal{A}(s)$. One step later, as a consequence of its action, the agent receives a numerical *reward*, $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$, and finds itself in a new state, S_{t+1}.

This framework is universal: the "agent" represents the decision-maker (the mind), while the "environment" represents everything outside the agent's direct control, including its own physical body in robotic applications. The boundary between agent and environment is defined by the limit of the agent's absolute control, not by physical constraints. The objective of the agent is to develop a *policy*, $\pi(a|s)$, which is a mapping from states to probabilities of selecting each possible action, such that a cumulative measure of the rewards is maximized over the long run.

Goals, Rewards, and Returns

In reinforcement learning, the agent's goal is formalized as the maximization of the expected value of the cumulative sum of a scalar reward signal. This is known as the *return*, G_t. In the simplest case, the return is the sum of the rewards:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

where T is a final time step. This formulation is suitable for *episodic tasks*, which naturally break into sequences that end in a terminal state (e.g., a game of chess).

However, many tasks are *continuing*, meaning the interaction does not have a natural end. To prevent the return from becoming infinite, we introduce a *discount factor*, $\gamma \in [0, 1]$. The discounted return is defined as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The discount factor determines the present value of future rewards: a reward received k steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately. As γ approaches 0, the agent becomes "myopic,"

caring only about immediate rewards. As γ approaches 1, the agent becomes more "farsighted," valuing future rewards nearly as much as current ones.

The Markov Property

A critical assumption in the Markov decision process framework is the **Markov property**. A state signal S_t is said to be Markov if and only if:

$$P(S_{t+1} = s', R_{t+1} = r | S_t, A_t, S_{t-1}, A_{t-1}, \ldots, S_0, A_0) = P(S_{t+1} = s', R_{t+1} = r | S_t, A_t)$$

for all s', r, and all possible histories of past states and actions. In essence, the Markov property asserts that the current state S_t captures all relevant information from the past required to predict the future. If the environment is Markov, then knowing the full history of interactions provides no additional advantage over knowing just the current state. This allows us to predict the next state S_{t+1} and reward R_{t+1} using only the current state S_t and action A_t, greatly simplifying the mathematical modeling of the system.

The Markov Decision Process

A *Markov Decision Process* provides a precise mathematical definition of the RL problem for environments that satisfy the Markov property. An Markov decision process is defined by a 5-tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$:

1. $\mathcal{S}$ is a finite set of states.
2. $\mathcal{A}$ is a finite set of actions.
3. $p(s'|s, a) = \mathbb{P}(S_{t+1} = s' | S_t = s, A_t = a)$ is the transition probability.
4. $r(s, a, s') = \mathbb{E}[R_{t+1} | S_t = s, A_t = a, S_{t+1} = s']$ is the expected reward.
5. $\gamma \in [0, 1]$ is the discount factor.

To ground these abstract definitions, consider a *Grid World* environment. The state space $\mathcal{S}$ consists of 16 cells in a 4×4 grid, indexed from $(0, 0)$ to $(3, 3)$. The action space $\mathcal{A}$ contains four discrete movements: {North, South, East, West}. The environment is characterized by:

- *Obstacles*: Certain cells are "Walls" that the agent cannot enter. If the agent takes an action that would lead into a wall, the transition probability $p(s|s, a) = 1$, meaning the agent stays in its current cell.

- *Traps*: Some cells represent "Traps" with a large negative reward, $r(s, a, s_{\text{trap}}) = -10$.
- *Goal*: A single "Goal" cell exists with a large positive reward, $r(s, a, s_{\text{goal}}) = +10$.
- *Step Penalty*: To encourage the agent to find the shortest path, every other transition yields a small negative reward, $r(s, a, s') = -1$.

The agent's task in this Grid World is to navigate from a starting cell to the Goal while avoiding Traps and minimizing the number of steps taken, all while subject to the stochastic or deterministic dynamics defined by $p(s'|s, a)$.

The Value Function

Almost all reinforcement learning algorithms involve estimating **value functions**—functions of states (or state-action pairs) that estimate how "good" it is for the agent to be in a given state. This "goodness" is defined in terms of the expected return.

The *state-value function* for a policy π, denoted $V^\pi(s)$, is the expected return when starting in state s and following policy π thereafter:

$$V^\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right]$$

Similarly, we define the *action-value function* (or Q-function), $Q^\pi(s, a)$, as the expected return starting from state s, taking action a, and thereafter following policy π:

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right]$$

In our Grid World, $V^\pi(s)$ might represent the "proximity" to the Goal cell under a random walk policy. States near the Goal would have higher values, while states near a Trap would have lower (potentially negative) values.

The Bellman Equation

A fundamental property of value functions in reinforcement learning is that they satisfy particular recursive relationships. We can derive the *Bellman*

equation by decomposing the return G_t into its immediate and future components. Recall the definition of the state-value function:

$$V^\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

By expanding the definition of the discounted return $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$, we can write:

$$G_t = R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots) = R_{t+1} + \gamma G_{t+1}$$

Substituting this back into the expectation gives:

$$V^\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$$

Using the Law of Total Expectation, we can marginalize over all possible actions a, next states s', and rewards r:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']\right]$$

Recognizing that $\mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']$ is exactly the value of the successor state $V^\pi(s')$, we arrive at the standard Bellman equation:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V^\pi(s')]$$

This equation expresses a consistency condition: the value of a state must equal the expected reward for the immediate transition plus the discounted value of the next state.

To *optimize* the Markov decision process, we seek the policy that maximizes the expected return. The *Bellman optimality equation* formalizes this by selecting the action that yields the highest expected value at each step. For the optimal value function $V^*(s) = \max_\pi V^\pi(s)$, the recursive relationship becomes:

$$V^*(s) = \max_{a \in \mathcal{A}(s)} \mathbb{E}[R_{t+1} + \gamma V^*(S_{t+1}) | S_t = s, A_t = a]$$

Expanding the expectation:

$$V^*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r|s, a)[r + \gamma V^*(s')]$$

This equation represents the pinnacle of MDP optimization. It states that under an optimal policy, the value of a state is equal to the expected return from the best possible action in that state. Solving this equation is equivalent to finding the optimal behavior in the environment. In our Grid World, the Bellman optimality equation ensures that at each cell, the agent's value is determined by the neighboring cell that offers the "shortest" path to the Goal, effectively aligning the entire value surface to the optimal objective.

Value Iteration using the Bellman Equation

Value Iteration is an iterative algorithm that computes the optimal value function V^* by turning the Bellman optimality equation into an update rule. We start with an arbitrary initial value function V_0 (e.g., $V_0(s) = 0$ for all s) and repeatedly apply the following update for every state s:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V_k(s')]$$

This process is guaranteed to converge to V^* as $k \to \infty$.

