Probabilistic learning and computation in brains and machines

Eszter Vértes

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Gatsby Computational Neuroscience Unit

University College London

2020

I, Eszter Vértes, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Humans and animals are able to solve a wide variety of perceptual, decision making and motor tasks with great flexibility. Moreover, behavioural evidence shows that this flexibility extends to situations where accuracy requires the correct treatment of uncertainty induced by noise and ambiguity in the available sensory information as well as noise internal to the brain. It has been suggested that this adequate handling of uncertainty is based on a learned internal model, e.g. in the case of perception, a generative model of sensory observations.

Learning latent variable models and performing inference in them is a key challenge for both biological and artificial learning systems. Here, we introduce a new approach to learning in hierarchical latent variable models called the Distributed Distributional Code Helmholtz Machine (DDC-HM), which emphasises flexibility and accuracy in the inferential process. The approximate posterior over unobserved variables is represented implicitly as a set of expectations, corresponding to mean parameters of an exponential family distribution. To train the generative and recognition models we develop an extended wake-sleep algorithm inspired by the original Helmholtz Machine. As a result, the DDC-HM is able to learn hierarchical latent models without having to propagate gradients across different stochastic layers—making our approach biologically appealing.

In the second part of the thesis, we review existing proposals for neural representations of uncertainty with a focus on representational and computational flexibility as well as experimental support. Finally, we consider inference and learning in dynamical environment models using Distributed Distributional

Codes to represent both the stochastic latent transition model and the inferred posterior distributions. We show that this model makes it possible to generalise successor representations to biologically more realistic, partially observed settings.

Impact Statement

We expect the thesis to have a direct impact on fundamental research in the fields of machine learning and neuroscience. In particular, the distributed distributional code Helmholtz machine, the algorithm we have introduced in chapter 2, presents a novel perspective on learning hierarchical generative models. Unlike most earlier methods, it places great emphasis on the inferential process and on accurately capturing uncertainty about latent variables. Generative latent variable models have numerous existing and potential applications from image processing, speech synthesis and recognition to medical diagnostics. In many of these applications, quantifying uncertainty about unobserved variables is essential when an algorithm is used in real-world scenarios.

Another area where the thesis should have a significant impact is theoretical and systems neuroscience. The work presented here contributes to our understanding of how we learn to reason about relevant but not directly observable variables in the world or how we make decisions in the face of uncertainty. While we have studied these questions from a computational and algorithmic perspective, generating hypothesis on these levels is key towards a more mechanistic understanding of how brains carry out these computations. Furthermore, our discussion on the origins and representations of uncertainty in the brain, clarifying the relationship between concepts like noise, variability and uncertainty, should facilitate a more nuanced treatment of the topic in future publications.

Finally, we hope that this work will also inspire further research at the intersection of these two fields that have a lot to benefit from one another.

Acknowledgements

I would like to thank my supervisor, Maneesh Sahani, for his continued support and guidance throughout my PhD. Especially early on, I have benefitted greatly from his rare skill of understanding my questions before I did. He listened carefully and showed enthusiasm for all my projects and ideas (even the very last minute ones) and that gave me freedom and confidence to explore questions I found the most exciting.

The Gatsby has been a unique, intellectually stimulating and socially welcoming environment and I learned a great deal from fellow students, postdocs and faculty. Special thanks to Wittawat and Michael for answering my countless questions about kernels.

I feel extremely lucky that I have made several close friends during my time here, had so much fun while travelling, going on hiking or climbing adventures together or simply sharing stories over a pint. Joana, Sofy, Alex, Kirsty, Lea, Billy, Bill and many others made these years truly memorable.

I am immensely grateful for Antonin for taking care of me in this past year (especially after my surgery), and always making sure I had everything I needed as a chef or personal assistant.

Finally, I'd like to thank my family for their love and support, and for their patience when I repeatedly tried (and often failed) to explain what my PhD would be about.

Contents

List of Figures

List of Algorithms

Nomenclature

Abbreviations

DAG	directed acyclic graph
DDC	distributed distributional code
DDPC	doubly distributional population codes
DPC	distributional population codes
EM	expectation maximisation
GSM	Gaussian scale mixture
HM	Helmholtz machine
ICA	independent component analysis
IWAE	importance weighted variational autoencoder
KDE	kernel density estimation
KL	Kullback–Leibler divergence
MDP	Markov decision process
MMD	maximum mean discrepancy
ORN	olfactory receptor neuron
POMDP	partially observable Markov decision process

PPC	probabilistic population code
RKHS	reproducing kernel Hilbert space
RL	reinforcement learning
SBN	sigmoid belief net
SF	successor feature
SR	successor representation
TD	temporal-difference
V1	primary visual cortex
VAE	variational autoencoder
VIMCO	variational inference for Monte Carlo objectives

Preface

Conscious of the rather broad topics in the title, here we start by describing the scope of the thesis in more detail. The main theme explored throughout the thesis is how to learn probabilistic models that serve as a statistical description of the world and how to invert these models to be able to reason about latent variables given available observations. This is a pivotal problem in both machine learning and neuroscience as both biological and artificial intelligent systems need to operate in a partially observable, *uncertain* world.

The thesis is structured as follows: in Part I, we introduce the *distributed distributional code* Helmholtz machine, an algorithm for learning hierarchical latent variable models that allows for accurate inference through a flexible exponential family posterior representation (chapter 2). In chapter 3, we generalise the algorithm to reproducing kernel Hilbert spaces and we present an analysis of convergence.

In Part II, we discuss origins of uncertainty in the context of neuroscience and the Bayesian brain hypothesis; we present a critical review of existing theoretical proposals for neural representations of uncertainty (chapter 4). Finally, we show how distributed distributional codes can generalise successor representations to partially observable environments and discuss a variety of experimental observations unified under our model.

Most of the results from chapter 2 on the DDC Helmholtz machine were published in Vértes and Sahani (2018), while chapter 5 on distributional successor features was published in Vértes and Sahani (2019).

Part I

Learning and inference in hierarchical generative models

Chapter 1

Introduction

There is substantial interest in applying variational methods to learn complex latent-variable generative models, for which the full likelihood function (after marginalising over the latent variables) and its gradients are intractable. Unsupervised learning of such models has two complementary goals: to learn a good approximation to the distribution of the observations; and also to learn the underlying structural dependence so that the values of latent variables may be inferred from new observations.

Variational methods rely on optimising a lower bound to the log-likelihood (the *free energy*), which depends on an approximation to the posterior distribution over the latents (Wainwright and Jordan, 2008). The performance of variational algorithms depends critically on the flexibility of the variational posterior. In cases where the approximating class does not contain the true posterior distribution, variational learning may introduce substantial bias to estimates of model parameters (Turner and Sahani, 2011).

Variational autoencoders (Rezende et al., 2014; Kingma et al., 2014) combine the variational inference framework with the earlier idea of the recognition network. This approach has made variational inference applicable to a large class of complex generative models. However, many challenges remain. Most current algorithms have difficulty learning hierarchical generative models with multiple layers of stochastic latent variables (Sønderby et al., 2016). Arguably, this class of models is crucial for modelling data where the underlying physical

process is itself hierarchical in nature. Furthermore, the generative models typically considered in the literature restrict the prior distribution to a simple form, most often a factorised Gaussian distribution, which makes it difficult to incorporate additional generative structure such as sparsity into the model.

We introduce a new approach to learning hierarchical generative models, the *Distributed Distributional Code (DDC) Helmholtz Machine*, which combines two ideas that originate in theoretical neuroscience: the Helmholtz Machine with wake-sleep learning (Dayan et al., 1995); and distributed or population codes for distributions (Zemel et al., 1998; Sahani and Dayan, 2003). A key element of our method is that the approximate posterior distribution is represented as a set of expected sufficient statistics, rather than by directly parametrising the probability density function. This allows an accurate posterior approximation without being restricted to a rigid parametric class. At the same time, the DDC Helmholtz Machine retains some of the simplicity of the original Helmholtz Machine in that it does not require propagating gradients across different layers of latent variables. The result is a robust method able to learn the parameters of each layer of a hierarchical generative model with far greater accuracy than achieved by current variational methods.

Here, we begin by briefly reviewing variational learning (section 1.1), deep exponential family models (section 1.2), and the original Helmholtz Machine and wake-sleep algorithm (section 1.3) relevant for our discussion of the DDC Helmholtz machine in chapter 2. Then, we introduce reproducing kernel Hilbert spaces and kernel mean embeddings, concepts that will allow us to generalise the DDC Helmholtz machine algorithm to infinite dimensional spaces (chapter 3).

1.1 Variational inference and learning in latent variable models

In most latent variable models of interest maximum likelihood learning is intractable since the objective requires evaluating the following integral:

$$\log p_\theta(x) = \log \int p_\theta(x, z) dz \tag{1.1}$$

where x and z are the observed and unobserved variables, respectively, and $p_\theta(x, z)$ is the joint density parametrised by θ. With a few exceptions, such as linear Gaussian models, the integral in eq. 1.1 has no analytic solution.

Variational methods (Jordan et al., 1999; Wainwright and Jordan, 2008) rely on optimising a lower bound $\mathcal{F}$ on the log-likelihood by introducing an approximate posterior distribution $q(z|x)$ over the latent variables:

$$\begin{aligned} \log p_\theta(x) \geq \mathcal{F}(q, \theta, x) &= \langle \log p_\theta(x, z) \rangle_{q(z|x)} + \mathrm{H}[q(z|x)] \\ &= \log p_\theta(x) - D_{KL}[q(z|x)||p_\theta(z|x)] \end{aligned} \tag{1.2}$$

where $\mathrm{H}[q(z|x)]$ is the entropy of q, D_{KL} is the Kullback-Leibler (KL) divergence between the approximate and true posterior distributions. $\mathcal{F}(q, \theta, x)$ is referred to as the negative variational free energy and can be jointly maximised with respect to the generative model parameters θ and the approximate posterior $q(z|x)$.

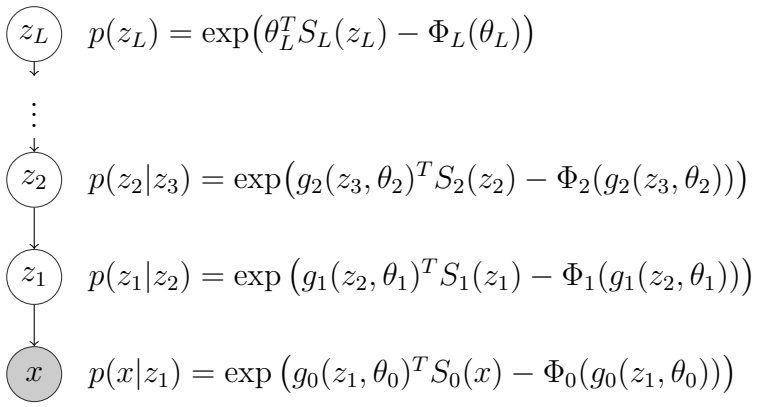
The cost of computing the posterior approximation for each observation can be efficiently amortised by using a recognition model (Gershman and Goodman, 2014), an explicit function (with parameters ϕ, often a neural network) that for each x returns the parameters (η) of an estimated posterior distribution: $x \mapsto \eta$ so that $q_\phi(z|x) = q(z; \eta_\phi(x))$.

For tractability, the approximate posterior is typically chosen to be a member of a simple parametric form, for example a factorised Gaussian distribution. The resulting mismatch between the approximate and exact posterior distributions introduces a bias in variational learning of the parameters θ. The variational objective penalises this error using the 'exclusive' Kullback-Leibler divergence (between the approximate and true posterior distributions, see eq. 1.2), which typically results in an approximation that underestimates the posterior uncertainty (Minka, 2005).

Multi-sample objectives (IWAE Burda et al. (2015); VIMCO Mnih and Rezende (2016)) have been proposed to remedy the disadvantages of a restrictive posterior approximation. Nonetheless, benefits of these methods are limited in cases when the proposal distribution is too far from the true posterior (see section 2.3).

1.2 Deep exponential family models

We consider hierarchical generative models in which each conditional distribution belongs to an exponential family, also known as *deep exponential family models* (Ranganath et al., 2015). Let $x \in \mathcal{X}$ denote a single (vector) observation. The distribution of data x is determined by a sequence of L latent variables $z_1 \in \mathbb{R}^{D_1}, \ldots . z_L \in \mathbb{R}^{D_L}$ arranged in a conditional hierarchy as follows:



Each conditional distribution is a member of a *tractable* exponential family, so that conditional sampling is possible. Using $l \in \{0, 1, 2, \ldots L\}$ to denote the layer (with $l = 0$ for the observation, $x \in \mathbb{R}^{D_0}$), these distributions have sufficient statistic function S_l, natural parameter given by a known function g_l of both the parent variable and a parameter vector θ_l, and a log normaliser Φ_l that depends on this natural parameter. At the top layer, we lose no generality by taking $g_L(\theta_L) = \theta_L$.

We will maintain the general notation here, as the method we propose is very broadly applicable (both to continuous and discrete latent variables), provided that the family remains tractable in the sense that we can efficiently sample from the conditional distributions given the natural parameters.

1.3 The classic Helmholtz Machine and the wake-sleep algorithm

The Helmholtz Machine (HM; Dayan et al., 1995) comprises a latent-variable generative model that is to be fit to data, and a *recognition* network, trained to perform approximate inference over the latent variables.

The latent variables of an HM generative model are arranged hierarchically in a directed acyclic graph, with the variables in a given layer conditionally independent of one another given the variables in the layer above. In the original HM, all latent and observed variables were binary and formed a sigmoid belief network (Neal, 1992), which is a special case of deep exponential family models introduced in the previous section with $S_l(z_l) = z_l$ and $g_l(z_{l+1}, \theta_l) = \theta_l z_{l+1}$.

$$\begin{aligned} p(z_L) &= \text{Bernoulli}(\theta_L) \\ p(z_{L-1}|z_L) &= \text{Bernoulli}(\sigma(\theta_{L-1} z_L)) \\ &\vdots \\ p(x|z_1) &= \text{Bernoulli}(\sigma(\theta_0 z_1)) \end{aligned} \tag{1.3}$$

The recognition network is a functional mapping with an analogous hierarchical architecture that takes each x to an estimate of the posterior probability of each z_l, using a factorised mean-field representation, i.e. $q(z_l|x) = \Pi_i q(z_{l,i}|x)$, where i is indexing the elements of the vector z_l.

The training of both generative model and recognition network follows a two-phase procedure known as *wake-sleep* (Hinton et al., 1995). In the *wake* phase, observations x are fed through the recognition network to obtain the posterior approximation $q_\phi(z_l|x)$. In each layer the latent variables are sampled independently conditioned on the samples of the layer below according to the probabilities determined by the recognition model parameters. These samples are then used to update the generative parameters to increase the expected joint likelihood – equivalent to taking gradient steps to increase the variational free energy. In the *sleep* phase, the current generative model is used to provide

joint samples of the latent variables and fictitious (or "dreamt") observations and these are used as supervised training data to adapt the recognition network. The algorithm allows for straightforward optimisation since parameter updates at each layer in both the generative and recognition models are based on locally generated samples of both the input and output of the layer.

Despite the resemblance to the two-phase process of expectation-maximisation and approximate variational methods, the sleep phase of wake-sleep does not necessarily increase the free-energy bound on the likelihood (Dayan, 2000). Even in the limit of infinite samples, the mean field representation $q_\phi(z|x)$ is learnt so that it minimises $D_{KL}[p_\theta(z|x)\|q_\phi(z|x)]$, rather than $D_{KL}[q_\phi(z|x)\|p_\theta(z|x)]$ as required by variational learning (see eq. 1.2). For this reason, the mean-field approximation provided by the recognition model is particularly limiting, since it not only biases the learnt generative model (as in the variational case) but it may also preclude convergence.

1.4 Reproducing kernel Hilbert spaces

Here, we give a definition of reproducing kernel Hilbert spaces (RKHS) and universal kernels—concepts that we will use extensively in our discussion of the Kernel Helmholtz machine in chapter 3.

Definition 1. *(RKHS) Consider a non-empty set $\mathcal{Z}$. A reproducing kernel Hilbert space $\mathcal{H}_\mathcal{Z}$ with kernel $k_\mathcal{Z} : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ is a Hilbert space of real-valued functions $f : \mathcal{Z} \to \mathbb{R}$. Its dot product $\langle ., . \rangle_{\mathcal{H}_\mathcal{Z}}$ satisfies the reproducing property:*

$$\forall z \in \mathcal{Z} : k_\mathcal{Z}(z, .) \in \mathcal{H}_\mathcal{Z}, \tag{1.4}$$

$$\forall z \in \mathcal{Z} \text{ and } \forall f \in \mathcal{H}_\mathcal{Z} : \langle f, k_\mathcal{Z}(z, .)\rangle_{\mathcal{H}_\mathcal{Z}} = f(z). \tag{1.5}$$

That is, the kernel function evaluated at one of its arguments $k_\mathcal{Z}(z, .)$ is a function in the RKHS $\mathcal{H}_\mathcal{Z}$ and we can represent the evaluation of a function at any $z \in \mathcal{Z}$ as an inner product in $\mathcal{H}_\mathcal{Z}$. It also follows that $\langle k(z, .), k(z', .)\rangle_\mathcal{H} = k(z, z')$. The function $k_\mathcal{Z}(z, .)$ can also be viewed as a

feature map $\phi(z) \in \mathcal{H}_{\mathcal{Z}}$ where $k(z, z') = \langle \phi(z), \phi(z') \rangle_{\mathcal{H}_{\mathcal{Z}}}$.

Definition 2. *(Universal kernel). Consider a compact metric space* $\mathcal{Z}$ *and continuous kernel* $k_{\mathcal{Z}} : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$. $k_{\mathcal{Z}}$ *is universal (or c-universal) if its associated RKHS,* $\mathcal{H}_{\mathcal{Z}}$ *is dense in the Banach space,* $C(\mathcal{Z})$ *of real-valued continuous functions (on* $\mathcal{Z}$ *) w.r.t. the uniform norm, i.e., for any* $f^\star \in C(\mathcal{Z})$*, there exists* $g \in \mathcal{H}_{\mathcal{Z}}$ *that uniformly approximates* $f^\star$.

Note that this definition of universality is limited to functions defined over compact spaces excluding many interesting spaces like $\mathbb{R}^d$. To address this limitation, the definition has been generalised to functions over non-compact spaces that vanish at infinity (Sriperumbudur et al., 2010). Examples of such c_0-*universal* kernels on $\mathbb{R}^d$ include the Gaussian $k(z, z') = \exp(-\alpha \|z - z'\|_2^2), \alpha > 0$ or Laplacian kernels $k(z, z') = \exp(-\alpha \|z - z'\|_1), \alpha > 0$.

1.4.1 Vector valued RKHSs

The definition of scalar-valued RKHS has been extended to the space of vector valued functions (Micchelli and Pontil, 2005). Here, we give the definition and review some of the properties of vector valued RKHSs that we will later use to analyse convergence of the Kernel Helmholtz machine.

Consider a non-empty set $\mathcal{X}$ and a Hilbert space $(\mathcal{V}, \langle ., . \rangle_{\mathcal{V}})$.

Definition 3. *A Hilbert space* $(\mathcal{H}_\Gamma, \langle ., . \rangle_\Gamma)$ *of (vector valued) functions* $f : \mathcal{X} \to \mathcal{V}$ *is an RKHS if for all* $x \in \mathcal{X}, v \in \mathcal{V}$ *the linear functional* $f \to \langle v, f(x) \rangle_{\mathcal{V}}$ *is continuous.*

The Riesz representer theorem implies that for each $x \in \mathcal{X}$ and $v \in \mathcal{V}$ there is a linear operator $\Gamma_x : \mathcal{V} \to \mathcal{H}_\Gamma$, such that for all $f \in \mathcal{H}_\Gamma$

$$\langle v, f(x) \rangle_{\mathcal{V}} = \langle f, \Gamma_x v \rangle_\Gamma. \tag{1.6}$$

The corresponding reproducing kernel in $\mathcal{H}_\Gamma$, $\Gamma(x, x')$ is a bounded linear

operator $\mathcal{V} \to \mathcal{V}$ and defined as:

$$\Gamma(x, x')v = (\Gamma_{x'}v)(x) \in \mathcal{V}. \tag{1.7}$$

1.5 Kernel mean embedding of distributions

Kernel mean embeddings define a representation of a probability distribution $\mathcal{P}$ in an RKHS as the expected kernel function, that is:

$$\mu_{\mathcal{P}} := \int_{\mathcal{Z}} k_{\mathcal{Z}}(z, .) d\mathcal{P}(z) \tag{1.8}$$

where $k_{\mathcal{Z}}(z, .) \in \mathcal{H}_{\mathcal{Z}}$ and $\mu_{\mathcal{P}} \in \mathcal{H}_{\mathcal{Z}}$ if $\mathbb{E}_{z\sim\mathcal{P}}[k_{\mathcal{Z}}(z, z)] < \infty$. Importantly, for *characteristic* kernels $k_{\mathcal{Z}}(z, z')$, the kernel mean embedding captures all information about the distribution $\mathcal{P}$.

Definition 4. *(Characteristic kernel). Consider a topological space $\mathcal{Z}$, a measurable, bounded kernel $k_{\mathcal{Z}}$ and a Borel probability measure $\mathcal{P}$ on $\mathcal{Z}$. The kernel $k_{\mathcal{Z}}$ is characteristic if and only if the embedding $\mathcal{P} \to \int_{\mathcal{Z}} k_{\mathcal{Z}}(z, .) d\mathcal{P}(z)$ is injective.*

The injective property of characteristic kernels implies that different distributions are mapped to different points in the RKHS: $\mu_{\mathcal{P}} = \mu_{\mathcal{P}'}$ if and only if $\mathcal{P} = \mathcal{P}'$ (Fukumizu et al., 2003). The characteristic property is closely related to universality of kernels (Sriperumbudur et al., 2010), for an in-depth review of kernel mean embeddings see Muandet et al. (2017).

The kernel mean embedding $\mu_{\mathcal{P}}$ can be used to compute expectations of functions $h \in \mathcal{H}_{\mathcal{Z}}$ with respect to $\mathcal{P}$ by an inner product in $\mathcal{H}_{\mathcal{Z}}$:

$$\mathbb{E}_{p(z)}[h(z)] = \langle h, \mu_{\mathcal{P}} \rangle_{\mathcal{H}_{\mathcal{Z}}} \quad \forall h \in \mathcal{H}_{\mathcal{Z}} \tag{1.9}$$

Similarly, conditional expectations can also be evaluated as an inner product:

$$\mathbb{E}_{p(z|x)}[h(z)] = \langle h, \mu(x) \rangle_{\mathcal{H}_{\mathcal{Z}}} \quad \forall h \in \mathcal{H}_{\mathcal{Z}}, x \in \mathcal{X} \tag{1.10}$$

where $\mu(x) \in \mathcal{H}_\mathcal{Z}$ is the conditional mean embedding. Note that μ is a mapping $\mathcal{X} \to \mathcal{H}_\mathcal{Z}$, while $\mu(x) \in \mathcal{H}_\mathcal{Z}$, i.e. an RKHS function on $\mathcal{Z}$.

For a general RKHS $\mathcal{H}_\mathcal{Z}$, the conditional embedding μ can be estimated by computing the operator $\mathcal{U}_{z|x} : \mathcal{H}_\mathcal{X} \to \mathcal{H}_\mathcal{Z}$ for which $\mu(x) = \mathcal{U}_{z|x} k_\mathcal{X}(x, .)$ as described in Song et al. (2009). The corresponding empirical estimator $\hat{\mathcal{U}}_{z|x}$ converges with rate $O(S^{-1/4})$ with the number of samples S.

Assuming that $\mathcal{H}_\mathcal{Z}$ is a finite dimensional RKHS, the conditional mean embedding can be approximated in a vector valued RKHS $\mathcal{H}_\Gamma$ with a much faster convergence rate of $O(\log S/S)$ (Grünewälder et al., 2012). Given a real valued kernel $k_\mathcal{X}(x, x')$ on $\mathcal{X}$, the kernel for $\mathcal{H}_\Gamma$ can be defined as $\Gamma(x, x') = k_\mathcal{X}(x, x') I_{\mathcal{H}_\mathcal{Z}}$, where $I_{\mathcal{H}_\mathcal{Z}}$ is the identity operator $\mathcal{H}_\mathcal{Z} \to \mathcal{H}_\mathcal{Z}$. This choice corresponds to a space isomorphic to $\mathcal{H}_\mathcal{X} \times \mathcal{H}_\mathcal{Z}$.

Chapter 2

The Distributed Distributional Code Helmholtz Machine

2.1 Distributed Distributional Codes

The key drawback of both the classic Helmholtz machine and most approximate variational methods is the need for a tractably parametrised posterior approximation. Our contribution is to instead adopt a flexible and powerful representation of uncertainty in terms of expected values of large set of (possibly random) arbitrary nonlinear functions. We call this representation a Distributed Distributional Code (DDC) in acknowledgement of its history in theoretical neuroscience (Zemel et al., 1998; Sahani and Dayan, 2003). In the DDC-HM, each posterior is represented by approximate expectations of non-linear *encoding functions* $\{\psi_i(z)\}_{i=1...K_l}$ with respect to the *true posterior* $p_\theta(z_l|x)$:

$$r_l^{(i)}(x,\phi) \approx \left\langle \psi_i(z_l) \right\rangle_{p_\theta(z_l|x)}, \tag{2.1}$$

where $r_l^{(i)}(x,\phi), i = 1...K_l$ is the output of the recognition network (parametrised by ϕ) at the lth latent layer, and the angle brackets denote expectations. A finite—albeit large—set of expectations does not itself fully specify the probability distribution $p_\theta(z_l|x)$. Thus, the recognition outputs $\{r_l^{(i)}\}_{i=1...K_l}$ are interpreted as representing an approximate posterior $q(z_l|x)$ defined by the distribution of maximum entropy that agrees with all of the encoded

expectations.

A standard calculation shows that this distribution has a density of the form (Wainwright and Jordan, 2008, Ch.3):

$$q(z_l|x) = \frac{1}{Z(\eta(x))} \exp\left(\sum_{i=1}^{K_l} \eta_i(x)\psi_i(z_l)\right) \tag{2.2}$$

where the η_i are natural parameters (derived as Lagrange multipliers enforcing the expectation constraints), and $Z(\eta)$ is a normalising constant. Thus, in this view, the encoded distribution q is a member of the exponential family whose sufficient statistic functions correspond to the encoding functions $\{\psi_i(z_l)\}$, and the recognition network returns the expected sufficient statistics, or *mean parameters*. Results on function approximation with random bases (Rahimi and Recht, 2008) suggest that given a sufficiently large set of encoding functions, we can approximate the true posterior distribution arbitrarily well. Throughout this chapter we will use encoding functions of the following form:

$$\psi_i(z_l) = \sigma(w^{(i)T} z_l + b^{(i)}),\ i = 1 \dots K_l\,, \tag{2.3}$$

where $w^{(i)}$ is a random linear projection with components sampled from a standard normal distribution, $b^{(i)}$ is a similarly distributed bias term, and σ is the logistic sigmoid function. That is, the representation is designed to capture information about the posterior distribution along K random projections in z-space. As a special case, we can recover the approximate posterior equivalent to the original HM if we consider linear encoding functions $\psi_i(z_l) = [z_l]_i$ (individual components of the vector z_l), corresponding to a factorised mean-field approximation.

Obtaining the posterior natural parameters $\{\eta^{(i)}\}$ (and thus evaluating the density in Eq. 2.2) from the mean parameters $\{r^{(i)}\}$ is not straightforward in the general case since $Z(\eta)$ is intractable. Thus, it is not immediately clear how a DDC representation can be used for learning. Our exact scheme will be developed below, but in essence it depends on the simple observation that most

of the computations necessary for learning (and indeed most computations involving uncertainty) depend on the evaluation of appropriate expectations. Given a rich set of encoding functions $\{\psi_i\}_{i=1\ldots K}$ sufficient to approximate a desired function f using linear weights $\{\alpha_i\}$, such expectations become easy to evaluate in the DDC representation:

$$f(z) \approx \sum_i \alpha^{(i)} \psi_i(z) \tag{2.4}$$

$$\Rightarrow \quad \langle f(z) \rangle_{q(z)} \approx \sum_i \alpha^{(i)} \langle \psi_i(z) \rangle_{q(z)} = \sum_i \alpha^{(i)} r^{(i)} \tag{2.5}$$

Thus, the richer the family of DDC encoding functions, the more accurate are both the approximated posterior distribution, *and* the approximated expectations[1]. We will make extensive use of this property in the following section where we discuss how this posterior representation is learnt (sleep phase) and how it can be used to update the generative model (wake phase).

2.2 The DDC Helmholtz Machine algorithm

Algorithm 1 DDC Helmholtz Machine training

Initialise θ
repeat
  Sleep phase:
  for $s = 1 \ldots S$, sample: $z_L^{(s)}, ..., z_1^{(s)}, x^{(s)} \sim p_\theta(x, z_1, ..., z_L)$
  update recognition parameters $\{\phi_l\}$ [eq. 2.7]
  update function approximators $\{\alpha_l, \beta_l\}$ [appendix]
  Wake phase:
  $x \leftarrow$ {minibatch}
  evaluate $r_l(x, \phi)$ [eq. 2.8]
  update θ: $\Delta\theta \propto \widehat{\nabla_\theta \mathcal{F}}(x, r(x, \phi), \theta)$ [appendix]
until $|\widehat{\nabla_\theta \mathcal{F}}| <$ threshold

Following (Dayan et al., 1995) the generative and recognition models in the DDC-HM are learnt in two separate phases (see Algorithm 1). The sleep phase involves learning a recognition network that takes data points x as

[1] In a suitable limit, an infinite family of encoding functions would correspond to a mean embedding representation in a reproducing kernel Hilbert space (Gretton et al., 2012)

input and produces expectations of the non-linear encoding functions $\{\psi_i\}$ as given by Eq. (2.1); *and* learning how to use these expectations to update the generative model parameters using approximations of the form of Eq. (2.4). The wake phase updates the generative parameters by computing the approximate gradient of the free energy, using the posterior expectations learned in the sleep phase. Below we describe the two phases of the algorithm in more detail.

2.2.1 Sleep phase

One aim of the sleep phase, given a current generative model $p_\theta(x, z)$, is to update the recognition network so that the Kullback-Leibler divergence between the true and the approximate posterior is minimised:

$$\underset{\phi}{\operatorname{argmin}}\ \mathrm{D}_{\mathrm{KL}}[p_\theta(z|x)||q_\phi(z|x)]. \tag{2.6}$$

Since the DDC $q_\phi(z|x)$ is in the exponential family, the KL-divergence in Eq. (2.6) is minimised if the expectations of the sufficient statistics vector $\boldsymbol{\psi} = [\psi_1, \ldots, \psi_K]$ under the two distributions agree: $\left\langle \boldsymbol{\psi}(z) \right\rangle_{p_\theta(z|x)} = \left\langle \boldsymbol{\psi}(z) \right\rangle_{q_\phi(z|x)}$. Hence the parameters of the recognition model should be updated so that: $r_l(x, \phi_l) \approx \left\langle \boldsymbol{\psi}(z_l) \right\rangle_{p_\theta(z_l|x)}$. This requirement can be translated into an optimisation problem by sampling $z_L^{(s)}, \ldots, z_1^{(s)}, x^{(s)}$ from the generative model and minimising the error between the output of the recognition model $r_l(x^{(s)}, \phi_l)$ and encoding functions $\boldsymbol{\psi}$ evaluated at the generated *sleep* samples. For tractability, we substitute the squared loss in place of Eq. (2.6):

$$\phi_l = \underset{\phi_l}{\operatorname{argmin}} \sum_s \left\| r_l(x^{(s)}, \phi_l) - \boldsymbol{\psi}(z_l^{(s)}) \right\|^2. \tag{2.7}$$

In principle, one could use any function approximator (such as a neural network) for the recognition model $r_l(x^{(s)}, \phi_l)$, provided that it is sufficiently flexible to capture the mapping from the data to the encoded expectations. Here, we parallel the original HM, and use a recognition model that reflects the hierarchical structure of the generative model. For a model with 2 layers of

latent variables:

$$h_1(x, W) = [Wx]_+ \tag{2.8}$$

$$r_1(x, \phi_1) = \phi_1 \cdot h_1(W, x) \tag{2.9}$$

$$r_2(x, \phi_2) = \phi_2 \cdot r_1(x, \phi_1) \tag{2.10}$$

where $W \in \mathbb{R}^{M \times D_0}, \phi_1 \in \mathbb{R}^{K_1 \times M}, \phi_2 \in \mathbb{R}^{K_2 \times K_1}$ and $[.]_+$ is a rectifying non-linearity. Throughout this chapter we use a fixed W sampled from a normal distribution, and update ϕ_1, ϕ_2 according to Eq. (2.7).

Recognition model learning in the DDC-HM thus parallels that of the original HM, albeit with a much richer posterior representation. The second aim of the DDC-HM sleep phase is quite different: a further set of weights must be learnt to approximate the gradients of the generative model joint likelihood. This step is derived in the appendix, but summarised in the following section.

2.2.2 Wake phase

The aim in the wake phase is to update the generative parameters to increase the variational free energy $\mathcal{F}(q, \theta)$, evaluated on data x, using a gradient step:

$$\Delta\theta \propto \nabla_\theta \mathcal{F}(q, \theta) = \langle \nabla_\theta \log p_\theta(x, z) \rangle_{q(z|x)} \tag{2.11}$$

The update depends on the evaluation of an expectation over $q(z|x)$. As discussed in Section 2.1, the DDC approximate posterior representation allows us to evaluate such expectations by approximating the relevant functions using the non-linear encoding functions $\boldsymbol{\psi}$.

For deep exponential family generative models, the gradients of the free energy take the following form (see appendix A.1):

$$\begin{aligned}
\nabla_{\theta_0}\mathcal{F} &= \nabla_{\theta_0}\langle \log p(x|z_1)\rangle_q = S_0(x)^T\langle \nabla g(z_1,\theta_0)\rangle_{q(z_1|x)} - \langle \mu_{x|z_1}^T \nabla g(z_1,\theta_0)\rangle_{q(z_1|x)} \\
&\vdots \\
\nabla_{\theta_l}\mathcal{F} &= \nabla_{\theta_l}\langle \log p(z_l|z_{l+1})\rangle_q = \langle S_l(z_l)^T \nabla g(z_{l+1},\theta_l)\rangle_{q(z_l,z_{l+1}|x)} - \langle \mu_{z_l|z_{l+1}}^T \nabla g(z_{l+1},\theta_l)\rangle_{q(z_{l+1}|x)} \\
&\vdots \\
\nabla_{\theta_L}\mathcal{F} &= \nabla_{\theta_L}\langle \log p(z_L)\rangle_q = \langle S_L(z_L)\rangle_{q(z_L|x)} - \nabla\Phi(\theta_L)\,,
\end{aligned} \tag{2.12}$$

where $\mu_{x|z_1}, \mu_{z_l|z_{l+1}}$ are expected sufficient statistic vectors of the conditional distributions from the generative model: $\mu_{x|z_1} = \langle S_0(x)\rangle_{p(x|z_1)}$, $\mu_{z_l|z_{l+1}} = \langle S_l(z_l)\rangle_{p(z_l|z_{l+1})}$. Now the functions that must be approximated are the functions of $\{z_l\}$ that appear within the expectations in Eqs. 2.12. As shown in appendix A.1, the coefficients of these combinations can be learnt by minimising a squared error on the sleep-phase samples, in parallel with the learning of the recognition model.

Thus, taking the gradient in the first line of Eq. (2.12) as an example, we write $\nabla_\theta g(z_1,\theta_0) \approx \sum_i \alpha_0^{(i)}\psi^{(i)}(z_1) = \alpha_0 \cdot \boldsymbol{\psi}(z_1)$ and evaluate the gradients as follows:

$$\text{sleep:} \qquad \alpha_0 \leftarrow \operatorname{argmin} \sum_s \left(\nabla_\theta g(z_1^{(s)},\theta_0) - \alpha_0 \cdot \boldsymbol{\psi}(z_1^{(s)})\right)^2 \tag{2.13}$$

$$\text{wake:} \qquad \langle \nabla_\theta g(z_1,\theta_0)\rangle_{q(z_1|x)} \approx \alpha_0 \cdot \langle \boldsymbol{\psi}(z_1)\rangle_{q(z_1|x)} = \alpha_0 \cdot r_1(x,\phi_1) \tag{2.14}$$

with similar expressions providing all the gradients necessary for learning derived in the appendix.

In summary, in the DDC-HM computing the wake-phase gradients of the free energy becomes straightforward, since the necessary expectations are computed using approximations learnt in the sleep phase, rather than by an explicit construction of the intractable posterior. Furthermore, as shown in the appendix, using the function approximations trained using the sleep samples and the posterior representation produced by the recognition network, we can learn

the generative model parameters without needing any explicit independence assumptions (within or across layers) about the posterior distribution.