In the *Grid World*, consider how the reward "propagates" through the grid. Initially, all states have a value of 0. After the first iteration, only the states immediately adjacent to the Goal cell will have their values updated (reflecting the immediate +10 reward). In the second iteration, the states two steps away from the Goal will "see" the high value of their neighbors and update their own values accordingly, discounted by γ and penalized by the step cost. This "wave" of value updates flows outward from the high-reward areas, eventually filling the entire grid with a gradient of values that "point" toward the Goal.

Policy Iteration

Policy Iteration is an alternative approach to finding the optimal policy π^*. It consists of two simultaneous, interacting processes: one making the value function consistent with the current policy (*Policy Evaluation*), and the other making the policy greedy with respect to the current value function (**Policy Improvement**).

Policy Evaluation

In this phase, we compute the state-value function V^π for a fixed, arbitrary policy π. This is done by iteratively applying the Bellman equation as an update rule:

$$V_{k+1}(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V_k(s')]$$

In our Grid World, if our current policy π is to always move East, Policy Evaluation will tell us the expected return from every cell under this specific strategy. If moving East eventually leads to a Trap or a Wall, the values will reflect that inefficiency or danger.

Policy Improvement

Once we have evaluated the current policy and obtained V^π, we can refine it. The *Policy Improvement Theorem* states that if we change the policy to be "greedy" with respect to V^π, the new policy π' will be at least as good as, if not better than, π:

$$\pi'(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V^\pi(s')]$$

In the Grid World, this is where the agent's behavior is optimized. Suppose the current policy was a wandering, stochastic one. After evaluation, the agent knows which neighboring cells have higher values. Policy Improvement shifts the probability mass toward the actions that lead to those higher-value cells. If moving South was previously ignored but now shows a higher expected return due to its proximity to the Goal, the policy is updated to favor South. This process of evaluation and improvement repeats until the policy is stable, at which point it is guaranteed to be the optimal policy π^*.

Policy gradient

While value-based methods like value iteration and Q-learning focus on estimating the value of states and actions to derive a policy (typically a greedy one), *policy gradient* methods seek to optimize the policy directly. In this framework, the policy is parameterized by a vector $\theta \in \mathbb{R}^d$, denoted as $p_{\pi_\theta}(a|s) = \mathbb{P}(A_t = a|S_t = s, \theta)$. The goal is to learn the parameters θ that maximize a performance measure $J(\theta)$, typically defined as the expected return.

Direct policy parameterization offers several advantages over value-based approaches. First, it can naturally handle continuous action spaces, where finding the maximum of a Q-function at every step would be computationally prohibitive. Second, policy gradient methods can learn stochastic policies, which are optimal in certain scenarios (such as partially observable environments or adversarial games) where a deterministic greedy policy might be easily exploited or fail to explore effectively. Finally, the policy parameterization can be simpler to approximate than the value function, especially when the optimal behavior has a simpler geometric structure than the corresponding value landscape.

As a side note, it is quite common in reinforcement learning for the policy function $p_{\pi_\theta}(a|s)$ to be written as $\pi_\theta(a|s)$ and for the expectation over a policy

p_{π_θ} to be written in the shorthand

$$\mathbb{E}_{p_{\pi_\theta}}[x] = \sum_{t=1}^{T} \mathbb{E}_{p_{\pi_\theta}(a_t|s_t)} \mathbb{E}_{p(s_{t+1}|a_t,s_t)}[x].$$

We prefer the p_{π_θ} as it makes it explicit that the policy function is a probability density rather than function.

The Policy Objective Function

To formalize the optimization problem, we define the objective function $J(\theta)$ as the expected return starting from the initial state S_0 under the policy p_{π_θ}:

$$J(\theta) = \mathbb{E}_{p_{\pi_\theta}}[G_0].$$

In the episodic case, this can be written as the sum over all possible trajectories $\tau = (s_0, a_0, r_1, s_1, \ldots, s_T, a_T, r_{T+1})$:

$$J(\theta) = \sum_{\tau} p(\tau|\theta) G(\tau)$$

where $p(\tau|\theta)$ is the probability of trajectory τ occurring under policy p_{π_θ}, and $G(\tau) = \sum_{t=0}^{T} \gamma^t r_{t+1}$ is the return of that trajectory. The probability of a trajectory is given by the product of the initial state distribution, the transition dynamics, and the policy:

$$p(\tau|\theta) = \mu(s_0) \prod_{t=0}^{T} p_{\pi_\theta}(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

The Policy Gradient Theorem

The challenge in optimizing $J(\theta)$ lies in the fact that the performance depends on the policy in two ways: first, the policy determines the actions taken, and second, the policy determines the distribution of states the agent encounters. While the effect on actions is easy to differentiate, the effect on the state distribution is complex and depends on the unknown environment dynamics $p(s'|s, a)$.

The *Policy Gradient Theorem* provides an elegant solution to this problem by showing that the gradient $\nabla_\theta J(\theta)$ can be expressed in a form that does not involve the derivative of the state distribution. For the discounted return objective, the gradient is proportional to:

$$\nabla_\theta J(\theta) \propto \sum_{s} d^\pi(s) \sum_{a} \nabla_\theta p_{\pi_\theta}(a|s) Q^\pi(s, a)$$

where $d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t P(S_t = s|S_0, p_{\pi_\theta})$ is the discounted state distribution. Using the "log-derivative trick" $\nabla_\theta p_{\pi_\theta}(a|s) = p_{\pi_\theta}(a|s)\nabla_\theta \log p_{\pi_\theta}(a|s)$, we can rewrite the gradient as an expectation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{p_{\pi_\theta}}[\nabla_\theta \log p_{\pi_\theta}(a|s) Q^\pi(s,a)]$$

By sampling actions from the current policy and weighting the gradient of the log-probability by the expected return, we can estimate the gradient of the objective function.

Advantage Estimation

The *Policy Gradient Theorem* provides a equation for calculating the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{p_{\pi_\theta}}[\nabla_\theta \log p_{\pi_\theta}(a|s) Q^\pi(s,a)].$$

However, naïvely sampling the policy gradient $\nabla_\theta J(\theta)$ produces a large variance. Because the returns G_t can vary significantly between episodes, the gradient estimates can be extremely noisy, leading to slow convergence. To mitigate this, we can introduce a *baseline* $b(s)$ that does not depend on the action a:

$$\nabla_\theta J(\theta) = \mathbb{E}_{p_{\pi_\theta}}[\nabla_\theta \log p_{\pi_\theta}(a|s)(Q^\pi(s,a) - b(s))]$$

It can be shown that subtracting any baseline $b(s)$ does not change the expectation of the gradient, as

$$\mathbb{E}_{p_{\pi_\theta}}[\nabla_\theta \log p_{\pi_\theta}(a|s) b(s)] = 0.$$

However, choosing a baseline that is close to the expected return—such as the state-value function $V^\pi(s)$—significantly reduces the variance of the gradient estimate.