2.3 Experiments

We first evaluated the performance of the DDC-HM on a directed graphical model comprising two stochastic latent layers and an observation layer. The prior on the top layer is a mixture of Gaussians, while the conditional distributions linking the layers below are Laplace and Gaussian, respectively:

$$\begin{aligned} p(z_2) &= 1/2\left(\mathcal{N}(z_2|m,\sigma^2) + \mathcal{N}(z_2|{-}m,\sigma^2)\right) \\ p(z_1|z_2) &= \text{Laplace}(z_1|\mu = 0, \lambda = \text{softplus}(Bz_2)) \\ p(x|z_1) &= \mathcal{N}(x|\mu = \Lambda z_1, \Sigma_x = \Psi_{diag}) \end{aligned} \tag{2.15}$$

We chose a generative model with a non-Gaussian prior distribution and sparse latent variables, models typically not considered in the VAE literature. Due to the sparsity and non-Gaussianity, learning in these models is challenging, and the use of a flexible posterior approximation is crucial. We show that the DDC-HM can provide a sufficiently rich posterior representation to learn accurately in such a model. We begin with low dimensional synthetic data to evaluate the performance of the approach, before evaluating performance on a data set of natural image patches (van Hateren and van der Schaaf, 1998).

2.3.1 Synthetic examples

To illustrate that the recognition network of the DDC-HM is powerful enough to capture dependencies implied by the generative model, we trained it on a data set generated from the model ($N = 10000$). The dimensionality of the observation layer, the first and second latent layers was set to $D_0 = 2, D_1 = 2, D_2 = 1$, respectively, for both the true generative model and the fitted models. We used a recognition model with a hidden layer of size 100, and $K_1 = K_2 = 100$ encoding functions for each latent layer, with 200 sleep samples, and learned the parameters of the conditional distributions $p(x|z_1)$ and $p(z_1|z_2)$ while keeping

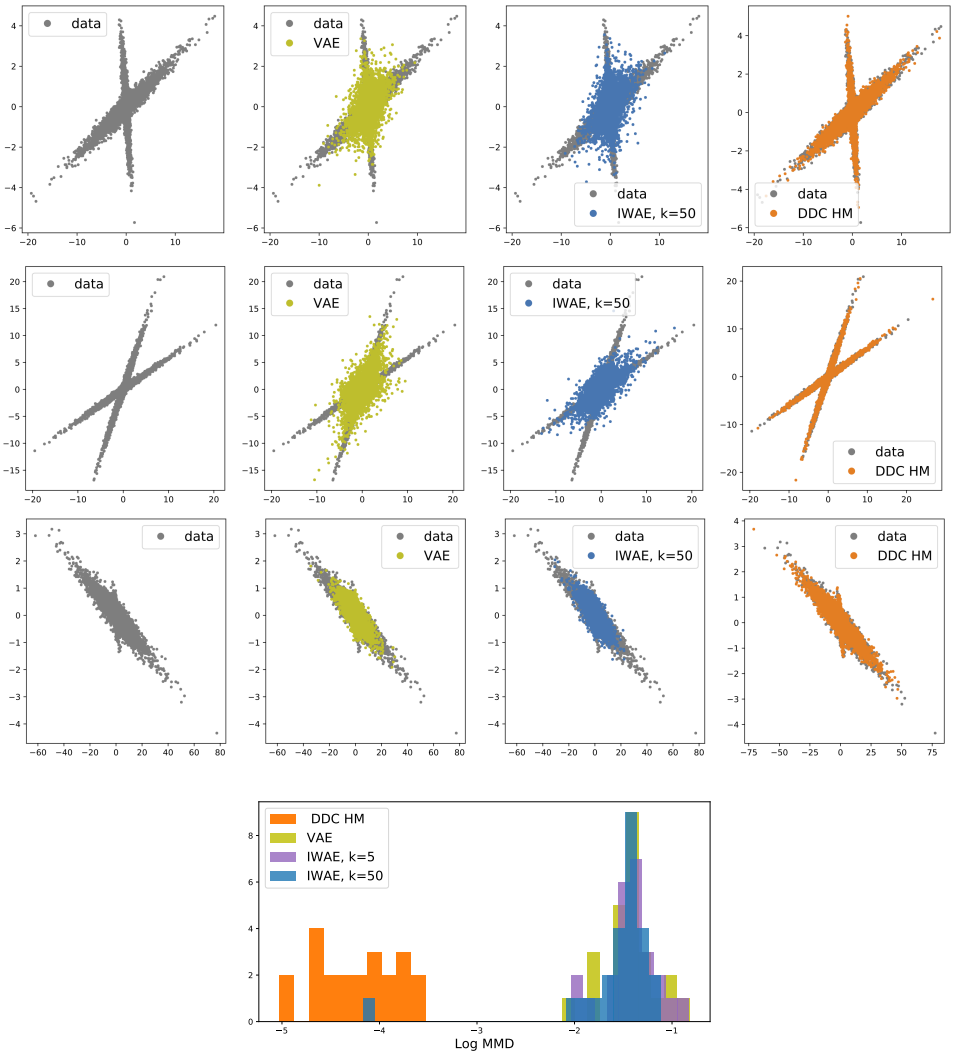


Figure 2.1: Top: Examples of the distributions learned by the Variational Autoencoder (VAE), the Importance Weighted Variational Autoencoder (IWAE) with $k = 50$ importance samples and the DDC Helmholtz Machine. Bottom: histogram of log MMD values for different algorithms trained on synthetic datasets.

the prior on z_2 fixed ($m = 3, \sigma = 0.1$).

As a comparison, we have also fitted both a Variational Autoencoder (VAE) and an Importance Weighted Autoencoder (IWAE), using 2-layer recognition networks with 100 hidden units each, producing a factorised Gaussian posterior approximation (or proposal distribution for the IWAE). To make the comparison between the algorithms clear (i.e. independent of initial conditions, local optima of the objective functions) we initialised each model to the true generative parameters and ran the algorithms until convergence (1000 epochs, learning rate: 10^{-4}, using the Adam optimiser; (Kingma and Ba, 2014)).

Figure 2.1 shows examples of the training data and data generated by the VAE, IWAE and DDC-HM models after learning. The solution found by the DDC-HM matches the training data, suggesting that the posterior approximation was sufficiently accurate to avoid bias during learning. The VAE, as expected from its more restrictive posterior approximation, could capture neither the strong anti-correlation between latent variables nor the heavy tails of the distribution. Similar qualitative features are seen in the IWAE samples, suggesting that the importance weighting was unable to recover from the strongly biased posterior proposal.

We quantified the quality of the fits by computing the *maximum mean discrepancy* (MMD) (Gretton et al., 2012) between the training data and the samples generated by each model (Bounliphone et al., 2015) [2]. We used an exponentiated quadratic kernel with kernel width optimised for maximum test power (Jitkrittum et al., 2016). We computed the MMD for 25 data sets drawn using different generative parameters, and found that the MMD estimates were significantly lower for the DDC-HM than for the VAE or the IWAE ($k = 5, 50$) (Figure 2.1).

Beyond capturing the density of the data, correctly identifying the underlying latent structure is also an important criterion when evaluating algorithms for learning generative models. Figure 2.2 shows an example where we have

[2]Estimating the log-likelihood by importance sampling in this model has proven to be unreliable due the lack of a good proposal distribution

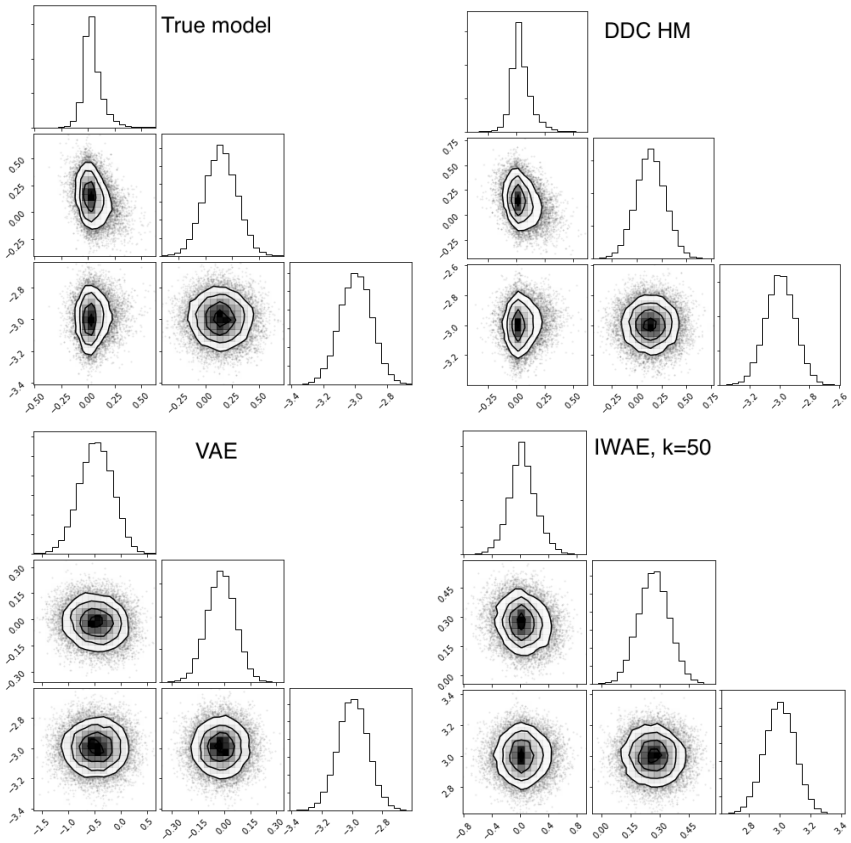


Figure 2.2: Example posterior distributions corresponding to the learned generative models. The corner plots show the pairwise and marginal densities of the three latent variables, for the true model (top left), the model learned by the VAE, IWAE ($k = 50$) and DDC-HM.

used Hamiltonian Monte Carlo to generate samples from the true posterior distribution for one data point under the generative models learnt by each approach. We found that there was close agreement between the posterior distributions of the true generative model and the one learned by the DDC-HM. However, the biased recognition of the VAE and IWAE in turn biases the learnt generative parameters so that the resulting posteriors (even when computed without the recognition networks) appear closer to Gaussian.

2.3.2 Natural image patches

We tested the scalability of the DDC-HM by applying it to a natural image data set (van Hateren and van der Schaaf, 1998). We trained the same generative model as in section 2.3.1 on image patches with dimensionality $D_0 = 16 \times 16$ and varying sizes of latent layers. The recognition model had a hidden layer of

size 500, $K_1 = 500$, $K_2 = 100$ encoding functions for z_1 and z_2, respectively, and used 1000 samples during the sleep phase. We compared the performance of our model with the IWAE (k=50) using the relative (three sample) MMD test (Bounliphone et al., 2015) with a exponentiated quadratic kernel (width chosen by the median heuristic). The test establishes whether the MMD distance between distributions P_x and P_y is significantly smaller than the distance between P_x and P_z. We used the image data set as our reference distribution and the IWAE being closer to the data as null hypothesis. Table 2.1 summarises the results obtained on models with different latent dimensionality, all of them strongly preferring the DDC-HM.

Table 2.1: 3-sample MMD results. The table shows the results of the 'relative' MMD test between the DDC-HM and the IWAE ($k = 50$) on the image patch data set for different generative model architectures. The null hypothesis tested: $\mathrm{MMD}_{\mathrm{IWAE}} < \mathrm{MMD}_{\mathrm{DDC-HM}}$. Small p values indicate that the model learned by the DDC HM matches the data significantly better than the one learned by the IWAE ($k = 50$). We obtained similar results when comparing to IWAE $k = 1, 5$ (not shown).

Latent dimensions		IWAE	DDC HM	p-value
$D_1 = 10$	$D_2 = 10$	0.126	0.0388	$\ll 10^{-5}$
$D_1 = 50$	$D_2 = 2$	0.0754	0.0269	$\ll 10^{-5}$
$D_1 = 50$	$D_2 = 10$	0.247	0.00313	$\ll 10^{-5}$
$D_1 = 100$	$D_2 = 2$	0.076	0.0211	$\ll 10^{-5}$
$D_1 = 100$	$D_2 = 10$	0.171	0.00355	$\ll 10^{-5}$

2.3.3 A generative model of olfactory stimuli

To further demonstrate that the DDC Helmholtz machine can exploit its rich posterior representation to accurately learn generative models of sensory relevance, we used the algorithm to learn a model of olfactory stimuli (see fig. 2.3). Latent variables in the generative model corresponded to concentration of odours (e.g. scent of a given fruit or flower) and odorants (different compounds contributing to an odour). Observations in this model can be thought of as the activity of olfactory receptor neurons (ORN), cells that relay chemical signals from the outside world to the brain. We chose to model odour and odorant

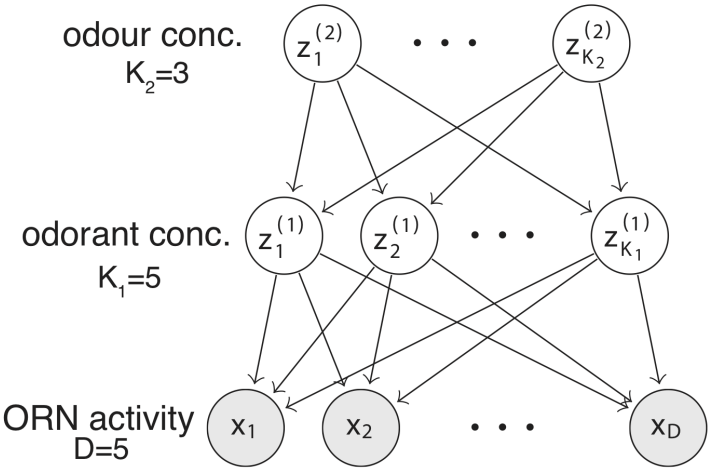


Figure 2.3: Generative model with 2 layers of latent variables corresponding to a set of odour and odorant concentrations and olfactory receptor neuron (ORN) activity as observations.

concentrations with Gamma distributions and the conditional distribution of the ORN activity (relative to some baseline) to be Gaussian:

$$p(\mathbf{z}_2) = \text{Gamma}(\mathbf{z}_2|\alpha_2, \beta_2) \tag{2.16}$$

$$p(\mathbf{z}_1|\mathbf{z}_2) = \text{Gamma}(\mathbf{z}_1|\alpha_1, \beta_1 = \frac{\alpha_1}{\text{softplus}(\theta_2 \mathbf{z_2})}) \tag{2.17}$$

$$p(\mathbf{x}|\mathbf{z}_1) = \mathcal{N}(\mathbf{x}|\mu = \theta_1 \mathbf{z_1}, \Sigma = \sigma^2 I) \tag{2.18}$$

We used a recognition model with hidden layer size of 80, with 40 DDC encoding functions for each layer of the hidden variables. For training, we used $N = 10^4$ observations generated from the true model and 10^3 samples during each sleep phase of the algorithm. We learned recognition network and generative parameters θ_1, θ_2 (through $3 * 10^3$ epochs, with learning rate 10^{-3}), while the remaining parameters were fixed at their true values. As a comparison, we also trained a VAE assuming the same generative model class and a recognition model producing factorised Gamma distributions as approximate posterior.

The DDC-HM learned the statistics of receptor activations accurately (fig. 2.4, top), while variational methods relying on independence assumptions – as used for olfactory inference by Grabska-Barwińska et al. (2017) – fall short

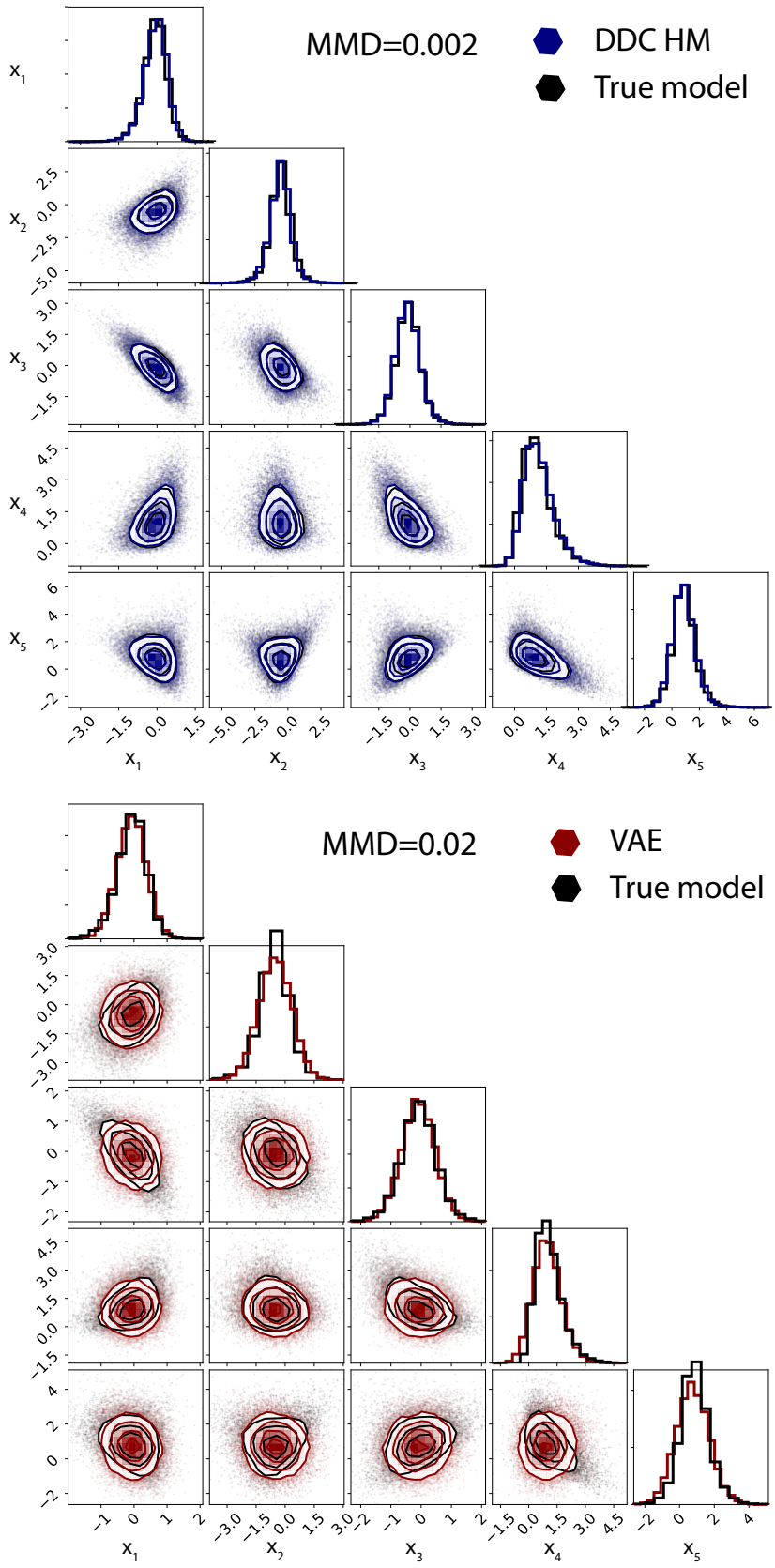


Figure 2.4: Models learned by the DDC Helmholtz machine (top) and a VAE (bottom; assuming factorised gamma posterior). Synthetic training data plotted in black, observations generated from the learned models in colour.

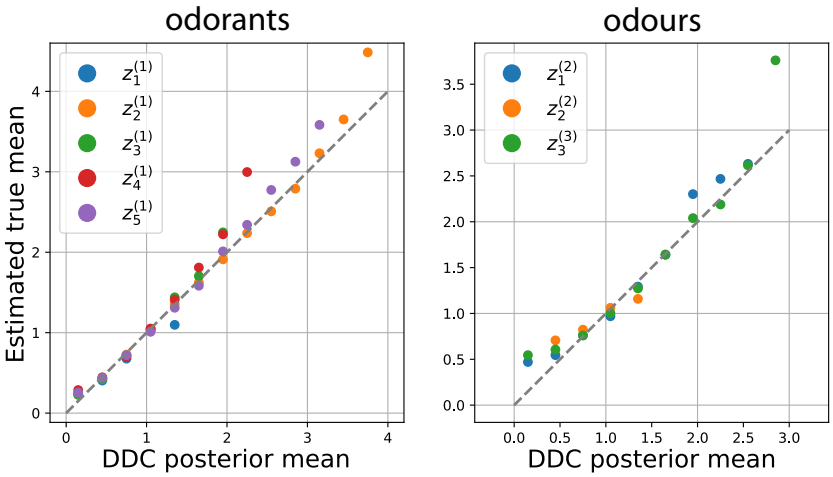


Figure 2.5: Evaluating the accuracy of the DDC-HM recognition model's prediction of the posterior mean for latent variables corresponding to odorants and odours. Vertical axis shows means estimated directly from samples.

of capturing the distribution in detail (fig. 2.4, bottom). We quantified the quality of fits by computing the MMD between data from the true generative model and data generated from the learned models (see fig. 2.4 legends).

We also found that the recognition model accurately predicted the posterior means of odorant and odour concentrations for new observations (fig. 2.5).

2.3.4 Sigmoid Belief Network trained on MNIST

Finally, we evaluated the capacity of our model to learn hierarchical generative models with discrete latent variables by training a sigmoid belief network (SBN). We used the binarised MNIST dataset of 28x28 images of handwritten digits (Salakhutdinov and Murray, 2008). The generative model had three layers of binary latent variables, with dimensionality of 200 in each layer. The recognition model had a sigmoidal hidden layer of size 300 and DDC representations of size 200 for each latent layer. As a comparison, we have also trained an SBN with the same architecture using the VIMCO algorithm (as described in Mnih and Rezende (2016)) with 50 samples from the proposal distribution [3]. To quantify the fits, we have performed the relative MMD test using the test set ($N = 10000$) as a reference distribution and two sets of samples of the same

[3]The model achieved an estimated negative log-likelihood of 90.97 nats, similar to the one reported by (Mnih and Rezende, 2016) (90.9 nats)

size generated from the SBN trained by the VIMCO and DDC-HM algorithms. Again, we used an exponentiated quadratic kernel with width chosen by the median heuristic. The test strongly favoured the DDC-HM over VIMCO with $p \ll 10^{-5}$ (with MMD values of 6×10^{-4} and 2×10^{-3}, respectively).

2.4 Related work

Following the Variational Autoencoder (VAE; Rezende et al. (2014); Kingma et al. (2014)), there has been a renewed interest in using recognition models – originally introduced in the HM – in the context of learning complex generative models. The Importance Weighted Autoencoder (IWAE; (Burda et al., 2015)) maximises a tighter lower bound constructed by an importance sampled estimator of the log-likelihood using the recognition model as a proposal distribution. This approach decreases the variational bias introduced by the factorised posterior approximation of the standard VAE. VIMCO (Mnih and Rezende, 2016) extends this approach to discrete latent variables and yields state-of-the-art generative performance on learning sigmoid belief networks. We compare our method to the IWAE and VIMCO in section 2.3. Sonderby et al. (Sønderby et al., 2016) demonstrate that the standard VAE has difficulty making use of multiple stochastic layers. To overcome this, they propose the Ladder Variational Autoencoder with a modified parametrisation of the recognition model that includes stochastic top-down pass through the generative model. The resulting posterior approximation is a factorised Gaussian as for the VAE. Normalising Flows (Rezende and Mohamed, 2015) relax the factorised Gaussian assumption on the variational posterior. Through a series of invertible transformations, an arbitrarily complex posterior can be constructed. However, to our knowledge, they have not yet been successfully applied to deep hierarchical generative models.

2.5 Discussion

The DDC Helmholtz Machine offers a novel approach to learning hierarchical generative models, which combines the basic idea of the wake-sleep algorithm

with a flexible posterior representation. The lack of strong parametric assumptions in the DDC representation allows the algorithm to learn generative models with complex posterior distributions accurately.

As in the classical Helmholtz Machine, the approximate posterior is found by seeking to minimise the "reverse" divergence $D_{\mathrm{KL}}[p(z|x)\|q(z|x)]$, albeit within a much richer class of distributions. Thus, the modified wake-sleep algorithm presented here still does not directly optimise a variational lower bound on the log-likelihood. Rather, it can be viewed as following an approximation to the *gradient* of the log-likelihood, where the quality of the approximation depends on the richness of the DDC representation used. We expect that when the approximation is rich enough for the error in the resulting gradient estimate to be bounded, the algorithm will always reach a region around a local mode in which the true gradient does not exceed that error bound. We explore precise conditions for convergence the in next chapter.

The DDC-HM recognition model can be trained layer-by-layer using the samples from the generative model, with no need to back-propagate gradients across stochastic layers. In the version discussed here, the recognition network depended on linear mappings between encoding functions and a fixed non-linear basis expansion of the input. This restrictive form allowed for closed-form updates in the sleep phase. However, this assumption could be relaxed by introducing a neural network between each latent variable layer, along with a modified learning scheme in the sleep phase. This approach may increase the accuracy of the posterior expectations computed during the wake phase.

Another future direction involves learning the non-linear encoding functions or choosing them in accordance with the properties of the generative model (e.g. requiring sparsity in the random projections). Finally, a natural extension of the DDC representation with expectations of a finite number of encoding functions, is to approach mean embedding in a reproducing kernel Hilbert space, corresponding to infinitely many encoding functions (Smola et al., 2007; Grünewälder et al., 2012), a direction we discuss in the following chapter.

Even without these extensions, however, the DDC-HM offers a novel and powerful approach to probabilistic learning with complex hierarchical models.

Chapter 3

The Kernel Helmholtz Machine

In the previous chapter, we considered the problem of parameter estimation in hierarchical latent variable models, where each layer follows an exponential family distribution conditioned on the one above. While the DDC Helmholtz machine showed notable empirical results for simple problems, it left unaddressed formal questions of convergence and accuracy of learning.

In this chapter, we work towards filling this gap. We first make an explicit connection between the DDC posterior representation introduced in section 2.1 and mean embeddings in reproducing kernel Hilbert spaces (Smola et al., 2007), leading to a "Kernel Helmholtz machine" which provides a more general framework, albeit with the loss of the neural motivation. In the Kernel Helmholtz machine, inference and log-likelihood gradients for learning are approximated by functions within a reproducing kernel Hilbert space.

We then exploit the properties of reproducing kernel Hilbert spaces to derive a bound on the accuracy of the approximate gradients used for learning, and so demonstrate approximate convergence of the algorithm to a region around a local mode of the marginal log-likelihood.

While the feature expectations in the DDC-HM can be seen as representing a maximum entropy distribution of the form $q(z_l|x) \propto \exp\left(\sum_i \eta_i(x)\psi_i(z_l)\right)$, with the natural parameters η depending on the recognition network outputs (the mean parameters), the posterior density itself did not need to be evaluated. Instead, the sleep samples were used to learn a linear expansion of the gradients

of the generative model joint log-likelihood in terms of the functions $\psi_i(z_l)$. In the wake phase, the same linear combination is applied to the *expectations* of $\psi_i(z_l)$ provided by the recognition network. The resulting estimates of the expected gradients could then be used to update the generative model parameters. In a sense then, the DDC-HM relies on the amortisation of *learning* by a network architecture rather than the amortisation of *inference* alone—unlike standard variational approaches.

While this richer form of posterior representation might be expected to yield results closer to the true variational optimum of EM, it is still trained to minimise the *reverse* KL divergence $D_{KL}[p(z|x)\|q(z|x)]$ and so it is not guaranteed to optimise the variational free energy $\mathcal{F}(q, \theta)$. Nonetheless, as we suggested in section 2.5, the critical issue was the accuracy of the gradient approximation, and if this was small, wake-sleep learning would approach a region close to a true optimum of the likelihood. Here we use the connection between the DDC and mean embeddings in a reproducing kernel Hilbert space to explore this claim further.

3.1 Kernel Helmholtz Machine

We can now introduce the Kernel Helmholtz machine (Kernel-HM), a new algorithm that can be viewed as a generalisation of the DDC-HM. The formulation of the Kernel-HM allows us to characterise the type of approximations made by the HM explicitly, and show that under certain conditions we can guarantee approximate convergence to the region of a stationary point of the true *log-likelihood* $\log p(x; \theta)$.

The Kernel-HM, like the DDC-HM, is an algorithm for learning in general deep exponential family models (section 1.2). As the likelihood is intractable for this class of models, it cannot be optimised directly with respect to the generative model parameters. A common approach used by the family of variational methods is to replace the log-likelihood by a lower bound that can be evaluated and its gradients computed using Monte Carlo approxima-

tion. Instead, the Kernel-HM constructs an approximation to the *gradients* of the marginal log-likelihood directly. Given a generative model with joint density $p(x, z_{1:L}; \theta)$ and (exact) posterior distribution $p(z_{1:L}|x)$ one can write the gradients of $\log p(x; \theta)$ in the following form:

$$\nabla_\theta \log p(x;\theta) = \mathbb{E}_{p(z|x)}[\nabla_\theta \log p(x, z_{1:L};\theta)]. \tag{3.1}$$

The above expression can be derived in a few steps:

$$\begin{aligned}
\nabla_\theta \log p(x;\theta) &= \nabla_\theta \log \int p(x, z_{1:L};\theta) dz_{1:L} && (3.2)\\
&= \frac{1}{p(x)} \nabla_\theta \int p(x, z_{1:L};\theta) dz_{1:L} && (3.3)\\
&= \frac{1}{p(x)} \int p(x, z_{1:L};\theta) \nabla_\theta \log p(x, z_{1:L};\theta) dz_{1:L} && (3.4)\\
&= \mathbb{E}_{p(z|x)}[\nabla_\theta \log p(x, z_{1:L};\theta)] && (3.5)
\end{aligned}$$

When the latent variables $z_{1:L}$ are arranged in a hierarchy, as in the case of deep exponential family models, the gradients can be split up according to the factorisation of the joint density:

$$\begin{aligned}
\nabla_{\theta_0} \log p(x;\theta) &= \mathbb{E}_{p(z_1|x)}[\nabla_{\theta_0} \log p(x|z_1;\theta_0)] && (3.6)\\
\nabla_{\theta_l} \log p(x;\theta) &= \mathbb{E}_{p(z_l, z_{l+1}|x)}[\nabla_{\theta_l} \log p(z_l|z_{l+1};\theta_l)]
\end{aligned}$$

Notice that the gradients of $\log p(x, z_{1:L})$ (eq. 3.1) can be evaluated analytically, since each of the generative conditional distributions is from a tractable family. Thus, estimating the gradients of the marginal log-likelihood $\log p(x;\theta)$ can be translated into a problem of *estimating conditional expectations* with respect to the posterior distribution $p(z|x)$ of functions of the latent variables $z_{1:L}$.

Before we discuss the algorithm in more detail, we expand on the functional form of the gradients for the case of deep exponential family models, using the notation introduced in section 1.2.

$$\mathbb{E}_{p(z_1|x)}[\nabla_{\theta_0} \log p(x|z_1;\theta_0)] = \tag{3.7}$$
$$= \mathbb{E}_{p(z_1|x)}[S_0(x)^T \nabla_{\theta_0} g_0(z_1,\theta_0) - \mathbb{E}_{p(x|z_1)}[S_0(x)]^T \nabla_{\theta_0} g_0(z_1,\theta_0)]$$
$$= S_0(x)^T \, \mathbb{E}_{p(z_1|x)}[\nabla_{\theta_0} g_0(z_1,\theta_0)] - \mathbb{E}_{p(z_1|x)}[m_0(z_1)^T \nabla_{\theta_0} g_0(z_1,\theta_0)] \tag{3.8}$$

Where $m_0(z_1) = \mathbb{E}_{x|z_1}[S_0(x)]$, the mean parameters of $p_\theta(x|z_1)$ as a function of the latent variable z_1. Similarly, for higher layers:

$$\begin{aligned}\mathbb{E}_{z_l,z_{l+1}|x}\left[\nabla_{\theta_l} \log p(z_l|z_{l+1};\theta_l)\right] =& \mathbb{E}_{z_l,z_{l+1}|x}\left[S(z_l)^T \nabla_{\theta_l} g(z_{l+1},\theta_l)\right. \qquad (3.9)\\ &\left. - \mathbb{E}_{z_l|z_{l+1}}[S(z_l)]^T \nabla_{\theta_l} g(z_{l+1},\theta_l)\right] \\ =& \mathbb{E}_{z_l|x}\left[S(z_l)^T \, \mathbb{E}_{z_{l+1}|z_l}[\nabla_{\theta_l} g(z_{l+1},\theta_l)]\right] \qquad (3.10)\\ &- \mathbb{E}_{z_{l+1}}\left[m_l(z_{l+1})^T \nabla_{\theta_l} g(z_{l+1},\theta_l)\right]\end{aligned}$$

with $m_l(z_{l+1}) = \mathbb{E}_{z_l|z_{l+1}}[S(z_l)]$.

As the expectations above will appear repeatedly, we define functions $f_{l,i}(z_l) = S(z_l)^T \, \mathbb{E}_{z_{l+1}|z_l}[\nabla_{\theta_{l,i}} g(z_{l+1},\theta_l)]$ and $h_{l+1,i}(z_{l+1}) = m_l(z_{l+1})^T \nabla_{\theta_{l,i}} g(z_{l+1},\theta_l)$ equal to their arguments, so that we can write the i^{th} element of the gradient vector for layers $l = 0 \ldots L-1$ as:

$$\nabla_{0,i} = S_i(x) \, \mathbb{E}_{z_1|x}[f_{0,i}(z_1)] + \mathbb{E}_{z_1|x}[h_{1,i}(z_1)] \tag{3.11}$$
$$\nabla_{l,i} = \mathbb{E}_{z_l|x}[f_{l,i}(z_l)] + \mathbb{E}_{z_{l+1}|x}[h_{l+1,i}(z_{l+1})] \tag{3.12}$$

where the scalar valued function $S_i(x) \in \mathbb{R}$ refers to the element in the sufficient statistic vector $S(x)$ for which the corresponding natural parameter depends on the i^{th} element of θ_0 (i.e. has non-zero derivative wrt. $\theta_{0,i}$).

Now the gradient estimation can be performed by estimating the conditional expectations of the functions $f_{l,i}$ and $h_{l,i}$ with respect to marginal posterior distributions $p(z_l|x)$.

Let $\mathcal{H}_\mathcal{Z}$ be an RKHS with kernel $k_\mathcal{Z}(z,z')$ and assume that $f_{l,i}, h_{l,i} \in \mathcal{H}_\mathcal{Z} \forall l,i$. Under these assumptions, the conditional expectations in Eq 3.11,

3.12 can be represented as an inner product in $\mathcal{H}_\mathcal{Z}$, eg.:

$$\mathbb{E}_{z_l|x}[h(z_l)] = \langle h, \mu_l(x) \rangle_{\mathcal{H}_\mathcal{Z}} \tag{3.13}$$

where $\mu_l(x) \in \mathcal{H}_\mathcal{Z}$ is conditional mean embedding of the posterior $p(z_l|x)$.

The DDC representation introduced in section 2.1 can be seen as a mean embedding in a finite-dimensional RKHS feature vector with a kernel given by $k_\mathcal{Z}(z_l, z_l') = \sum_i \psi_i(z_l)\psi_i(z_l')$, and the vector of expected sufficient statistics corresponding to the mean embedding: $\mu_l = \mathbb{E}_{p(z_l|x)}[\boldsymbol{\psi}(z_l)]$.

We can now write the gradients in each layer of the generative model as follows:

$$\nabla_{0,i} = S_i(x)\langle f_{0,i}, \mu_1(x) \rangle_{\mathcal{H}_\mathcal{Z}} + \langle h_{0,i}, \mu_1(x) \rangle_{\mathcal{H}_\mathcal{Z}} \tag{3.14}$$

$$\nabla_{l,i} = \langle f_{l,i}, \mu_l(x) \rangle_{\mathcal{H}_\mathcal{Z}} + \langle h_{l,i}, \mu_{l+1}(x) \rangle_{\mathcal{H}_\mathcal{Z}} \tag{3.15}$$

In practice, while the functions $h_{l,i}$ are available in closed form, $f_{l,i}$ and the mean embeddings μ_l need to be estimated from samples, yielding an approximation ($\hat{\nabla}$) to the true gradient of $\log p(x;\theta)$. However, as we show in section 3.2, the errors from the finite sample estimators can be used to consistently bound the bias and variance of the approximate gradient.

Each iteration of the algorithm for the Kernel-HM, similarly to the HM and DDC-HM, can be broken up into two phases.

3.1.1 Sleep phase

In the sleep phase a set of samples $\{x^s, z_1^s, \dots z_L^s\}_{s=1}^S$ is produced from the current generative model. Note that this can be done efficiently through ancestral sampling. These samples are then used to estimate the functions $f_{l,i} \in \mathcal{H}_\mathcal{Z}$ and the conditional mean embedding $\mu_l(x)$.