The quantity $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ is known as the *advantage function*. It measures how much better an action a is compared to the average action in state s. Using the advantage function in the policy gradient update leads to the *Actor-Critic* architecture, where one model (the critic) learns to estimate $V^\pi(s)$ while another model (the actor) learns to optimize p_{π_θ} using the critic's feedback.

Generalized advantage estimation

A central challenge in policy gradient methods is the estimation of the advantage function $A^\pi(s,a)$. The naïve estimator uses the return G_t suffers from extremely high variance. Conversely, using the value function $V^\pi(s)$ produces a low-variance estimate but introduces significant bias if the value function $V^\pi(s)$ is inaccurate. *Generalized Advantage Estimation*[56] provides a framework to navigate this bias-variance tradeoff by interpolating between these two methods.

The TD Error and Multi-Step Advantages

We begin by defining the one-step temporal difference (TD) error δ_t as the discrepancy between the actual reward plus the discounted value of the next state and the current value estimate:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

The TD error can be viewed as an estimate of the advantage $A^\pi(s_t, a_t)$ using only immediate information. We can extend this to a k-step advantage estimator by summing the discounted TD errors over a horizon k:

$$\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l \delta_{t+l} = -V(s_t) + r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^k V(s_{t+k}).$$

As $k \to 1$, we recover the one-step TD estimate $\hat{A}_t^{(1)} = \delta_t$, which has low variance but high bias. As $k \to \infty$, we recover the naïve return minus the baseline $\hat{A}_t^{(\infty)} = G_t - V(s_t)$, which is unbiased but high-variance.

The GAE Estimator

GAE defines the advantage estimator $\hat{A}_t^{GAE(\gamma,\lambda)}$ as the exponentially weighted average of these k-step advantage estimators, analogous to the TD(λ) algorithm in value-based learning:

$$\hat{A}_t^{GAE(\gamma,\lambda)} = (1-\lambda) \sum_{k=1}^{\infty} \lambda^{k-1} \hat{A}_t^{(k)}$$

where $\lambda \in [0, 1]$ is a hyperparameter that controls the tradeoff. By substituting the definition of $\hat{A}_t^{(k)}$ and simplifying, we arrive at the recursive form of the GAE estimator:

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

This formulation is remarkably elegant: it shows that the generalized advantage at time t is simply the discounted sum of all future TD errors, where the effective discount factor is the product $\gamma\lambda$.

Reparameterization and the Bias-Variance Tradeoff

The GAE estimator serves as a powerful reparameterization of the policy gradient. By adjusting λ, we can smoothly transition between two extremes:

- $\lambda = 0$**:** $\hat{A}_t^{GAE(\gamma,0)} = \delta_t$. This yields a low-variance estimate suitable for stable, albeit potentially biased, updates.
- $\lambda = 1$**:** $\hat{A}_t^{GAE(\gamma,1)} = \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} = G_t - V(s_t)$. This yields the unbiased Monte Carlo advantage used in REINFORCE, which is often too noisy for deep RL.

In practice, values of λ between 0.9 and 0.99 are typically used. This allows the agent to incorporate information from far into the future (reducing bias) while still benefiting from the variance-reduction properties of the learned value function. GAE is particularly critical for the success of TRPO and PPO, as the stability of these "trust-region" updates depends heavily on the quality and consistency of the advantage estimates.

Trust-region policy optimization

While policy gradient methods provide a solid theoretical basis for direct policy optimization, they often suffer from instability during training. A single large update to the policy parameters θ can drastically change the distribution of states the agent visits, leading to a catastrophic collapse in performance from which the agent may never recover. *Trust-Region Policy Optimization* (TRPO)[57] addresses this by ensuring that each update stays within a "trust region" where the local approximation of the objective function is reliable.

TRPO is motivated by the desire to achieve *monotonic improvement* in the policy. The key insight is to constrain the step size not in the parameter space

$\mathbb{R}^d$, but in the space of probability distributions. This is achieved by imposing a constraint on the Kullback-Leibler (KL) divergence between the old policy and the new policy.

The Surrogate Objective

TRPO optimizes a *surrogate objective* that approximates the true performance measure $J(\theta)$. Let $p_{\pi_{\theta_{\text{old}}}}$ be the current policy. We define the objective function $L_{\theta_{\text{old}}}(\theta)$ as:

$$L_{\theta_{\text{old}}}(\theta) = \mathbb{E}_{s\sim d^{\pi_{\text{old}}}, a\sim\pi_{\text{old}}}\left[\frac{p_{\pi_\theta}(a|s)}{p_{\pi_{\theta_{\text{old}}}}(a|s)} A^{\pi_{\text{old}}}(s,a)\right]$$

where $A^{\pi_{\text{old}}}(s,a)$ is the advantage function. The ratio $\frac{p_{\pi_\theta}(a|s)}{p_{\pi_{\theta_{\text{old}}}}(a|s)}$ is an importance sampling weight that accounts for the difference between the new and old policies. When $\theta = \theta_{\text{old}}$, the gradient of the surrogate objective is exactly the policy gradient: $\nabla_\theta L_{\theta_{\text{old}}}(\theta)|_{\theta=\theta_{\text{old}}} = \nabla_\theta J(\theta)$.

The KL Divergence Constraint

To ensure the surrogate objective remains a good approximation of the true performance, TRPO solves the following constrained optimization problem:

$$\begin{aligned} \text{maximize}_\theta \quad & L_{\theta_{\text{old}}}(\theta) \\ \text{subject to} \quad & \bar{D}_{KL}(\theta_{\text{old}}||\theta) \leq \delta \end{aligned}$$

where

$$\bar{D}_{KL}(\theta_{\text{old}}||\theta) = \mathbb{E}_{s\sim d^{\pi_{\text{old}}}}[D_{KL}(p_{\pi_{\theta_{\text{old}}}}(\cdot|s)||p_{\pi_\theta}(\cdot|s))]$$

is the expected KL divergence over the state distribution, and $\delta > 0$ is a small step size parameter. This constraint forces the new policy to be "close" to the old one in terms of the information it carries, preventing large, erratic jumps in behavior.

Second-Order Approximation and Conjugate Gradient

Solving the constrained optimization problem directly is computationally expensive. TRPO simplifies this by taking a linear approximation of the objective and a quadratic approximation of the constraint. Let $g = \nabla_\theta L_{\theta_{\text{old}}}(\theta)$ be the

policy gradient and $H = \nabla_\theta^2 \bar{D}_{KL}(\theta_{\text{old}}||\theta)$ be the Hessian of the KL divergence. The Hessian H is also known as the *Fisher information matrix*.