The functions $f_{l,i}$ can be estimated by kernel ridge regression (Murphy (2012), ch. 14) using the sleep samples as training data, analogously to the

sleep phase regression in the DDC Helmholtz machine (eq. 2.13).

For estimating the conditional mean embedding μ, we follow Grünewälder et al. (2012) and consider the operator $\mu : \mathcal{X} \to \mathcal{H}_\mathcal{Z}$, an element of a vector valued RKHS $\mathcal{H}_\Gamma$ with operator valued kernel $\Gamma(x, x') = k_\mathcal{X}(x, x) I_{\mathcal{H}_\mathcal{Z}}$ with properties discussed in section 1.4.1.

The empirical estimate of μ_l minimises the following regularised objective:

$$\mathcal{E}^s_\lambda(\mu_l) = \sum_{s=1}^{S} \|k_\mathcal{Z}(z^s, .) - \mu(x^s)\|^2_{\mathcal{H}_\mathcal{Z}} + \lambda \|\mu_l\|^2_{\mathcal{H}_\Gamma} \tag{3.16}$$

with the resulting estimate: $\mu^s_\lambda = \underset{\mu}{\operatorname{argmin}}\, \mathcal{E}^s_\lambda(\mu)$. μ^s_λ can be found in closed form (Song et al., 2009):

$$\mu^s_\lambda(x) = \sum_{s=1}^{S} \alpha(x^s) k_\mathcal{Z}(z^s_l, .) \quad \alpha(x^s) = \sum_{t=1}^{S} W_{st} k_\mathcal{X}(x^t, x) \tag{3.17}$$

where $\mathbf{W} = (\mathbf{K} + \lambda S \mathbf{I})^{-1}$ and $\mathbf{K} = \left(k_\mathcal{X}(x^s, x^t)\right)^S_{s,t=1}$, with regularisation parameter λ.

Song et al. (2010) has shown that the estimator in eq. 3.17 is consistent with rate $O(S^{-1/4})$ under strong smoothness assumptions on the true conditional embedding and assuming that $\mu \in \mathcal{H}_\Gamma$. Recently, Singh et al. (2019) has provided guarantees of consistency under weaker smoothness assumptions (but still requiring that $\mu \in \mathcal{H}_\Gamma$) with similar convergence rates. In the case of a finite dimensional RKHS $\mathcal{H}_\mathcal{Z}$, Grünewälder et al. (2012) showed that estimating μ can be translated into a vector valued regression problem with much faster convergence rates $O(\log S/S)$.

3.1.2 Wake phase

The wake phase simply evaluates the approximate gradients $\hat{\nabla}$ by computing the inner products between the estimated functions an mean embeddings according to Eq. 3.14, 3.15 and updates the generative model parameters θ by a stochastic gradient step.

The intuition is that as long as the approximate gradient produced on

average remains close to the true gradient and has finite variance the Kernel-HM should reach the vicinity of a stationary point of the log-likelihood. We formalise these intuitions in the next section and give explicit conditions for approximate convergence of the algorithm.

Algorithm 2 Kernel Helmholtz Machine training

Initialise θ
while not converged **do**
  Sleep phase:
  sample: $z_L^{(s)}, ..., z_1^{(s)}, x^{(s)} \sim p_\theta(x, z_1, ..., z_L)$
  update $\mu_l^s = \text{argmin}_{\mu_l \in \mathcal{H}_\Gamma} \sum_s \|k_\mathcal{Z}(z_l^{(s)}, .) - \mu_l(x^{(s)})\|^2_{\mathcal{H}_\mathcal{Z}} + \lambda \|\mu_l\|^2_{\mathcal{H}_\Gamma}$
  update functions $f_{l,i}^s \in \mathcal{H}_\mathcal{Z}$
  Wake phase:
  $x \leftarrow$ {minibatch}
  evaluate $\hat{\nabla}$ according to Eq. 3.14-3.15
  update θ: $\theta \leftarrow \theta + \alpha \hat{\nabla}$
end while

3.2 Approximate convergence of the Kernel Helmholtz machine

In the following, we establish conditions under which the wake-sleep algorithm for the Kernel Helmholtz machine converges approximately to the region of stationary point of the log-likelihood $\log p(x; \theta)$. We consider a standard stochastic gradient ascent algorithm with initial condition θ_{init} and updates:

$$\theta_{t+1} = \theta_t + \alpha_t \hat{\nabla}_t \,,$$

where $t = 1 \dots T-1$ and $\hat{\nabla}_t$ is the approximation to the true gradient $\nabla_\theta \log p(x; \theta_t)$, computed in the Kernel-HM wake phase.

As wake-sleep is a stochastic optimization scheme, approximate convergence can be established by considering the norm of the gradients after a sufficiently large number of iterations. Applying results from non-convex stochastic programming (Ghadimi and Lan, 2013; Sanjabi et al., 2018), and assuming appropriate smoothness properties for the gradient function, we can relate the

norm of the true gradients to the quality of the gradient approximation (i.e., its bias and variance).

Theorem 3.2.1. *Suppose that the gradient of* $\log p(x;\theta)$ *wrt. the generative parameters* θ *is Lipschitz with constant* M*. Let* $\Delta = \log p(x;\theta_{init}) - \sup_\theta \log p(x;\theta)$ *and* $G_t = \mathbb{E}[\hat{\nabla}_t|\theta_t]$*. Further assume that* $\|G_t - \nabla_\theta \log p(x;\theta_t)\| \leq \delta$ *and* $\mathbb{E}[\|\nabla_t - G_t\|^2] \leq \sigma^2$, $\forall t$

1. *If* $T < \frac{2\Delta M}{\sigma^2}$ *then* $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla_\theta \log p(x;\theta_t)\|^2] \leq \frac{2M\Delta}{T} + \delta^2 + \sigma^2$, *with learning rate* $\alpha_t = \frac{1}{M}$

2. *If* $T \geq \frac{2\Delta M}{\sigma^2}$ *then* $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla_\theta \log p(x;\theta_t)\|^2] \leq \sigma\sqrt{8\frac{M\Delta}{T}} + \delta^2$, *with learning rate* $\alpha_t = \sqrt{\frac{2\Delta}{M\sigma^2 T}}$

The proof follows Sanjabi et al. (2018), inspired by Ghadimi and Lan (2013). If the bias in the gradient estimate at each step t is δ_t then Theorem 3.2.1 still holds with $\delta^2 = \frac{1}{T}\sum_{t+1}^{T}\delta_t^2$. Thus, it is sufficient to keep the average bias of the estimate small enough throughout the optimisation.

Theorem 3.2.1 implies that as $T \to \infty$ the algorithm converges to a *shallow* region defined by the bias δ around a local optimum, while the speed of convergence depends on the variance of the estimator (σ^2) and the smoothness (M) of the gradient function.

3.2.1 Properties of the Kernel-HM gradient estimator

As shown by Theorem 3.2.1, the bias and the variance of the gradient estimator determine the convergence properties of the Kernel-HM. Below we derive relevant bounds on these quantities and show they depend on the approximations made in the algorithm.

Following the assumptions from section 3.1, i.e. $f_{l,i}, h_{l,i} \in \mathcal{H}_\mathcal{Z}$ and $\exists \mu \in \mathcal{H}_\Gamma, \mu : \mathcal{X} \to \mathcal{H}_\mathcal{Z}$ such that $E_{z|x}[h(z)] = \langle h, \mu(x)\rangle_{\mathcal{H}_\mathcal{Z}} \quad \forall x \in \mathcal{X}, h \in \mathcal{H}_\mathcal{Z}$, the true gradient can be expressed using inner products in $\mathcal{H}_\mathcal{Z}$:

$$\nabla_{0,i} = S_i(x)\langle f_{0,i}, \mu_1(x)\rangle_{\mathcal{H}_\mathcal{Z}} + \langle h_{0,i}, \mu_1(x)\rangle_{\mathcal{H}_\mathcal{Z}} \tag{3.18}$$

Similarly, the approximate gradient can be written as:

$$\hat{\nabla}_{0,i} = S_i(x)\langle f^s_{0,i}, \mu^s_1(x)\rangle_{\mathcal{H}_\mathcal{Z}} + \langle h_{0,i}, \mu^s_1(x)\rangle_{\mathcal{H}_\mathcal{Z}} \tag{3.19}$$

where $f^s \in \mathcal{H}_\mathcal{Z}$ and $\mu^s \in \mathcal{H}_\Gamma$. Importantly, we are making the assumption that the true functions approximated are a member of the approximating family used. This assumption, however, may not be very strict for rich enough RKHSs $\mathcal{H}_\Gamma, \mathcal{H}_\mathcal{Z}$.

For simplicity, we will first consider the gradients for the lowest layer and for most derivations we omit the indices l, i for the layer and component of the gradient. Our results straightforwardly carry through to all gradients of the generative model.

Bias of the gradient

The bias of the approximate gradient vector $\hat{\nabla}$ at each iteration t can be decomposed into the bias of the individual components.

$$\|\nabla \log p(x;\theta) - \mathbb{E}[\hat{\nabla}]\|^2 = \sum_i (\mathbb{E}[\hat{\nabla}_i] - \frac{d}{d\theta_i} \log p(x;\theta))^2 \tag{3.20}$$

For the Kernel-HM, the difference in the true and approximate gradients can be expressed using inner products in $\mathcal{H}_\mathcal{Z}$. In the case of gradients with respect to parameters of the likelihood $\log p(x|z_1;\theta_0)$, using the notation $\nabla_i := \frac{d}{d\theta_{0,i}} \log p(x;\theta) = \mathbb{E}_{z_1|x}\left[\frac{d}{d\theta_{0,i}} \log p(x|z_1;\theta_0)\right]$, we can write:

$$\begin{aligned}\left|\nabla_i - \mathbb{E}_s\, \hat{\nabla}_i\right| =& \big|S(x)\langle f, \mu(x)\rangle_{\mathcal{H}_\mathcal{Z}} + \langle h, \mu(x)\rangle_{\mathcal{H}_\mathcal{Z}} \\ & - \mathbb{E}_s[S(x)\langle f^s, \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}} + \langle h, \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}}]\big|\end{aligned} \tag{3.21}$$

where $\mathbb{E}_s$ denotes expectations with respect to the sleep samples generated from the model.

Next, we will show that we can construct an upper-bound on the bias of the gradient estimator that depends on the approximation error in μ and f.

By reducing the error in the functions estimated in the sleep phase, the bias can be made arbitrarily small. Which, in turn, leads to accurate convergence to the stationary solution of $\log p(x;\theta)$.

Theorem 3.2.2. *Assume that*

$\exists f \in \mathcal{H}_\mathcal{Z} : \frac{d}{d\theta_i} g(z_1,\theta) = \langle f, k_\mathcal{Z}(z_1,.)\rangle_{\mathcal{H}_\mathcal{Z}}$,

$\exists h \in \mathcal{H}_\mathcal{Z} : m_{x|z_1}(z_1)^T \frac{d}{d\theta_i} g(z_1,\theta) = \langle h, k_\mathcal{Z}(z_1,.)\rangle_{\mathcal{H}_\mathcal{Z}}$,

and $f^s, \mu^s \in \mathcal{H}_\mathcal{Z}$

Further assume that $\exists \mu \in \mathcal{H}_\Gamma, \mu : \mathcal{X} \to \mathcal{H}_\mathcal{Z}$ *such that* $E_{z|x}[h(z)] = \langle h, \mu(x)\rangle_{\mathcal{H}_\mathcal{Z}} \quad \forall x \in \mathcal{X}, h \in \mathcal{H}_\mathcal{Z}$

$\|\mu^s\|_{\mathcal{H}_\Gamma} \leq \lambda_\mu, \|\Gamma(x,x)\|_{op} \leq \kappa, \quad \lambda_\mu, \kappa \in \mathbb{R}^+$

Then:

$$\begin{aligned}\left|\nabla_i - \mathbb{E}_s[\hat{\nabla}_i]\right| \leq & \kappa\, \mathbb{E}_s\left[\|\mu - \mu^s\|_{\mathcal{H}_\Gamma}\right]\left(|S(x)|\|f\|_{\mathcal{H}_\mathcal{Z}} + \|h\|_{\mathcal{H}_\mathcal{Z}}\right) \\ & + \kappa\lambda_\mu |S(x)|\, \mathbb{E}_s[\|f - f^s\|_{\mathcal{H}_\mathcal{Z}}]\end{aligned} \tag{3.22}$$

Proof. See appendix A.2. □

The bound on the bias in theorem 3.2.2 depends on the errors resulting from the finite sample approximations in the sleep phase and the smoothness of the true functions. As the estimators μ^s, f^s are consistent, equation 3.22 also implies that the gradient estimator $\hat{\nabla}_i$ is consistent as well, i.e. $\|\nabla - \mathbb{E}_s[\hat{\nabla}]\|_2 \to 0$ as the number of sleep samples $S \to \infty$.

Variance of the gradient

We proceed with showing that the variance of the gradient estimate $\hat{\nabla}$ can also be bounded and that the bound depends on the norms (smoothness) of $\mu^s \in \mathcal{H}_\Gamma, f^s, h \in \mathcal{H}_\mathcal{Z}$. As Thm. 3.2.1 requires the sum of the variances for each component to be bounded (i.e. $\mathbb{E}[\|\nabla_t - G_t\|^2] \leq \sigma^2$), it is sufficient to bound the marginal variances individually. Under the assumptions of theorem 3.2.2

and further assuming $\|f^s\|_{\mathcal{H}_\mathcal{Z}} < \lambda_f, \|h\|_{\mathcal{H}_\mathcal{Z}} < \lambda_h$, with $\lambda_f, \lambda_h \in \mathbb{R}^+$ we have:

$$\mathbb{V}\mathrm{ar}_s[\hat{\nabla}_i] \leq 4\kappa^2\big(|S(x)|\lambda_\mu\lambda_f + \lambda_\mu\lambda_h\big)^2 \tag{3.23}$$

See appendix A.2 for a derivation.

So far we focused on the gradients with respect to parameters appearing in $\log p(x|z_1;\theta_0)$. Analogously, the bias and variance can be bounded for parameters higher up in the generative model $\theta_1, ...\theta_L$. The absolute error of the gradient for parameters θ_l linking z_l and z_{l+1} in the generative model:

$$\begin{aligned}|\nabla_{l,i} - \mathbb{E}_s\,\hat{\nabla}_{l,i}| \leq \kappa\big(&\mathbb{E}_s\left[\|\mu_l - \mu_l^s\|_{\mathcal{H}_\Gamma}\right]\|f_l\|_{\mathcal{H}_\mathcal{Z}} + \mathbb{E}_s\left[\|\mu_{l+1} - \mu_{l+1}^s\|_{\mathcal{H}_\Gamma}\right]\|h_{l+1}\|_{\mathcal{H}_\mathcal{Z}} \\ &+ \lambda_{\mu_l}\,\mathbb{E}_s\left[\|f_l - f_l^s\|_{\mathcal{H}_\mathcal{Z}}\right]\big)\end{aligned} \tag{3.24}$$

And similarly for the variance:

$$\mathbb{V}\mathrm{ar}_s[\hat{\nabla}_{l,i}] \leq 4\kappa^2\big(\lambda_{\mu_l}\lambda_{f_l} + \lambda_{\mu_{l+1}}\lambda_{h_{l+1}}\big)^2 \tag{3.25}$$

Special case

In the case where both RKHSs $\mathcal{H}_K$ and $\mathcal{H}_L$ are finite dimensional, the Kernel-HM admits the DDC-HM as a special case. The mean embedding map μ is a matrix with dimensions of the corresponding feature spaces on $\mathcal{X}$ and $\mathcal{Z}$. The norm $\|.\|_\Gamma$ corresponds to the Frobenius norm.

3.3 Discussion

Here, we introduced the Kernel Helmholtz machine, an algorithm for learning hierarchical generative models that optimises the marginal log-likelihood directly using approximate gradients. The model can be seen as a generalisation of the DDC Helmholtz Machine, built on the theory of conditional mean embeddings of distributions. The formulation of the algorithm allowed us to make formal statements about convergence; we were able to relate the different sources of approximation error to the speed of convergence and behaviour around the

stationary point of the log-likelihood. This is in contrast to methods relying on the variational free energy, where a lower bound is optimised, and in general there is no relationship between the stationary points of the objective being optimised and the desired objective.

The only previous work that we are aware of on the convergence of the Helmholtz Machine and wake-sleep algorithm was applied to an analytically tractable model (a factor analysis model with a single latent variable; Ikeda et al. (1999)).

An important assumption we made here is that the estimation problems in the kernel-HM are well specified, that is, $\mu \in \mathcal{H}_\Gamma$ and $f_{l,i}, h_{l,i} \in \mathcal{H}_\mathcal{Z}$. Showing consistency of the gradient estimates when this is not the case—as for the DDC Helmholtz machine corresponding to a finite dimensional RKHS—will be important future work.

A Appendix

A.1 Computing and learning gradients for the generative model parameters

The variational free energy for a hierarchical generative model over observations x with latent variables $z_1 \dots z_L$ can be written:

$$\mathcal{F}(q,\theta) = \langle \log p(x|z_1) \rangle_{q(z_1)} + \sum_{l=1}^{L-1} \langle \log p(z_l|z_{l+1}) \rangle_{q(z_l,z_{l+1})} \tag{3.26}$$

$$+ \langle \log p(z_L) \rangle_{q(z_L)} + H[q(z_1 \dots z_L)] \tag{3.27}$$

where the distributions q represent components of the approximate posterior and $H[\cdot]$ is the Shannon entropy. We take each conditional distribution to have exponential family form. Thus (for example)

$$\log p(z_l|z_{l+1}) = g_l(z_{l+1},\theta_l)^T S_l(z_l) - \Phi_l(g_l(z_{l+1},\theta_l)) \tag{3.28}$$

from which it follows that:

$$\nabla_{\theta_l} \log p(z_l|z_{l+1}) = S_l(z_l)^T \nabla_{\theta_l} g_l(z_{l+1},\theta_l) - \Phi_l'(g_l(z_{l+1},\theta_l)) \nabla_{\theta_l} g_l(z_{l+1},\theta_l) \tag{3.29}$$

$$= S_l(z_l)^T \nabla_{\theta_l} g_l(z_{l+1},\theta_l) - \mu_{z_1|z_{l+1}} \nabla_{\theta_l} g_l(z_{l+1},\theta_l) \tag{3.30}$$

where we have used the standard result that the derivative of the log normaliser of an exponential family distribution with respect to the natural parameter is the mean parameter

$$\mu_{z_l|z_{l+1}} = \langle S_l(z_l) \rangle_{p_{\theta_l}(z_l|z_{l+1})} . \tag{3.31}$$

Thus, it follows that:

$$\nabla_{\theta_0}\mathcal{F} = \nabla_{\theta_0}\langle \log p(x|z_1)\rangle_{q(z_1)} \tag{3.32}$$

$$= S_0(x)^T\langle \nabla g(z_1,\theta_0)\rangle_{q(z_1)} - \langle \mu_{x|z_1}^T \nabla g(z_1,\theta_0)\rangle_{q(z_1)} \tag{3.33}$$

$$\nabla_{\theta_l}\mathcal{F} = \nabla_{\theta_l}\langle \log p(z_l|z_{l+1})\rangle_q \tag{3.34}$$

$$= \langle S_l(z_l)^T \nabla g(z_{l+1},\theta_l)\rangle_{q(z_l,z_{l+1})} - \langle \mu_{z_l|z_{l+1}}^T \nabla g(z_{l+1},\theta_l)\rangle_{q(z_{l+1})} \tag{3.35}$$

(for $l = 1 \dots L-1$) and

$$\nabla_{\theta_L}\mathcal{F} = \nabla_{\theta_L}\langle \log p(z_L)\rangle_q = \langle S_L(z_L)\rangle - \nabla\Phi(\theta_L) \tag{3.36}$$

These are the gradients that must be computed in the wake phase.