The approximate problem becomes:

$$\begin{aligned} &\text{maximize}_\theta \quad g^T(\theta - \theta_{\text{old}}) \\ &\text{subject to} \quad \frac{1}{2}(\theta - \theta_{\text{old}})^T H(\theta - \theta_{\text{old}}) \leq \delta \end{aligned}$$

The solution to this quadratic program is:

$$\theta - \theta_{\text{old}} = \sqrt{\frac{2\delta}{g^T H^{-1} g}} H^{-1} g$$

In practice, computing the inverse of the high-dimensional Fisher information matrix H^{-1} is infeasible. TRPO instead uses the *Conjugate Gradient* (CG) algorithm to approximately solve the linear system $Hx = g$ for $x = H^{-1}g$. This approach only requires computing Hessian-vector products Hv, which can be efficiently calculated using automatic differentiation without ever explicitly forming the matrix H. Finally, because the linear and quadratic approximations may be inaccurate for larger steps, a line search is typically performed along the direction $H^{-1}g$ to ensure that the KL constraint is satisfied and the objective L actually increases.

TRPO's use of the natural gradient direction ($H^{-1}g$) makes it invariant to the parameterization of the policy, allowing it to navigate the geometry of the policy space more effectively than standard first-order methods. However, the complexity of the second-order optimization and the requirement for large batches to accurately estimate the Fisher information matrix led to the development of simpler first-order alternatives like Proximal Policy Optimization.

Proximal policy optimization

While Trust-Region Policy Optimization (TRPO) provides robust and monotonic improvements, its reliance on second-order derivatives and the Conjugate Gradient method makes it computationally complex and difficult to integrate with architectures that share parameters between the policy and value functions (e.g., deep neural networks where the actor and critic share convolutional layers). *Proximal Policy Optimization* (PPO)[58] was developed to retain the data efficiency and reliable performance of TRPO while relying strictly on first-order optimization techniques.

PPO achieves this by substituting TRPO's explicit KL-divergence constraint with a modified objective function that penalizes excessively large policy updates. Instead of solving a complex constrained optimization problem, PPO can be trained using standard stochastic gradient descent algorithms like Adam.

The Probability Ratio

Like TRPO, PPO relies on importance sampling to evaluate a new policy p_{π_θ} using trajectory data gathered by an old policy $p_{\pi_{\theta_{\text{old}}}}$. We define the probability ratio $r_t(\theta)$ as:

$$r_t(\theta) = \frac{p_{\pi_\theta}(a_t|s_t)}{p_{\pi_{\theta_{\text{old}}}}(a_t|s_t)}$$

Notice that when $\theta = \theta_{\text{old}}$, the ratio $r_t(\theta) = 1$. Using this ratio, the unconstrained surrogate objective from TRPO can be rewritten as:

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[r_t(\theta) \hat{A}_t \right]$$

where $\hat{A}_t$ is an estimator of the advantage function at time t, and the superscript CPI denotes the Conservative Policy Iteration objective upon which TRPO is based. Optimizing L^{CPI} directly without constraints would lead to destructively large policy updates.

The Clipped Surrogate Objective

To prevent the new policy from diverging too far from the old policy, PPO introduces a clipping mechanism into the objective function. The **clipped surrogate objective**, $L^{CLIP}(\theta)$, is defined as:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t,\, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right]$$

where ϵ is a small hyperparameter (typically $\epsilon = 0.1$ or 0.2) that defines the bounds of the trust region.

This objective function operates by taking the minimum of two terms:

1. The standard, unclipped surrogate objective $r_t(\theta)\hat{A}_t$.
2. A clipped version where the probability ratio $r_t(\theta)$ is constrained to the interval $[1-\epsilon, 1+\epsilon]$.

The behavior of this objective depends on the sign of the advantage function $\hat{A}_t$:

- *Positive Advantage ($\hat{A}_t > 0$):* The action a_t was better than average. The unclipped objective wants to increase $p_{\pi_\theta}(a_t|s_t)$ indefinitely. However, the clipping mechanism restricts $r_t(\theta)$ to a maximum of $1+\epsilon$. Once the ratio hits this upper bound, the gradient becomes zero, preventing further increases in probability for that action during the current update phase.
- *Negative Advantage ($\hat{A}_t < 0$):* The action a_t was worse than average. The unclipped objective wants to decrease $p_{\pi_\theta}(a_t|s_t)$ toward zero. Here, the clipping mechanism restricts $r_t(\theta)$ to a minimum of $1-\epsilon$. Once the ratio hits this lower bound, the gradient again becomes zero, stopping further decreases.

By taking the minimum of the unclipped and clipped terms, the objective function forms a lower bound (a pessimistic estimate) on the unclipped performance. It allows the policy to change as long as the change is beneficial and within the $(1\pm\epsilon)$ neighborhood, but it aggressively penalizes changes that move the policy far outside this region, effectively enforcing a trust region without the need for computationally expensive KL-divergence calculations.

The Complete PPO Objective

In practical implementations where the policy (actor) and value function (critic) share neural network parameters, the loss function must simultaneously optimize the policy, minimize the value function error, and encourage sufficient exploration. The full PPO objective is often constructed as a combination of three terms:

$$L^{PPO}(\theta) = \hat{\mathbb{E}}_t \left[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[p_{\pi_\theta}](s_t) \right]$$

where $L^{VF}(\theta) = (V_\theta(s_t) - V_t^{\text{target}})^2$ is the squared-error value function loss, $S[p_{\pi_\theta}](s_t)$ is the entropy bonus which encourages exploration by preventing the policy from prematurely converging to a deterministic state, and c_1, c_2 are weighting coefficients. This unified, first-order objective makes PPO highly scalable and structurally simple, cementing its status as a standard algorithm in modern deep reinforcement learning.

4.2 Planning using predictive coding maps

The preceding sections have established a rigorous framework for reinforcement learning, progressing from the foundational Markov decision process to advanced policy gradient methods like Proximal Policy Optimization (PPO) and variance reduction techniques like Generalized Advantage Estimation (GAE). While these methods provide a powerful engine for behavior optimization, their success in complex environments depends heavily on the agent's ability to access a Markovian state representation.

In most real-world applications, however, agents do not have direct access to the underlying state of the environment. Instead, they operate in partially observable settings, where sensory observations—such as high-dimensional camera images or noisy sensor readings—only provide a filtered and incomplete view of the world. In such scenarios, standard RL algorithms often fail because the observations lack the history required to satisfy the Markov property, making the transition dynamics and value functions non-stationary and difficult to learn.

Predictive coding provides a principled solution to this challenge of partial observability. As demonstrated in Chapters II and III, predictive coding neural networks learn to minimize the discrepancy between predicted and actual future observations. Through this process, the network's latent space recovers a globally consistent map of the environment, effectively lifting local, partially observable inputs into a latent coordinate system that captures the underlying topological and geometric structure. Furthermore, as shown in the humanoid locomotion experiments of Chapter III, these maps are not mere static representations; they can be adapted into task-specific maps where the latent dynamics are tuned to the specific requirements of the agent's morphology and objectives.

In this section, we formulate the task of planning as the application of reinforcement learning methods—specifically Proximal Policy Optimization—directly within these learned predictive coding maps. Rather than learning a policy over raw observations o_t, the agent learns a policy over the latent states x_t extracted from the predictive coding network. Because the latent state x_t is constructed to be a sufficient statistic for future observations, it serves as a learned Markovian state that satisfies the requirements of the MDP framework, even when the original observations do not.

Reformulating PPO with predictive coding latent states

To formally integrate predictive coding with PPO, we redefine the agent's policy and value functions to operate on the latent space $\hat{\mathcal{S}}$. Let $x_t = f_\phi(o_{\leq t})$ be the latent state generated by the predictive coding network with parameters ϕ, taking the history of observations as input. The policy $p_{\pi_\theta}(a|x)$ and value function $V_\psi(x)$ are now parameterized functions of the latent representation with parameters θ and ψ, respectively.

The reformulated PPO objective function for an agent planning in the predictive coding latent space is given by:

$$L^{PPO}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right]$$

where the probability ratio $r_t(\theta)$ is defined in terms of the latent state:

$$r_t(\theta) = \frac{p_{\pi_\theta}(a_t|x_t)}{p_{\pi_{\theta_{\text{old}}}}(a_t|x_t)}$$

and the advantage estimator $\hat{A}_t$ is computed using the latent value function $V_\psi(x_t)$ through GAE:

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l (r_{t+l+1} + \gamma V_\psi(x_{t+l+1}) - V_\psi(x_t))$$

By optimizing this objective, the agent learns to navigate and plan not in the complex, high-dimensional observation space, but on the structured space of the predictive coding map. This separation of representation learning (predictive coding) and behavior learning (PPO) allows the agent to leverage the geometric priors recovered in the map to achieve goal-directed movement with significantly higher sample efficiency and robustness to partial observability. In the following sections, we will demonstrate how this synergy enables complex navigation tasks and systematic exploration.

4.3 Exploration using predictive coding maps

The previous section demonstrated how an agent can achieve robust, goal-directed planning by applying reinforcement learning algorithms, such as Proximal Policy Optimization, to the latent spatial representations recovered by predictive coding. However, this planning capability rests on a critical assumption: that the predictive coding network has already been exposed to

Algorithm 1 Proximal Policy Optimization with Predictive Coding

Input: predictive coder parameters ϕ, initial policy parameters θ_0, initial value function parameters ψ_0.

for $k = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy π_{θ_k} in the environment.

Compute latent states $x_t = f_\phi(o_{\leq t})$ for all t in $\mathcal{D}_k$.

Compute rewards-to-go $\hat{R}_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l+1}$.

Compute advantage estimates $\hat{A}_t$ (e.g., via GAE) based on the current value function $V_{\psi_k}(x_t)$.

Update the policy by maximizing the PPO-Clip objective:

$\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{p_{\pi_\theta}(a_t|x_t)}{p_{\pi_{\theta_k}}(a_t|x_t)} \hat{A}_t, \text{clip}\left(\frac{p_{\pi_\theta}(a_t|x_t)}{p_{\pi_{\theta_k}}(a_t|x_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right)$

typically via stochastic gradient ascent with Adam.

Fit value function by regression on mean-squared error:

$\psi_{k+1} = \arg\min_\psi \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} (V_\psi(x_t) - \hat{R}_t)^2$

typically via some gradient descent algorithm.

end for

sufficient data to construct an accurate, globally consistent map of the environment. In reality, an agent must actively gather this data itself. To build a comprehensive world model, the agent cannot rely on random walks, which scale poorly in complex spaces. Instead, it must systematically seek out novel states—it must *explore*.

As introduced in Chapter I (Sectiona 1.5), efficient mapping can be formalized as a wavefront propagation problem. In this theoretical framework, exploration operates as an iterative four-step cycle: (1) generating observation-action sequences through movement, (2) using predictive coding to recover a latent subgraph of the explored space, (3) propagating a computational wavefront across this latent space to identify the "frontier" between known and unknown states, and (4) executing targeted planning to reach these frontiers and unhide new topological edges. While Chapter I established that this wavefront expansion is guaranteed to converge to a complete isomorphic map of the physical environment, it treated the targeted expansion step as an abstract graph-traversal operation.

In this section, we transition this theoretical graph-traversal concept into a practical, continuous reinforcement learning problem. Rather than relying on discrete graph searches to find frontiers, we can formulate exploration as the optimization of a surrogate objective function, mathematically similar to the

PPO objective. By defining a "curiosity" or intrinsic reward based on the predictive model's prediction error, we can train a policy to naturally seek out the expanding wavefront.

Furthermore, we will demonstrate how to integrate this intrinsic exploration objective directly with the extrinsic policy optimization framework. By balancing the exploitation of known rewards with the systematic discovery of high-uncertainty regions, we can incentivize the agent to learn diverse, robust policies capable of mapping complex, partially observable environments autonomously.

What is exploration?

Before elaborating on practical implementations of exploration using predictive coding maps, we must first precisely define the problem of exploration within the Markov decision process framework. Exploration is fundamentally the process of systematically discovering the structure of an unknown environment to build a comprehensive world model. However, unlike supervised learning where the data distribution is fixed and independent of the model's predictions, the data in reinforcement learning is generated by the agent's own behavior. This closed-loop dependence introduces two primary challenges.

The first challenge is the *bottleneck problem*. Because an agent cannot naively sample the state space uniformly, it must physically navigate through the environment to gather new data. A poor exploratory policy can generate trajectories that become trapped in local, highly-revisited regions—or "bottlenecks"—preventing the discovery of novel states. In the context of the wavefront exploration algorithm introduced in Chapter I, a bottleneck is akin to the wavefront becoming trapped behind a topological obstacle. If the agent randomly walks, the probability of passing through a narrow doorway drops exponentially with time.

The second challenge is the *approximation problem*. In complex, real-world environments, it is impossible to enumerate every trajectory or state. When an agent operates in a continuous or high-dimensional observation space—such as raw RGB pixel inputs—identifying whether two observations correspond to the same underlying physical state becomes highly non-trivial. Without a mechanism to cluster or approximate these observations, an agent might endlessly explore a single room, mistaking sensory noise for uncharted territory,

effectively rendering the wavefront infinite.

To overcome these challenges and successfully implement the wavefront exploration algorithm in continuous Markov decision processes, a practical algorithm must satisfy two critical properties that parallel the theoretical framework.

First, it must systematically *propose paths that escape the bottleneck*. Just as wavefront propagation across the latent subgraph $\hat{G}_k$ calculates the "time of arrival" to identify the shortest paths to the unknown, a practical algorithm must leverage its current internal model to intentionally route the agent out of well-explored subregions, avoiding the inefficiency of redundant sampling.