In order to compute these gradients from the DDC posterior representation we need to express the functions of z_l that appear in Eqs. 3.33–3.36 as linear combinations of the encoding functions $\boldsymbol{\psi}(z_l)$. The linear coefficients can be learnt using samples from the generative model produced during the sleep phase. The example given in chapter 2, section 2.2.2 is the most straightforward. We wish to find

$$\nabla_{\theta_0} g(z_1,\theta_0) \approx \sum_i \alpha_0^{(i)} \psi_i(z_1) \tag{3.37}$$

in which the coefficients α_0 can be obtained from samples by evaluating the gradient of g with respect to θ_0 at $z_1^{(s)}$ and minimising the squared error:

$$\alpha_0 \leftarrow \operatorname{argmin} \sum_s \left(\nabla_\theta g(z_1^{(s)},\theta_0) - \alpha_0 \cdot \boldsymbol{\psi}(z_1^{(s)})\right)^2. \tag{3.38}$$

Once these coefficients have been found in the sleep phase, the wake phase

expectations are found from the DDC recognition model very simply:

$$\langle \nabla_{\theta_0} g(z_1, \theta_0) \rangle_{q(z_1)} \approx \sum_i \alpha_0^{(i)} \langle \psi_i(z_1) \rangle_{q(z_1)} \tag{3.39}$$

$$= \sum_i \alpha_0^{(i)} r_1^{(i)}(x, \phi_1) \tag{3.40}$$

Some of the gradients (see Eq.3.35) require taking expectations using the joint posterior distribution $q(z_l, z_{l+1}|x)$. However, the recognition network as we have described it in the main text only contains information about the marginal posteriors $q(z_l|x), q(z_{l+1}|x)$. It turns out that it is nevertheless possible to estimate these expectations without imposing an assumption that the approximate posterior factorises across the layers z_l, z_{l+1}.

We begin by noticing that due to the structure of the generative model the posterior distribution q can be factorised into $q(z_{l+1}, z_l|x) = p(z_{l+1}|z_l)q(z_l|x)$ without any further assumptions (where $p(z_{l+1}|z_l)$) is the conditional implied by the generative model. Thus, we can rewrite the term in Equation 3.35 as:

$$\langle S_l(z_l)^T \langle \nabla_{\theta_l} g(z_{l+1}, \theta_l) \rangle_{p(z_{l+1}|z_l)} \rangle_{q(z_l)}$$

Now, we can replace the expectation $\langle \nabla_{\theta_l} g(z_{l+1}, \theta_l) \rangle_{p(z_{l+1}|z_l)}$ by an estimate using *sleep* samples, i.e. for each pair of samples $\{z_l^{(s)}, z_{l+1}^{(s)}\}$ from the prior, we have a single sample from the true posterior distribution $z_{l+1}^{(s)} \sim p(z_{l+1}|z_l = z_l^{(s)})$. Thus, we obtain coefficients α_l during the sleep phase so:

$$\alpha_l = \text{argmin} \sum_s \left(S_l(z_l^{(s)})^T \nabla_{\theta_l} g(z_{l+1}^{(s)}, \theta_l) - \alpha_l \cdot \boldsymbol{\psi}(z_l^{(s)}) \right)^2 \tag{3.41}$$

which will converge so that

$$\alpha_l \cdot \boldsymbol{\psi}(z_l) \approx S_l(z_l)^T \langle \nabla_{\theta_l} g(z_{l+1}, \theta_l) \rangle_{p(z_{l+1}|z_l)} . \tag{3.42}$$

Now, during the wake phase it follows that

$$\left\langle S_l(z_l)^T \nabla_{\theta_l} g(z_{l+1}, \theta_l) \right\rangle_{q(z_{l+1}, z_l)} = \langle S_l(z_l)^T \langle \nabla_{\theta_l} g(z_{l+1}, \theta_l) \rangle_{p(z_{l+1}|z_l)} \rangle_{q(z_l)} \tag{3.43}$$

$$\approx \left\langle \sum_i \alpha_l^{(i)} \psi_i(z_l) \right\rangle_{q(z_l)} \tag{3.44}$$

$$= \sum_i \alpha_l^{(i)} r_l^{(i)}(x, \phi_l) \tag{3.45}$$

Thus, all the function approximations needed to evaluate the gradients are carried out by using samples from the generative model (*sleep* samples) as training data. The full set of necessary function approximations with (matrix) parameters describing the linear mappings denoted by $\{\alpha_l\}_{l=0\ldots L}, \{\beta_l\}_{l=1\ldots L}$ is:

$$\alpha_0 : \boldsymbol{\psi}(z_1^{(s)}) \mapsto \nabla_\theta g(z_1^{(s)}, \theta_0) \tag{3.46}$$

$$\beta_l : \boldsymbol{\psi}(z_l^{(s)}) \mapsto \mu^T_{z_{l-1}|z_l^{(s)}} \nabla_\theta g(z_l^{(s)}, \theta_{l-1}) \tag{3.47}$$

$$\alpha_l : \boldsymbol{\psi}(z_l^{(s)}) \mapsto S_l(z_l^{(s)}) \nabla_\theta g(z_{l+1}^{(s)}, \theta_l) \tag{3.48}$$

$$\alpha_L : \boldsymbol{\psi}(z_L^{(s)}) \mapsto S_L(z_L^{(s)}) \tag{3.49}$$

where the expressions on the right hand side are easy to compute given the current parameters of the generative model. Note that $\mu_{z_{l-1}|z_l^{(s)}}$ appearing in equation 3.47 are expectations that can be evaluated analytically for tractable exponential family models, as a consequence of the conditionally independent structure of the generative model. Alternatively, they can also be estimated using the *sleep* samples, by training

$$\beta_l : \boldsymbol{\psi}(z_l^{(s)}) \mapsto S_{l-1}(z_{l-1}^{(s)})^T \nabla_\theta g(z_l^{(s)}, \theta_{l-1}) \tag{3.50}$$

Finally, putting the sleep and the wake phase together, the updates for

the generative parameters during the wake phase are:

$$\begin{aligned}\Delta\theta_0 &\propto S_0(x)^T\alpha_0 r_1(x,\phi_1) - \beta_1 r_1(x,\phi_1)\\ \Delta\theta_l &\propto \alpha_l r_l(x,\phi_l) - \beta_{l+1} r_{l+1}(x,\phi_{l+1})\\ \Delta\theta_L &\propto \alpha_L r_L(x,\phi_L) - \nabla_{\theta_L}\Phi(\theta_L)\end{aligned} \tag{3.51}$$

A.2 Bias and variance of the Kernel-HM gradient estimator

Theorem A.1. *Assume that*
$\exists f \in \mathcal{H}_\mathcal{Z} : \frac{d}{d\theta_i} g(z_1,\theta) = \langle f, k_\mathcal{Z}(z_1,.)\rangle_{\mathcal{H}_\mathcal{Z}}$,
$\exists h \in \mathcal{H}_\mathcal{Z} : m_{x|z_1}(z_1)^T \frac{d}{d\theta_i} g(z_1,\theta) = \langle h, k_\mathcal{Z}(z_1,.)\rangle_{\mathcal{H}_\mathcal{Z}}$,

and $f^s, \mu^s \in \mathcal{H}_\mathcal{Z}$

Further assume that $\exists \mu \in \mathcal{H}_\Gamma, \mu : \mathcal{X} \to \mathcal{H}_\mathcal{Z}$ *such that* $E_{z|x}[h(z)] = \langle h, \mu(x)\rangle_{\mathcal{H}_\mathcal{Z}} \quad \forall x \in \mathcal{X}, h \in \mathcal{H}_\mathcal{Z}$,

$\|\mu^s\|_{\mathcal{H}_\Gamma} \leq \lambda_\mu, \|\Gamma(x,x)\|_{op} \leq \kappa, \quad \lambda_\mu, \kappa \in \mathbb{R}^+$

Then:

$$\begin{aligned}\left|\nabla_i - \mathbb{E}_s[\hat{\nabla}_i]\right| \leq\, & \mathbb{E}_s\left[\|\mu - \mu^s\|_{\mathcal{H}_\Gamma}\right]\left(|S(x)| * \|f\|_{\mathcal{H}_\mathcal{Z}} + \|h\|_{\mathcal{H}_\mathcal{Z}}\right)\\ & + \lambda_\mu |S(x)|\, \mathbb{E}_s[\|f - f^s\|_{\mathcal{H}_\mathcal{Z}}]\end{aligned} \tag{3.52}$$

Proof.

$$\begin{aligned}\left|\nabla_i^* - \mathbb{E}_s\,\hat{\nabla}_i\right| =& |S(x)\langle f, \mu(x)\rangle_{\mathcal{H}_\mathcal{Z}} + \langle h, \mu(x)\rangle_{\mathcal{H}_\mathcal{Z}}\\ & - \mathbb{E}_s[S(x)\langle f^s, \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}} + \langle h, \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}}]|\\ =& |\,\mathbb{E}_s[S(x)\big(\langle f, \mu(x)\rangle_{\mathcal{H}_\mathcal{Z}} - \langle f^s, \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}}\big)\\ & + \langle h, \mu(x)\rangle_{\mathcal{H}_\mathcal{Z}} - \langle h, \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}}]|\end{aligned}$$

Using the identity:

$$\langle f, \mu(x)\rangle_{\mathcal{H}_\mathcal{Z}} - \langle f^s, \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}} = \langle f - f^s, \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}} + \langle f, \mu(x) - \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}} \tag{3.53}$$

$$\begin{aligned}
&= |\,\mathbb{E}_s[S(x)\big(\langle f - f^s, \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}} + \langle f, \mu(x) - \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}}\big) \\
&\quad + \langle h, \mu(x) - \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}}]| \\
&\text{As } \mu \in \mathcal{H}_\Gamma, \text{ we can use } \langle f, \mu(x)\rangle_{\mathcal{H}_\mathcal{Z}} = \langle \mu, \Gamma_x f\rangle_{\mathcal{H}_\Gamma} \\
&= |\,\mathbb{E}_s[S(x)\big(\langle \mu^s, \Gamma_x(f - f^s)\rangle_{\mathcal{H}_\Gamma} + \langle \mu - \mu^s, \Gamma_x f\rangle_{\mathcal{H}_\Gamma}\big) \\
&\quad + \langle \mu - \mu^s, \Gamma_x h\rangle_{\mathcal{H}_\Gamma}]| \\
&\leq \mathbb{E}_s[|S(x)|\big(|\langle \mu^s, \Gamma_x(f - f^s)\rangle_{\mathcal{H}_\Gamma}| + |\langle \mu - \mu^s, \Gamma_x f\rangle_{\mathcal{H}_\Gamma}|\big) \\
&\quad + |\langle \mu - \mu^s, \Gamma_x h\rangle_{\mathcal{H}_\Gamma}|]| \\
&\leq \mathbb{E}_s\left[|S(x)|\big(\|\mu^s\|_{\mathcal{H}_\Gamma}\|\Gamma_x(f - f^s)\|_{\mathcal{H}_\Gamma} + \|\mu - \mu^s\|_{\mathcal{H}_\Gamma}\|\Gamma_x f\|_{\mathcal{H}_\Gamma}\big)\right. \\
&\quad \left. + \|\mu - \mu^s\|_{\mathcal{H}_\Gamma}\|\Gamma_x h\|_{\mathcal{H}_\Gamma}\right] \\
&\leq \kappa\,\mathbb{E}_s\left[|S(x)|\big(\|\mu^s\|_{\mathcal{H}_\Gamma}\|f - f^s\|_{\mathcal{H}_\mathcal{Z}} + \|\mu - \mu^s\|_{\mathcal{H}_\Gamma}\|f\|_{\mathcal{H}_\mathcal{Z}}\big)\right. \\
&\quad \left. + \|\mu - \mu^s\|_{\mathcal{H}_\Gamma}\|h\|_{\mathcal{H}_\mathcal{Z}}\right] \\
&\leq \kappa\,\mathbb{E}_s\left[\|\mu - \mu^s\|_{\mathcal{H}_\Gamma}\right]\big(|S(x)|\|f\|_{\mathcal{H}_\mathcal{Z}} + \|h\|_{\mathcal{H}_\mathcal{Z}}\big) \\
&\quad + \kappa\lambda_\mu|S(x)|\,\mathbb{E}_s\left[\|f - f^s\|_{\mathcal{H}_\mathcal{Z}}\right] \qquad \square
\end{aligned}$$

Where we have used the assumption that the operator $\Gamma(x, x)$ has norm $\|\Gamma(x, x)\|_{op} \leq \kappa, \kappa \in \mathbb{R}^+$, therefore:

$$\begin{aligned}
\|\Gamma_x h\|_{\mathcal{H}_\Gamma} &= \sqrt{\langle \Gamma_x h, \Gamma_x h\rangle_{\mathcal{H}_\Gamma}} = \sqrt{\langle h, (\Gamma_x h)(x)\rangle_{\mathcal{H}_\mathcal{Z}}} \\
&\leq \sqrt{\|h\|_{\mathcal{H}_\mathcal{Z}} * \|(\Gamma_x h)(x)\|_{\mathcal{H}_\mathcal{Z}}} \\
&= \sqrt{\|h\|_{\mathcal{H}_\mathcal{Z}} * \|\Gamma(x, x)h\|_{\mathcal{H}_\mathcal{Z}}} \leq \kappa\|h\|_{\mathcal{H}_\mathcal{Z}}
\end{aligned}$$

$$\begin{aligned}
\mathbb{V}\mathrm{ar}_s[\hat{\nabla}_i] =& \mathbb{E}_s\left[\left(S(x)\langle f^s, \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}} + \langle h, \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}}\right.\right. && (3.54)\\
&\left.\left. - \mathbb{E}_s[S(x)\langle f^s, \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}} + \langle h, \mu^s(x)\rangle_{\mathcal{H}_\mathcal{Z}}]\right)^2\right] && (3.55)\\
=& \mathbb{E}_s\left[\left(S(x)\langle \mu^s, \Gamma_x f^s\rangle_{\mathcal{H}_\Gamma} + \langle \mu^s, \Gamma_x h\rangle_{\mathcal{H}_\Gamma}\right.\right. && (3.56)\\
&\left.\left. - \mathbb{E}_s[S(x)\langle \mu^s, \Gamma_x f^s\rangle_{\mathcal{H}_\Gamma} + \langle \mu^s, \Gamma_x h\rangle_{\mathcal{H}_\Gamma}]\right)^2\right] && (3.57)\\
\leq& \mathbb{E}_s\left[\left(|S(x)\langle \mu^s, \Gamma_x f^s\rangle_{\mathcal{H}_\Gamma}| + |\langle \mu^s, \Gamma_x h\rangle_{\mathcal{H}_\Gamma}|\right.\right. && (3.58)\\
&\left.\left. + \mathbb{E}_s[|S(x)\langle \mu^s, \Gamma_x f^s\rangle_{\mathcal{H}_\Gamma}| + |\langle \mu^s, \Gamma_x h\rangle_{\mathcal{H}_\Gamma}|]\right)^2\right] && (3.59)\\
\leq& \kappa^2\, \mathbb{E}_s\left[\left(|S(x)|\|\mu^s\|_{\mathcal{H}_\Gamma}\|f^s\|_{\mathcal{H}_\mathcal{Z}} + \|\mu^s\|_{\mathcal{H}_\Gamma}\|h\|_{\mathcal{H}_\mathcal{Z}}\right.\right. && (3.60)\\
&\left.\left. + \mathbb{E}_s[|S(x)|\|\mu^s\|_{\mathcal{H}_\Gamma}\|f^s\|_{\mathcal{H}_\mathcal{Z}} + \|\mu^s\|_{\Gamma}\|h\|_{\mathcal{H}_\mathcal{Z}}]\right)^2\right] &&\\
\leq& 4\kappa^2\left(|S(x)|\lambda_\mu\lambda_f + \lambda_\mu\lambda_h\right)^2 && (3.61)
\end{aligned}$$

Part II

Learning and computing with uncertainty in the brain

Chapter 4

Neural Representations of Uncertainty

1 Behavioural evidence for probabilistic computations

It was perhaps first put forward by Hermann von Helmholtz that sensory perception should be viewed as a process of *unconscious inference* (Helmholtz, 1867). He argued that percepts are influenced not only by the physical properties of sensory stimuli directly but also by one's (often unconscious) interpretations or conclusions derived from them.

A growing body of behavioural evidence supports the hypothesis that the brain is able to perform probabilistic reasoning, often resulting in near optimal performance in tasks that require explicit handling of uncertainty. While Bayesian theories of the brain have been most dominant in the study of perception (Knill and Richards, 1996; van Beers et al., 1999; Ernst and Banks, 2002; Knill, 1998; Jacobs, 1999), experimental evidence extends to a wide range of brain functions from sensorimotor (Wolpert et al., 1995; Körding and Wolpert, 2004; Todorov, 2004) to cognitive (Tenenbaum and Griffiths, 2002; Chater et al., 2006; Steyvers et al., 2006) domains.

A critical feature of perceptual experiments demonstrating probabilistic computations and representations is that they require the use of uncertainty

information about the sensory evidence available on each trial, rather than just point estimates. This requirement can be quite subtle, as not all probabilistic tasks require explicit handling of uncertainty. Yang and Shadlen (2007) designed a weather-prediction task where monkeys needed to use probabilistic evidence provided by four different shapes to predict which side will be rewarded in a given trial. While the neural data was consistent with probabilistic evidence integration, the optimal solution to the behavioural task did not necessarily require probabilistic reasoning. As the monkeys received trial-to-trial feedback, it is in fact possible to solve this task without having to keep track of uncertainty about the correct decision.