Second, it must continually *extend the frontier*. Escaping known regions is only useful if it results in the discovery of new topological structure. The agent must execute targeted expansions that push the boundaries of its current world model, generating novel observation-action sequences that "un-hide" new latent edges and nodes. By coupling predictive coding's ability to solve the approximation problem—mapping noisy observations into a coherent geometric structure—with a policy that aggressively seeks the frontier, the agent iteratively propagates the wavefront until the environment is fully mapped.

Mutual information as a measure of exploration

Unsupervised reinforcement learning methods such as Eysenbach *et al.* (2018)[59] and Achiam *et al.* (2018)[60] address the bottleneck problem by generating trajectories that a disjoint from each other. Each trajectory is given a different index c, called a skill. The principle is that the trajectories must be disjoint—that is, each trajectory's states must be distinguishable from each other as well as explore a wide variety of states. As information theoretic formulation, these constraints become

$$J(p_\pi) = \underbrace{I(c;\tau)}_{\text{disjoint trajectories}} + \beta \underbrace{\mathcal{H}(p_\pi|c))}_{\text{random trajectories}} .$$

Using the definition of mutual information, we can rewrite $I(c;\tau)$ in terms of entropy

$$I(c;\tau) = \mathcal{H}(c) - \mathcal{H}(c|\tau).$$

Substitute this back into our objective,

$$J(\pi) = \mathcal{H}(c) - \mathcal{H}(c|\tau) + \beta\mathcal{H}(\pi|c).$$

Because the context c is sampled from a fixed prior distribution G, its initial entropy $\mathcal{H}(c)$ is a constant. Since it cannot be optimized, we drop it from the maximization objective. Thus,

$$\max_{p_\pi} \left[-\mathcal{H}(c|\tau) + \beta \mathcal{H}(p_\pi|c) \right].$$

The conditional entropy $-\mathcal{H}(c|\tau)$ represents the log-probability of the true context given the trajectory. We can rewrite this using expectation notation:

$$-\mathcal{H}(c|\tau) = \mathbb{E}_{c\sim G, \tau\sim\pi, c}[\log p(c|\tau)]$$

However, the true posterior distribution $p(c|\tau)$—the exact probability of a context given a trajectory across the entire environment dynamics—is intractable to compute.

To make this computable, we introduce a learned discriminator network q, which outputs an approximate distribution $q(c|\tau)$. Because the Kullback-Leibler divergence between the true distribution and the approximate distribution is always non-negative ($D_{KL}(p||q) \geq 0$), replacing the true posterior with our discriminator yields a variational lower bound

$$\mathbb{E}[\log p(c|\tau)] \geq \mathbb{E}[\log q(c|\tau)]$$

Instead of maximizing the intractable true probability, we maximize this lower bound by jointly training the policy p_π and the discriminator q. Substituting the expectation and the learned discriminator q back into our objective yields the equation

$$\max_{p_\pi, q} \mathbb{E}_{c\sim G} \left[\mathbb{E}_{\tau\sim\pi, c}[\log q(c|\tau)] + \beta \mathcal{H}(p_\pi|c) \right]$$

In practice, this objective is realized using the discriminator $q_\omega(z|\tau)$ that earns an intrinsic reward $r_t = \log q_\omega(z|\tau) - \log p(z)$ by accurately predicting the active skill from the agent's path. This self-supervised competitive-cooperative loop drives the policy to develop non-overlapping behaviors, directly solving the bottleneck problem by quantifying diversity as identifiability. Consequently, the gradient of the mutual information objective pushes the skills outward to traverse narrow passages and discover uncharted regions, effectively probing the boundaries of the world model and ensuring that the wavefront of discovery is not hindered by local traps.

Detachment and derailment

While mutual information can implicitly encourage an agent to cross topological bottlenecks by rewarding distinct behaviors, they remain vulnerable to two fundamental failure modes of continuous exploration: *detachment* and *derailment*. Detachment occurs when an agent discovers a promising frontier but subsequently abandons it because its intrinsic reward mechanism drives it to explore a different, newly discovered region. Derailment occurs when an agent attempts to return to a previously discovered frontier but fails to reach it because its exploratory policy introduces noise that causes the trajectory to diverge along the way. Both issues prevent systematic expansion and cause the agent to endlessly thrash within known regions, failing to penetrate deep bottlenecks.

Algorithm 2 Go-Explore Algorithm

Require: Environment $\mathcal{E}$, cell mapping function ϕ, selection heuristic h

Initialize archive $\mathcal{A} \leftarrow \emptyset$
Observe starting state s_0 from $\mathcal{E}$
Add s_0 to $\mathcal{A}$ under cell representation $c_0 = \phi(s_0)$

while Evaluation budget is not exhausted **do**
  Select a cell $c \in \mathcal{A}$ based on heuristic h (e.g., favoring unexplored cells)
  Go to the state s associated with c using trajectory replay
  Explore from s for k steps to collect a new trajectory τ
  for each state $s_i \in \tau$ **do**
    Compute cell $c_i = \phi(s_i)$
    if $c_i \notin \mathcal{A}$ **or** trajectory to s_i is better than the stored trajectory for c_i **then**
      Update $\mathcal{A}$ with c_i, s_i, and the new trajectory to s_i
    end if
  end for
end while

The *Go-Explore* algorithm[61] addresses these challenges by explicitly decomposing exploration into a process of returning and exploring, mirroring the mechanics of wavefront propagation. To solve the bottleneck problem, Go-Explore maintains an explicit *archive* of interesting, previously visited states. Instead of relying on a monolithic policy to continuously discover new areas from the starting state, the algorithm iteratively selects a high-potential state from this archive, reliably returns to it, and then executes a brief phase of

exploration to expand the frontier.

By explicitly tracking the most interesting states—often defined by sparsity of visitation, distance from the origin, or heuristic novelty—Go-Explore ensures that the exploration wavefront is continuously pushed forward. This archive acts as a memory of the wavefront's current boundary, completely eliminating the detachment problem because promising frontiers are explicitly saved and never forgotten. Furthermore, by returning to these states without injecting exploratory noise, Go-Explore eliminates derailment. However, a limitation of this algorithm is that it relies on handcrafted heuristics to propose interesting states, which can include the presence of specific objects or its distance from the origin. In most settings, these features are not directly observable. For example, the distance from the origin is not available without access to the environment's coordinates.

The Exploration Algorithm

The theoretical wavefront propagation algorithm introduced in Section 1.5 provides a foundation for mapping discrete, deterministic environments. Here we integrate the insights from the previous sections into an exploration and mapping algorithm using predictive coding. Similar to Eysenbach *et al.* (2018)[59] and Achiam *et al.* (2018)[60], we want to propose trajectories by optimizing the mutual information

$$J(p_\pi) = \underbrace{I(c;\tau)}_{\text{disjoint trajectories}} + \beta \underbrace{\mathcal{H}(p_\pi|c))}_{\text{random trajectories}} .$$

Similar to Section 4.3, the objective function becomes

$$\max_{p_\pi, q} \mathbb{E}_{c\sim G} \left[\mathbb{E}_{\tau\sim\pi, c}[\log q(c|\tau)] + \beta \mathcal{H}(p_\pi|c) \right].$$

However, unlike traditional Markov decision processes, we introduce a predictive coder ϕ that generates latent states from the observations taken through trajectories. The intermediate algorithm becomes

The algorithm must ensure that the wavefront actually progresses without losing access to previous frontiers. We address *detachment* and *derailment* by adopting the "return-and-explore" strategy of Go-Explore. By maintaining an archive of frontier states in the latent space, the agent can reliably return to the boundary of its knowledge without the exploratory noise that would

Algorithm 3 Wavefront exploration using predictive coding (partial)

Initialize: PC parameters ϕ, policy parameters θ, discriminator parameters ω.
for iteration $k = 1, 2, \ldots$ **do**
Roll out exploration policy $\pi_\theta(a|x, z)$ to collect trajectory $\tau = (o_1, a_1, \ldots, o_T)$.
Update PC parameters ϕ to minimize $\mathbb{E}_\tau[-\log p_\phi(o_t|o_{<t})]$.
Compute intrinsic reward $r_t = \log q_\omega(z|\tau) - \log p(z)$.
Update discriminator ω to maximize $\mathbb{E}[\log q_\omega(z|\tau)]$ (identifiability).
Update policy θ to maximize expected return $\mathbb{E}[\sum r_t]$ using PPO.
end for

otherwise cause it to diverge. This structured return ensures that every targeted expansion starts from the current edge of the wavefront, allowing for systematic and deep penetration into unknown territory.

Algorithm 4 Wavefront exploration using predictive coding

Initialize: predictive coder parameters ϕ, policy parameters θ, discriminator parameters ω, and frontier archive $\mathcal{B}$.
for iteration $k = 1, 2, \ldots$ **do**
Identify frontier nodes $\hat{F}_k \in \mathcal{X}$ (states with low visitation or high PC error).
Select a frontier state $x_{goal} \in \hat{F}_k$ and sample a skill code $z \sim p(z)$.
Return the agent to x_{goal} using trajectory replay.
Roll out exploration policy $\pi_\theta(a|x, z)$ to collect trajectory $\tau = (o_1, a_1, \ldots, o_T)$.
Update predictive coder parameters ϕ to minimize $\mathbb{E}_\tau[-\log p_\phi(o_t|o_{<t})]$.
Update inverse dynamics model $p_d(a|x, x')$ to refine latent transitions.
Compute intrinsic reward $r_t = \log q_\omega(z|\tau) - \log p(z)$.
Update discriminator ω to maximize $\mathbb{E}[\log q_\omega(z|\tau)]$ (identifiability).
Update policy θ to maximize expected return $\mathbb{E}[\sum r_t]$ using PPO.
Add newly discovered frontier states from τ to archive $\mathcal{B}$ given the distance from other latent states in the predictive coder ϕ.
end for

The main difference is that the predictive coder implicitly generates unique states in its latent space—even if the observations are ambiguous. The latent states are always identifiable to the policy function p_{π_θ}. In addition, the predictive coder captures the environment's topology, which acts as an implicit heuristic for frontier states. Unexplored latents states far from other latent states are likely to imply that the agent is in a frontier state. If an agent has

not closed a loop trajectory, the latent state would appear to occupy a state from states, but the expansion would eventually close that loop trajectory.

References

55. Sutton, R. S. & Barto, A. G. *Reinforcement Learning, Second Edition: An Introduction* (Bradford Books, Cambridge, Massachusetts London, England, 2020).

56. Schulman, J., Moritz, P., Levine, S., Jordan, M. & Abbeel, P. *High-Dimensional Continuous Control Using Generalized Advantage Estimation* Oct. 2018.

57. Schulman, J., Levine, S., Moritz, P., Jordan, M. I. & Abbeel, P. *Trust Region Policy Optimization* Apr. 2017.

58. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. *Proximal Policy Optimization Algorithms* Aug. 2017.

59. Eysenbach, B., Gupta, A., Ibarz, J. & Levine, S. *Diversity Is All You Need: Learning Skills without a Reward Function* (2026). Pre-published.

60. Achiam, J., Edwards, H., Amodei, D. & Abbeel, P. *Variational Option Discovery Algorithms* (2026). Pre-published.

61. Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O. & Clune, J. *Go-Explore: A New Approach for Hard-Exploration Problems* (2026). Pre-published.

BIBLIOGRAPHY

Sivak, D. A. & Thomson, M. Environmental statistics and optimal regulation. *PLoS computational biology* **10.** e1003826 (2014).

Wang, Z. J. & Thomson, M. Localization of signaling receptors maximizes cellular information acquisition in spatially structured natural environments. *Cell Systems* **13.** 530–546 (2022).

Epstein, R. A., Patai, E. Z., Julian, J. B. & Spiers, H. J. The Cognitive Map in Humans: Spatial Navigation and Beyond. *Nature Neuroscience* **20.** 1504–1513 (Nov. 2017).

Lynen, S. *et al. Get Out of My Lab: Large-scale, Real-Time Visual-Inertial Localization* (July 2015).

Mourikis, A. I. & Roumeliotis, S. I. *A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation* in *Proceedings 2007 IEEE International Conference on Robotics and Automation* (Apr. 2007), 3565–3572.

Mur-Artal, R. & Tardós, J. D. Visual-Inertial Monocular SLAM With Map Reuse. *IEEE Robotics and Automation Letters* **2** (Apr. 2017).

Thrun, S. & Montemerlo, M. The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *The International Journal of Robotics Research* **25.** 403–429 (May 2006).

Thrun, S., Burgard, W. & Fox, D. *Probabilistic Robotics* 647 pp. (MIT Press, Cambridge, Mass, 2005).

Aronov, D., Nevers, R. & Tank, D. W. Mapping of a Non-Spatial Dimension by the Hippocampal–Entorhinal Circuit. *Nature* **543.** 719–722 (Mar. 2017).

Rosenthal, I. A. *et al.* S1 Represents Multisensory Contexts and Somatotopic Locations within and Outside the Bounds of the Cortical Homunculus. *Cell Reports* **42.** 112312 (Apr. 2023).

Huth, A. G., de Heer, W. A., Griffiths, T. L., Theunissen, F. E. & Gallant, J. L. Natural Speech Reveals the Semantic Maps That Tile Human Cerebral Cortex. *Nature* **532.** 453–458 (Apr. 2016).

Garvert, M. M., Dolan, R. J. & Behrens, T. E. A Map of Abstract Relational Knowledge in the Human Hippocampal–Entorhinal Cortex. *eLife* **6** (ed Davachi, L.) e17086 (Apr. 2017).