Multisensory cue integration has been a powerful paradigm for characterising Bayesian perception, allowing for trial-to-trial manipulation of the relative reliability of the combined modalities. In an early study on this subject, Ernst and Banks (2002) asked subjects to report the height of a bar (relative to a reference bar) using visual and haptic information. The reliability of visual evidence was varied across trials and information from the two different modalities provided slightly conflicting evidence about the height of the bar. They found that subjects weighted the visual and haptic information according to their reliability, consistent with Bayesian cue combination. Importantly, subjects received no feedback throughout the experiment, thus, arguably the task could probe whether they used existing sensory representations of uncertainty.

Several other human psychophysics studies have demonstrated similar results including visual-auditory (Battaglia and Schrater, 2007; Alais and Burr, 2004; Shams et al., 2005), visual-proprioceptive (van Beers et al., 1999) and visual-vestibular (MacNeilage et al., 2007) cue integration experiments (see Trommershauser et al. (2011) for an extensive review of the subject).

A complementary line of work has studied multimodal sensory integration and the corresponding neural representations in non-human primates (Gu et al., 2008; Morgan et al., 2008; Fetsch et al., 2009). In accord with human data, they found that macaques were able to take into account uncertainty about

visual information when combining it with vestibular self-motion cues in a heading direction discrimination task.

Other studies have focused on how prior statistical knowledge influences decisions when combined with sensory evidence. Körding and Wolpert (2004) have shown using a sensorimotor task that subjects are able to take into account both the prior distribution induced by the task and uncertainty about sensory feedback. It has also been shown that subjects use prior distributions based on natural statics to estimate passage of time (Ahrens and Sahani, 2011) or speed of visual stimuli (Stocker and Simoncelli, 2006) in a way that is consistent with Bayesian inference.

Whiteley and Sahani (2008) used a visual discrimination task with asymmetric reward contingencies varied across trials and withheld trial-to-trial feedback. They found that subjects adjusted their decisions to approximately maximise expected rewards. While the underlying sensory uncertainty was not explicitly modulated in this experiment, it is required to make decisions that maximise the expected reward on each trial.

These experiments provide converging evidence about the brain's ability to learn about and compute with uncertainty. The high degree of flexibility with which behaviour is adapted to changes in uncertainty about task-relevant variables, suggests that neural systems represent distributional information, at least implicitly.

2 Uncertainty in neural systems

While there is strong behavioural evidence that neural systems learn to perform probabilistic inference, how uncertainty is represented in population of neurons remains elusive. In the following, we introduce some relevant concepts and terminology in this framework that will allow us to discuss specific proposals for how neural systems might perform probabilistic inference.

Consistent with experimental data, it has been postulated that the brain acquires an *internal model* capturing the statistics of its sensory environment.

Internal models can be mathematically formalised as latent variable *generative models* discussed in Part I of this thesis. Generative models summarise the organism's knowledge about *latent variables* not directly observable and how they give rise to observed sensory stimuli in the form of a probability distribution. It is often instructive to write the joint distribution over latent and observed variables as a product of the *likelihood* function and the *prior* distribution:

$$p(x, z) = \underbrace{p(x|z)}_{\text{likelihood}} \underbrace{p(z)}_{\text{prior}} \tag{4.1}$$

where x, z denote the collection of all observed and latent variables, respectively.

Latent variables in directed acyclic graphical models (DAGs), such as the deep exponential family models introduced in section 1.2, are also referred to as *latent causes*[1] or *explanatory* variables. DAGs can encode structure through conditional independence relationships, e.g. when the presence of objects in the scene introduce correlations between some lower level features of the image, but when conditioned on the identity of the object those features become independent.

Figure 4.1 illustrates the computational problem using the example of vision. In this setting, the objective for the generative model is to capture the distribution of natural images, while learning a prior distribution over the relevant latent variables (e.g. objects, shapes, textures). Generative models used to model natural image or sound statistics are often hierarchical, reflecting the hierarchical composition of the real world.

In machine learning and unsupervised learning it is customary to treat images (or raw audio signals) as observed variables. However, from the brains perspective there is an additional step of sensory transduction. So in practice, peripheral receptor activations should be considered as observations. To illustrate this subtlety, in fig. 4.1 we have introduced an intermediate variable $\hat{x}$ (in grey), denoting the transduced stimulus (e.g. retinal activity). The

[1]Note that the statistical relationships encoded in DAGs do not correspond to causality in a formal sense.

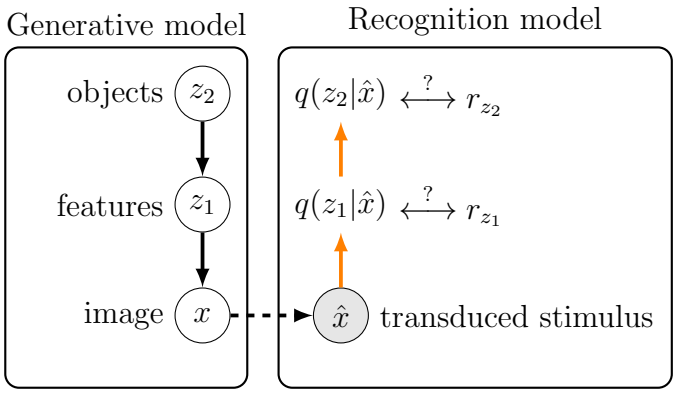


Figure 4.1: Perception as Bayesian inference.
Illustration of the generative and recognition models using the example of vision. Black arrows represent conditional dependences between random variables denoted by circles, nodes with grey background represent observed variables. Orange arrows depict mappings corresponding to a possible recognition model reflecting the structure of the generative model. r_{z_1} and r_{z_2} correspond to neural activity (e.g. firing rate) representing the inferred (approximate) posterior distributions $q(z_1|\hat{x})$ and $q(z_2|\hat{x})$.

relationship between the image presented on a screen and the retinal image (dashed arrow in fig. 4.1) is typically stochastic, with noise being introduced by the variable number of photons reaching the eye and *sensory noise*, the variability in receptor responses.

Ultimately, the problem that is most relevant for sensory systems is learning to invert the generative model, that is, performing inference over the latent variables given sensory observations. This process requires probabilistic reasoning as the generative model does not typically specify a deterministic relationship between latent and observed variables but describes dependencies using probability distributions. Formally, the result of inference is a posterior distribution over the possible latent causes that might have generated the sensory stimulus.

A biologically plausible way to implement inference is through a recognition model, a function that maps observations to the inferred posterior distribution (Gershman and Goodman, 2014; Dasgupta et al., 2018). Performing inference using a recognition or inference network is also referred to as amortised inference.

The computational cost of computing the corresponding posterior for individual observations is amortised by a learned model—typically a neural network—with parameters shared across observations.

The posterior distributions $q(z_i|x)$ returned by the recognition model are intended to capture uncertainty about latent variables z_i. One source of this uncertainty is variability in the neural responses due to for example noisy sensory transduction or variability in synaptic release. That is, given retinal or V1 activity there will be a degree of uncertainty about the physical attributes of an image (e.g. colour and luminance of each pixel). Neural and sensory noise can be thought of as a form of measurement noise and can be in some cases averaged out by longer stimulus presentation time or pooling over a larger population of neurons.

Perhaps more importantly, uncertainty also arises from ambiguity in the generative process itself, i.e. multiple settings of the latent variables might give rise to the same image. A ubiquitous example is when the inference problem is underdetermined, as in the case of inferring three-dimensional shapes from images or when the scene contains some form of occlusion. In these cases, the resulting uncertainty is irreducible, sometimes referred to as *inferential* uncertainty, and would be present even with fully deterministic neural machinery.

Capturing inferential uncertainty correctly can be vital especially when reasoning about more abstract, behaviourally relevant variables, such as the presence of a predator. Yet, it is often overlooked in some of the most influential works on this topic, e.g. from Ma et al. (2006):

> It is critical to realize, however, that variability and uncertainty go hand in hand: if neuronal variability did not exist, that is, if neurons were to fire in exactly the same way every time you saw the same object, then you would always know with certainty what object was presented.

Relatedly, while variability in neural responses introduces an additional

source of uncertainty, the neural representation of uncertainty need not rely on variability per se. Deterministic codes, such as the mean and variance of a Gaussian, can convey information about uncertainty. We will return to these points in the following sections when discussing concrete proposals for neural representations of distributions.

3 Proposals for neural representation of uncertainty

The wealth of behavioural evidence for probabilistic computations motivated a number of theoretical proposals for how uncertainty (i.e. probability distributions) may be represented and how the relevant computations are performed by neural systems. In this section, we review the most significant proposals, discussing their key properties and potential short-comings.

3.1 Linear density codes

Possibly the earliest work considering population codes for probability distributions—rather than just point estimates—was Anderson (1994). Anderson introduced a coding scheme closely related to kernel density estimation (KDE, Rosenblatt (1956); Parzen (1962)), a simple non-parametric method for estimating distributions. The KDE code assigns a basis or kernel function $\psi_i(s)$ to each neuron i in the population. Then, the distribution over the variable s represented by the population can be linearly reconstructed using the population activity and the basis functions:

$$p(s) = \sum_i \tilde{r}_i \psi_i(s) \tag{4.2}$$

where $\tilde{r}_i$ is the normalised firing rate of neuron i, ensuring that the distribution $p(s)$ integrates to 1. If the basis functions $\psi_i(s)$ are density functions themselves, a suitable choice is $\tilde{r}_i = r_i/N$, where r_i is the firing rate of neuron i and N is the size of the population. Figure 4.2 illustrates the relationship between the basis functions, neural activity and the resulting distribution.

Importantly, the kernel functions ψ_i are not directly related to tuning functions typically measured in experiments, and need not be explicitly represented. When the kernel functions are densities (i.e. non-negative functions) the range of spatial frequencies that the KDE code can represent is limited by the width of the kernel functions. The KDE code—or linear density code—is formulated using a *decoding* perspective, that is, it provides a recipe for how the density function $p(s)$ can be decoded from the population. Deriving an encoding model, that predicts neural activity consistent with the KDE code for a given distribution $p(s)$, is non-trivial in general.

Given a set of kernel functions and a distribution $p^\star(s)$, the corresponding neural activity can be found by minimising a divergence measure (e.g. the Kullback-Leibler divergence) between the distribution to be encoded and the distribution implied by firing rates r: $r = \underset{r}{\operatorname{argmin}}\, D[p^\star(s)||p_r(s)]$

Anderson (1994) suggested an alternative method of *projections*, where the firing rates r_i can be found analogously to a linear regression problem:

$$r_i = \int p^\star(s) f_i(s) ds \tag{4.3}$$

$$\text{with} \quad f_i(s) = \sum_j A^{-1}_{ij} \psi_j(s) \tag{4.4}$$

$$A_{ij} = \int \psi_i(s) \psi_j(s) ds \tag{4.5}$$

Intuitively, in equation 4.3 the function $f_i(s)$ plays the role of a tuning curve, and the firing rate of neuron i is equal to the projection of the distribution $p^\star(s)$ onto the corresponding tuning curve.

3.2 Distributional population codes

Zemel et al. (1998) started from a simple statistical description of population responses, often called the Poisson encoding model, that has been widely used to study coding properties of noisy neural populations (Seung and Sompolinsky, 1993; Salinas and Abbott, 1994; Snippe, 1996). The Poisson model is motivated by the apparent neural variability observed on repeated presentation of the same

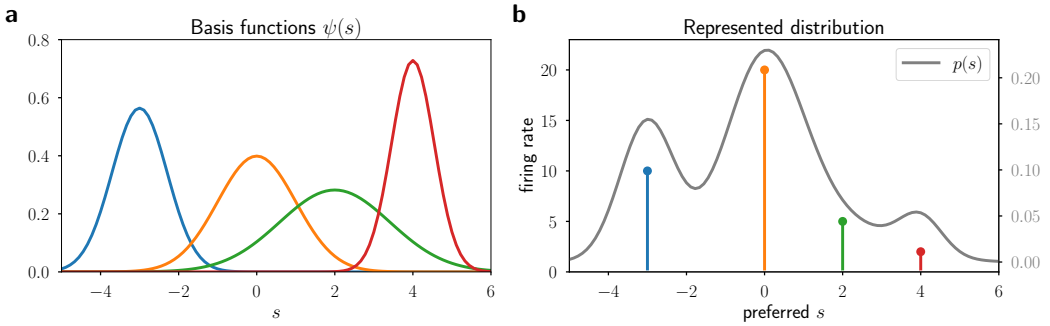


Figure 4.2: Kernel density estimator (KDE) code. (a) Basis functions corresponding to individual neurons in the population. (b) Stem plot (in color) shows firing rates of four neurons corresponding to the basis functions in (a). The distribution represented can be decoded by a kernel density estimator, i.e. a sum of the basis functions weighted by the firing rates.

stimulus value. The neural responses thus can be captured as a conditional distribution given the stimulus s:

$$p(r_i|s) = \exp(-f_i(s))\frac{f_i(s)^{r_i}}{r_i!} \tag{4.6}$$

where r_i is the firing rate of neuron i in the population and f_i is the corresponding tuning curve describing the mean response.

As the authors argued, the Poisson model has a crucial limitation that it considers encoding only a single value s rather than a distribution $p(s)$. While the stochasticity in the neural activity introduces uncertainty about the value of s when trying to decode it from the responses, the decoded distribution $p(s|r)$ necessarily collapses to a Dirac-delta function as the number of neurons or the observed time interval grows.

Motivated by the shortcomings of the Poisson model, Zemel et al. introduced a coding scheme that describes response variability given an explicit distribution over s: $p(r|p(s))$, the extended Poisson model or, as it was later referred to, the *distributional population codes* (Zemel and Dayan, 1999). Distributional population codes (DPCs) encode uncertainty in a population of neurons using a set of expectations of *encoding* functions with respect to the

represented distribution:

$$\mathbb{E}[r_i] = \mathbb{E}_{p(s)}[\psi_i(s)], \tag{4.7}$$

where $\mathbb{E}[r_i]$ is the mean firing rate of neuron i and ψ_i is the corresponding encoding function. The encoding functions are closely related to classical tuning curves, with an exact correspondence when the encoded distribution is a Dirac-delta function (i.e. when only a single value is encoded). The expectation with respect to a Dirac-delta function centred at s_0 amounts to evaluating the encoding function at that value:

$$\mathbb{E}_{\delta(s-s_0)}[\psi_i(s)] = \psi_i(s_0) \tag{4.8}$$

DPCs have also been interpreted as convolutional codes (Pouget et al., 2003), viewing equation 4.7 as a convolution between the density $p(s)$ and kernel functions $\psi_i(s)$.

Eq. 4.7 defines an encoding model that describes neural activity as a function of the encoded distribution. Unlike in the case of KDE codes, recovering the density function from a DPC is non-trivial in general. Zemel et al. (1998) proposed a maximum a posteriori decoding scheme assuming Poisson neural variability with regularisation ensuring smoothness of the density function.

DPCs have been generalised to be able to jointly encode uncertainty and multiplicity over sensory variables (Sahani and Dayan, 2003). Multiplicity is the simultaneous presence of multiple distinct stimuli, for example, multiple sound sources at different azimuth angles. *Doubly distributional population codes* (DDPC) represent uncertainty about a *function* m—capturing multiplicity of the variable s—rather than simply uncertainty about the variable s itself. DDPCs follow much the same logic as their DPC cousin, m is first encoded using a set of features ψ_i:

$$\phi_i(m) = \int m(s)\psi_i(s)ds \tag{4.9}$$

Then, neural activity is set to be the expectation of features $\phi_i(m)$ with respect to the distribution over *functions* $m(s)$:

$$\mathbb{E}[r_i] = \mathbb{E}_{p(m)}[\phi_i(m)] \tag{4.10}$$

Note that in our own work, we refer to a close variant of DPCs as *distributed distributional code* or DDC (see also section 2.1). DDCs differ from the original formulation of the DPCs in that they specify only the encoding model (eq. 4.7), providing an implicit representation of the distribution through a set of expectations. This coding scheme provides a set of constraints on a distribution in the form of its generalised moments, analogous to the first, second, third etc. moments in statistics. DDCs can be viewed as specifying an exponential family distribution—the maximum entropy distribution that satisfies those constraints— with sufficient statistics vector $\boldsymbol{\psi}(s) = [\psi_1(s), \ldots, \psi_N(s)]$:

$$p(s) \propto \exp\big(\theta^T \boldsymbol{\psi}(s)\big) \tag{4.11}$$

DDCs correspond to the *mean parameters* or expected sufficient statistics of the distribution $\boldsymbol{\mu} = \mathbb{E}_{p(s)}[\boldsymbol{\psi}(s)]$—a parametrisation equivalent to the commonly used natural parameters θ (Wainwright and Jordan, 2008). Computing the corresponding natural parameters θ (and thus the density function $p(s)$) from the mean parameters is often analytically intractable. However, from the brain's perspective, recovering the density function $p(s)$ is rarely necessary. Most computations involving uncertainty amount to computing expectations with respect to the represented distribution and can therefore be approximated as a linear readout from the DDC representation (see eq. 2.4).

Figure 4.3 illustrates the effects of uncertainty on the DDC population response representing Gaussian distributions with different variances, assuming Gaussian encoding functions. In this setting, DDCs predict that population activity (and tuning curves) get wider with higher uncertainty. This is broadly consistent with some neural data, like the widening of visual heading tuning

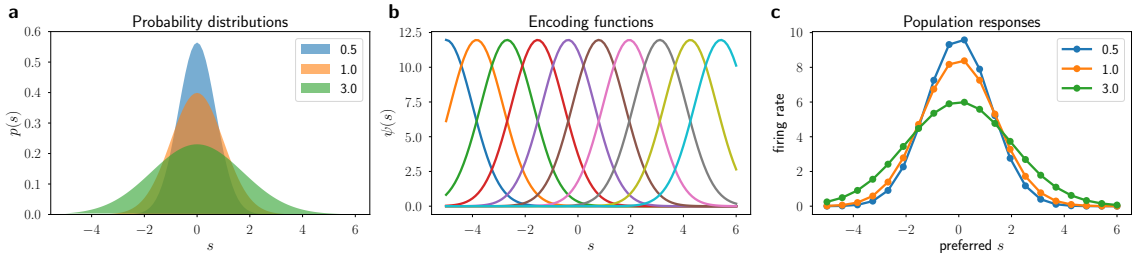


Figure 4.3: Distributed distributional codes. (a) Gaussian densities with different variances. (b) Gaussian encoding functions ($\psi_i(s)$) with different preferred values of s. (c) DDC population firing rates with neurons sorted by their preferred s. Wider population activity corresponds to distributions with higher variance.

curves observed in macaque MSTd at lower motion coherence (Morgan et al., 2008).