Constantinescu, A. O., O'Reilly, J. X. & Behrens, T. E. J. Organizing Conceptual Knowledge in Humans with a Gridlike Code. *Science* **352.** 1464–1468 (June 2016).

Nieh, E. H. *et al.* Geometry of Abstract Learned Knowledge in the Hippocampus. *Nature* **595.** 80–84 (July 2021).

Corkin, S. Lasting Consequences of Bilateral Medial Temporal Lobectomy: Clinical Course and Experimental Findings in H.M. *Seminars in Neurology* **4.** 249–259 (June 1984).

Behrens, T. E. J. *et al.* What Is a Cognitive Map? Organizing Knowledge for Flexible Behavior. *Neuron* **100.** 490–509 (Oct. 2018).

Whittington, J. C., McCaffary, D., Bakermans, J. J. & Behrens, T. E. How to build a cognitive map. *Nature neuroscience* **25.** 1257–1272 (2022).

Rescorla, M. Cognitive maps and the language of thought. *The British Journal for the Philosophy of Science* (2009).

Anderson, J. *Cognitive Psychology and Its Implications* Ninth edition (Worth Publishers, New York City, Jan. 2020).

Gornet, J. & Thomson, M. Automated Construction of Cognitive Maps with Visual Predictive Coding. *Nature Machine Intelligence* **6.** 820–833 (July 2024).

Duan, Y. *et al.* RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning (Nov. 2016).

Mirowski, P. *et al. Learning to Navigate in Cities Without a Map* in *Advances in Neural Information Processing Systems* **31** (Curran Associates, Inc., 2018).

Gupta, S., Davidson, J., Levine, S., Sukthankar, R. & Malik, J. Cognitive Mapping and Planning for Visual Navigation. 10.

Yao, S. *et al. ReAct: Synergizing Reasoning and Acting in Language Models* (2026). Pre-published.

Wang, L. *et al. Large Action Models: From Inception to Implementation* (2026). Pre-published.

Zhang, C. *et al. AppAgent: Multimodal Agents as Smartphone Users* (2026). Pre-published.

Nguyen, D. *et al. GUI Agents: A Survey* (2026). Pre-published.

Zhang, X., Li, G., Shokouhi, S. & Thein, M.-W. L. Modern Control Meets Machine Learning: A Review and Taxonomy of Synergistic Approaches for Robotics Applications. *Actuators* **15.** 235 (May 2026).

Brown, T. B. *et al.* Language Models Are Few-Shot Learners. *arXiv:2005.14165 [cs]* (July 2020).

Vaswani, A. *et al. Attention Is All You Need* Aug. 2023.

Felicia, G. *et al. From Perception to Action: Spatial AI Agents and World Models* (2026). Pre-published.

Maes, L., Lidec, Q. L., Scieur, D., LeCun, Y. & Balestriero, R. *LeWorldModel: Stable End-to-End Joint-Embedding Predictive Architecture from Pixels* (2026). Pre-published.

Yang, S. *World Models as an Intermediary between Agents and the Real World* (2026). Pre-published.

Xu, M. *et al. Kinema4D: Kinematic 4D World Modeling for Spatiotemporal Embodied Simulation* (2026). Pre-published.

Markowitz, J. E. *et al.* Spontaneous Behaviour Is Structured by Reinforcement without Explicit Reward. *Nature* **614.** 108–117 (Feb. 2, 2023).

Lindsey, J. W., Markowitz, J., Gillis, W. F., Datta, S. R. & Litwin-Kumar, A. Dynamics of Striatal Action Selection and Reinforcement Learning. *eLife* **13.** RP101747 (May 8, 2025).

Rabin, M. O. & Scott, D. Finite Automata and Their Decision Problems. *IBM Journal of Research and Development* **3.** 114–125 (Apr. 1959).

Nerode, A. Linear Automaton Transformations. *Proceedings of the American Mathematical Society* **9.** 541–544 (1958).

Crane, K., Weischedel, C. & Wardetzky, M. The Heat Method for Distance Computation. *Communications of the ACM* **60.** 90–99 (Oct. 24, 2017).

Sutton, R. S. & Barto, A. G. *Reinforcement Learning, Second Edition: An Introduction* 552 pp. (Bradford Books, Cambridge, Massachusetts, 2018).

Dieck, T. T. *Algebraic Topology* (American Mathematical Society, Zürich, 2008).

May, J. P. *A Concise Course in Algebraic Topology* (University of Chicago Press, Sept. 1999).

Hatcher, A. *Algebraic Topology* (Cambridge University Press, Cambridge ; New York, 2002).

Shannon, C. E. A Mathematical Theory of Communication. *The Bell System Technical Journal* **27.** 379–423 (July 1948).

Johnson, M., Hofmann, K., Hutton, T. & Bignell, D. *The Malmo Platform for Artificial Intelligence Experimentation* in *International Joint Conference on Artificial Intelligence* (2016).

He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]* (Dec. 2015).

Ronneberger, O., Fischer, P. & Brox, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation* May 2015.

Sutskever, I., Martens, J., Dahl, G. & Hinton, G. On the Importance of Initialization and Momentum in Deep Learning.

Smith, L. N. & Topin, N. *Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates* May 2018.

Tenenbaum, J. B., de Silva, V. & Langford, J. C. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science* **290.** 2319–2323 (Dec. 2000).

Bush, D., Barry, C., Manson, D. & Burgess, N. Using Grid Cells for Navigation. *Neuron* **87.** 507–520 (Aug. 5, 2015).

Behrisch, M., Kerkhoff, S., Pöschel, R., Schneider, F. M. & Siegmund, S. Dynamical Systems in Categories. *Applied Categorical Structures* **25.** 29–57 (Feb. 2017).

Vries, J. de. *Topological Dynamical Systems* 498 pp. (De Gruyter, Berlin ; Boston, 2014).

Sakai, S. *Operator Algebras in Dynamical Systems* 232 pp. (Cambridge University Press, Cambridge England New York, 1991).

Sutton, R. S. & Barto, A. G. *Reinforcement Learning, Second Edition: An Introduction* (Bradford Books, Cambridge, Massachusetts London, England, 2020).

Schulman, J., Moritz, P., Levine, S., Jordan, M. & Abbeel, P. *High-Dimensional Continuous Control Using Generalized Advantage Estimation* Oct. 2018.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I. & Abbeel, P. *Trust Region Policy Optimization* Apr. 2017.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. *Proximal Policy Optimization Algorithms* Aug. 2017.

Eysenbach, B., Gupta, A., Ibarz, J. & Levine, S. *Diversity Is All You Need: Learning Skills without a Reward Function* (2026). Pre-published.

Achiam, J., Edwards, H., Amodei, D. & Abbeel, P. *Variational Option Discovery Algorithms* (2026). Pre-published.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O. & Clune, J. *Go-Explore: A New Approach for Hard-Exploration Problems* (2026). Pre-published.