3.3 Probabilistic population codes

Ma et al. (2006) introduced probabilistic population codes (PPC) starting from the ubiquitous observation that neurons in sensory areas show trial-to-trial variability in their responses when repeatedly presented with the same stimulus (Tolhurst et al., 1983). Due to these variable responses, there are multiple stimulus values consistent with any given observed neural activity. Ma et al. argued that this variability implicitly represents uncertainty about the value of the stimulus. In contrast to earlier proposals of how populations of neurons might encode probability distributions (Anderson and Essen, 1994; Zemel et al., 1998), in the case of PPCs, neural variability was presented as an essential part of the code for uncertainty. In this section, we describe PPCs as presented by Ma et al., followed by a discussion of some conceptual limitations of the original framework as well as a closely related proposal that resolves those shortcomings.

Formally, the variability in the population response given a stimulus s can be described by a conditional probability distribution $p(r|s)$. When viewed as a function of the stimulus, this distribution is equivalent to the likelihood function $\mathcal{L}(s|r) = p(r|s)$ which expresses how likely a stimulus value is given the observed neural response. In a *probabilistic population code* neural activity

r encodes uncertainty about the stimulus s through this likelihood function, exploiting the relationship given by Bayes' rule:

$$p(s|r) \propto p(r|s)p(s) \tag{4.12}$$

That is, PPCs are defined using a decoding perspective: if the form of the likelihood $p(r|s)$ and the prior distribution $p(s)$ are known (or a uniform prior is assumed), we can read out the posterior distribution over the presented stimulus from the population activity r.

Spike count variability is often described with a Poisson-like distribution; for independent Poisson neural variability the likelihood function $p(r|s)$ takes the following form:

$$p(r|s) = \prod_i \frac{1}{r_i!} \exp(-f_i(s)) f_i(s)^{r_i} \tag{4.13}$$

where $f_i(s)$ is the tuning curve of neuron i, determining the mean spike count.

Assuming a flat prior $p(s) \propto 1$, the posterior distribution over the encoded stimulus s given the neural response r is proportional to the likelihood:

$$p(s|r) \propto \exp\big(\sum r_i \log f_i(s) - \sum_i f_i(s)\big) \tag{4.14}$$

$$\log p(s|r) = \sum_i r_i \log f_i(s) + \text{const.} \tag{4.15}$$

One of the key features of PPCs is that the log-posterior over the stimulus s is linear in the neural activity r (see eq. 4.15). Log-linear codes like PPC enable implementation of computations that require multiplying distributions in a particularly simple form. A common example of such a computation is cue combination, i.e. when sensory evidence about the stimulus coming from different sources needs to be combined. As in the classic experiment from Ernst and Banks (2002), the size of an object could be inferred from the available haptic and visual information. In this case, Bayesian cue combination

involves computing the posterior belief about the size of the object given the bimodal sensory evidence from beliefs constructed based on the visual or haptic information alone.

Assuming that sensory variability for a given object size is independent in the two modalities, i.e. $p(r^v, r^h|s) = p(r^v|s)p(r^h|s)$, the posterior given both haptic and visual evidence can be constructed as follows:

$$p(s|r^v, r^h) \propto p(r^v|s)p(r^h|s)p(s) \tag{4.16}$$

If the visual and haptic likelihood functions, $p(r_v|s)$ and $p(r_h|s)$ are represented in two corresponding populations in the form of a PPC—with two sets of tuning curves $\{f_i(s)\}$ identical up to proportionality—the bimodal posterior distribution can be represented as a PPC as well, by simply summing the population codes: $r_{vh} = r_v + r_h$. Following from eq. 4.16:

$$p(s|r^v, r^h) \propto \exp\big(\sum_i r_i^v \log(g_v f_i(s)) + r_i^h \log(g_h f_i(s))\big)p(s) \tag{4.17}$$

$$\propto \exp\big(\sum_i (r_i^v + r_i^h) \log f_i(s)\big)p(s) \tag{4.18}$$

$$\propto p(r^{vh}|s)p(s) \tag{4.19}$$

where g_v, g_h are the gain parameters of the corresponding visual and haptic tuning curves. For Gaussian shaped tuning curves $f_i(s)$ the likelihood function represented by a PPC is also an unnormalized Gaussian as a function of s: $p(r|s) \propto \mathcal{N}(s|\mu, \sigma^2)$, with mean and variance determined by the population activity r:

$$\mu = \frac{\sum_i r_i s_i}{\sum_i r_i}, \qquad \sigma^2 = \frac{\sigma_f^2}{\sum_i r_i} \tag{4.20}$$

where s_i and σ_f^2 are the mean and variance of the tuning curves $f_i(s)$. If the tuning curves are scaled by a scalar *gain* parameter g, scaling the population activity r accordingly; the width of the encoded likelihood function will be

inversely proportional to this gain: $g \propto 1/\sigma^2$ (see fig. 4.4 c,e).

The assumption of identical tuning in the two populations can be relaxed if the tuning curves can be linearly constructed from a common basis set. In that case, the bimodal PPC can be computed as weighted combination of the unimodal representations instead of simple summation (Ma et al., 2006).

PPCs retain this summation property if generalised to certain types of non-Poisson variability, where the log-likelihood $\log p(r|s)$ is linear in the neural response:

$$p(r|s, g) = \phi(r, g) \exp\big(r^T h(s)\big) \tag{4.21}$$

i.e. the likelihood is of an exponential family with linear sufficient statistics on r with gain dependent base measure $\phi(r, g)$.

3.4 Natural parameter codes

PPCs, as presented by Ma et al. (2006), are inspired by the observation that neural responses are variable given repeated presentation of the same stimulus and this variability inevitably introduces uncertainty about the value of the stimulus. This perspective is equivalent to a noisy encoding of a *single value* of the stimulus which then can only be decoded with some uncertainty, rather than directly encoding a *distributional* quantity as discussed by Zemel et al. (1998). Thus, PPCs are designed to capture uncertainty introduced by variability within the area encoding the stimulus itself (or in upstream areas) and do not allow for encoding uncertainty arising from partial observability or ambiguity.

While uncertainty is often induced at least partly by neural variability and in those cases it is naturally reflected in the likelihood $p(r|s)$, in general, the *neural code* for uncertainty need not involve variability at all. In particular, a log–linear encoding scheme that does not rely on the variability in the population activity to represent uncertainty can be defined. Assuming that the class of posterior distributions $q(s|x)$ needed to be represented are in the exponential family with sufficient statistic $T(s)$, the neural activity (firing rates

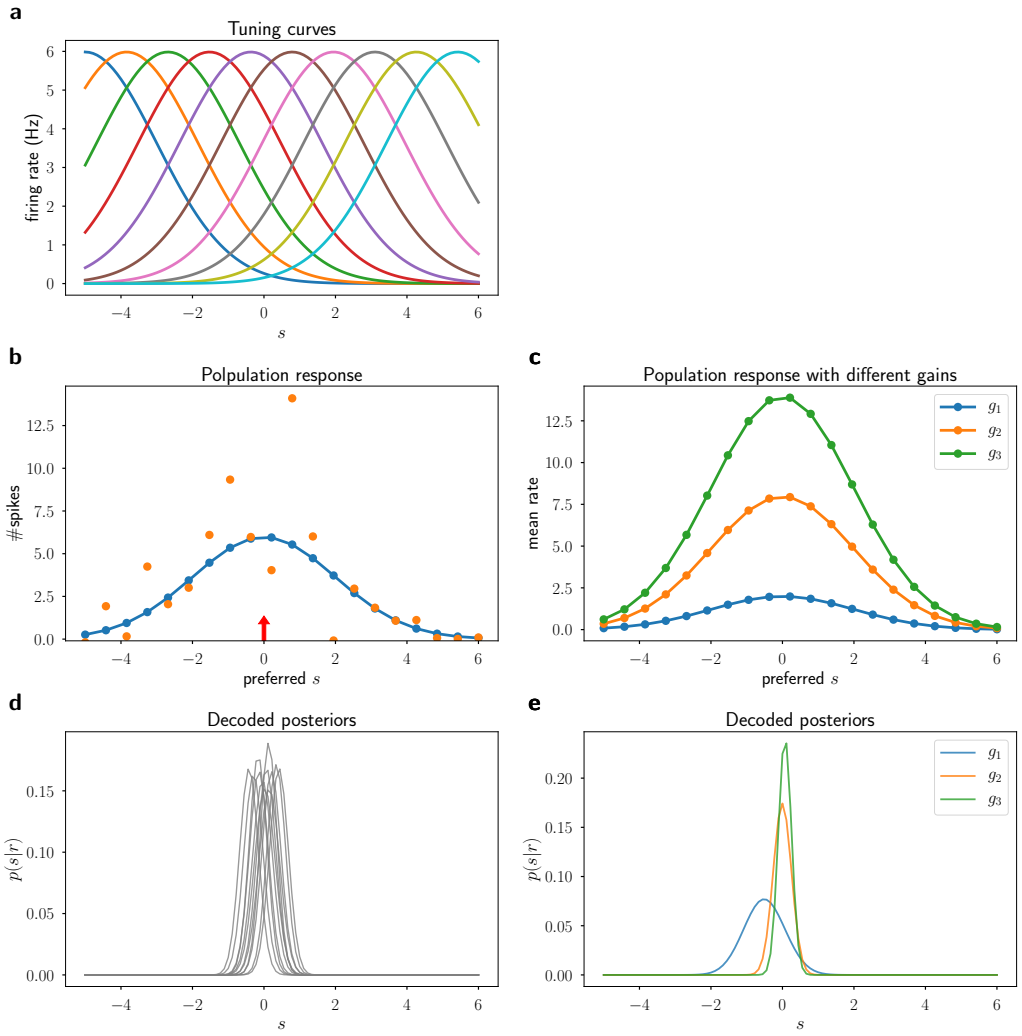


Figure 4.4: Probabilistic population codes. (a) Tuning curves describing the mean firing rate of individual neurons in the population. (b) Population response to a stimulus $s = 0$ with neurons sorted by their preferred stimulus, spike counts (orange) and mean rate (blue). (c) Mean population responses with three different gains. (d) Decoded posteriors $p(s|r)$ on single trials. (e) Decoded posteriors from single trial responses with gains as in (c).

r) can be used to set the natural parameters of this distribution:

$$q(s|x) \propto \exp\big(r(x)^T T(s)\big) \tag{4.22}$$

where the $r(x)$ reflects the dependence of the natural parameters r on the observations due to conditioning. We will refer to this encoding scheme as *natural parameter codes.* The key distinction between PPCs and natural parameter codes is that neural activity r encodes the distribution $q(s|x)$ using the parameters of the density function and therefore depends directly on the observed variable x, rather than defined implicitly through $p(r|s)$. However, log-linear codes share the property of PPCs that cue combination can be implemented by summing the population activities.

Importantly, deterministic codes for uncertainty like the above (eq. 4.22) are also expected to show trial-to-trial variability, as sensory noise (e.g. transduction noise in the retina) will lead to variability in the transduced sensory observations (x). Hence, the distribution describing this variability $p(r|s)$ can be computed for any deterministic encoding scheme: $r = \rho(x)$, with a function ρ that maps from the observations to some parametric representation of the posterior belief $q(s|x)$:

$$p(r|s) = \int dx \delta(r - \rho(x)) p(x|s) \tag{4.23}$$

where $p(x|s)$ is the likelihood in the true generative model and $\delta(.)$ is the Dirac delta function.

Thus, the structure of the apparent trial-to-trial variability measured when we condition on an external stimulus feature s depends on a number of factors: the encoding scheme and the encoded belief $q(s|x)$ through $\rho(x)$ and the form of the generative likelihood function $p(x|s)$.

In a special case of deterministic log-linear codes (eq. 4.22), the distribution of neural responses $p(r|s)$ computed based on eq. 4.23 is log-linear in r—just like in the case of PPCs—when the prior $p(s)$ is flat and the represented belief $q(s|x)$

is in the same exponential family as the true posterior $p(s|x)$ (see appx. D.1):

$$p(r|s) \propto \exp\big(r^T T(s)\big) \tag{4.24}$$

In general, however, this need not be the case, for example when the represented belief is from a different (typically more restricted) exponential family than the true posterior distribution. (This means that the distribution describing trial-to-trial neural variability (when viewed as a function of the latent variable) does not necessarily correspond to the represented posterior belief.)

A related point that is worth discussing here is about the relationship between how distributions might be *encoded* in a population of neurons versus how such distributions can be *decoded*. Equation 4.23 implies that for an arbitrary deterministic encoding scheme $r = \rho(x)$ we can construct a Bayesian decoder $p(s|r)$ and the decoded distribution—under mild conditions on $\rho(x)$—will coincide with the posterior $p(s|x)$ implied by the generative model. For instance, if the encoding function is the identity: $r = \rho(x) = x$, then the statement is trivially true as $p(s|r) = p(s|x)$. In general, decoding $p(s|r)$ from the population activity r will result in the posterior $p(s|x)$ as long as $r = \rho(x)$ preserves the statistics of x that are sufficient to construct the posterior $p(s|x)$.

Log-linear encoding schemes imply that the decoder $p(s|r)$ will be log-linear in r, however, the converse is not necessarily true: the fact that the posterior can be decoded by a log-linear decoder does not imply that the encoding scheme was a log-linear one. Figure 4.5 illustrates this on an example when the population activity encodes posterior uncertainty using DDCs: $r_i(x) = \mathbb{E}_{s|x}[\psi_i(s)]$. Neural responses in this case would show trial-to-trial variability when conditioned on a given value of the stimulus (fig. 4.5a), and the encoded posterior $p(s|x)$ can be decoded from the population activity r using a log-linear decoder (fig. 4.5b).

As this toy example illustrates, the decoding perspective adopted by PPCs (e.g. in Walker et al. (2019)) can not address the question of what encoding scheme is used in the population, it can merely be used to demonstrate that

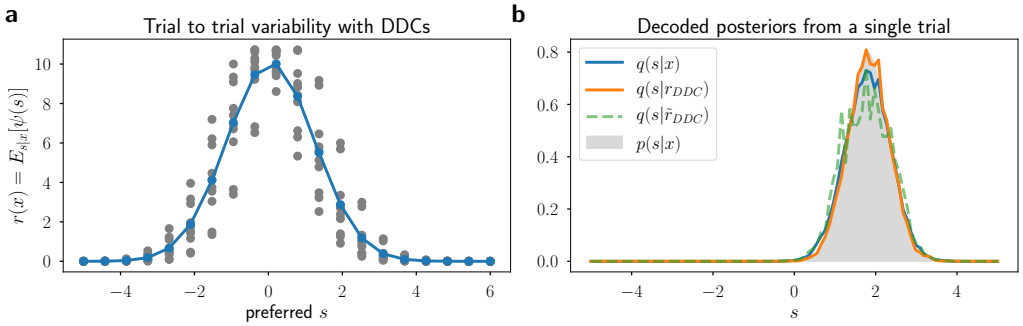


Figure 4.5: Bayesian decoding from DDCs. (a) Trial-to-trial variability of a population encoding posterior uncertainty by a DDC, neurons sorted by their preferred value of s. On each trial the observations x vary according to the generative process $p(x|s)$, resulting in a variability of encoded posteriors $r(x) = \mathbb{E}_{s|x}[\psi(s)]$. (b) Bayesian decoding of s on a single trial from, the observation x (blue), from the noiseless DDC representation (orange), from DDC representations embedded in Poisson noise (green).

information about uncertainty is present in the neural activity.

3.5 Neural sampling

The neural sampling hypothesis was first introduced by Hoyer and Hyvärinen (2003), suggesting that neural variability is directly related to the uncertainty represented in the population. The idea has been inspired by a family of approaches in Bayesian statistics called Monte Carlo sampling, originating from mathematical physics (Metropolis and Ulam, 1949). Monte Carlo methods are designed to represent and compute with intractable densities using *samples* from the distribution rather than an explicit parametric form.

Analogously, Hoyer and Hyvärinen proposed that neural firing rates might represent samples from posterior distributions over latent variables of interest, making a direct link between neural variability and uncertainty. The authors suggested two (not mutually exclusive) possibilities for how sampling might be implemented in the brain. Uncertainty over a continuous variable could be reflected in the firing rate variability of a single neuron over time, where the instantaneous rate at certain points in time represents a sample from the distribution. Alternatively, variability across different neurons in the population can instantaneously represent uncertainty. While this approach does not require

integrating neural activity thorough time, it assumes a degree of redundancy, multiple neurons representing samples of the same variable in parallel.

The sampling hypothesis provides a possible account for trial-to-trial neural variability widely observed in the cortex (Dean, 1981; Tolhurst et al., 1983). It also makes the prediction that firing rate variability will grow with uncertainty about the represented variables.

Hoyer and Hyvärinen showed that posterior sampling in a probabilistic independent component analysis model (ICA; Bell and Sejnowski (1997); Olshausen and Field (1997))—often used to model V1 responses—can reproduce Poisson-like variability. They also argued that neural sampling can explain the phenomenon of visual competition observed when subjects view an image with multiple likely interpretations, such as the Necker cube or Rubin's vase/face figure. The reported percept alternates between the different interpretations and neural correlates of the change in percept have been found in higher visual areas (Logothetis and Schall, 1989; Blake and Logothetis, 2002). In this setting, bistable percepts correspond to a bimodal posterior distribution and neural activity will sample regions around the modes, occasionally switching between different modes of the distribution.

The idea of neural sampling was taken further by Berkes et al. (2011), who proposed that spontaneous activity in visual cortex is consistent with sampling from the prior corresponding to natural statistics, while evoked activity can be interpreted as samples from a posterior distribution in a generative model of natural images. They reported that during development the statistics of spontaneous activity in ferret V1 showed increased similarity to responses evoked by naturalistic visual stimuli. These observations are consistent with Bayesian theories of perception where we expect that the brain's internal model of its sensory environment (captured as a prior distribution) gradually adapts to natural statistics.

Predictions of the sampling representation has been analysed by Orbán et al. (2016) in the context of performing inference in a Gaussian scale mixture

model (GSM; Wainwright and Simoncelli 2000). They have shown that sampling from the posterior distribution over the latent (feature activation) variables in a GSM model can account for a number of experimental observations in V1 electrophysiology. The parameters of the model were set (or learned) to match the distributions of natural image patches (van Hateren and van der Schaaf, 1998) and the samples of the latent variables were interpreted as the membrane potential of individual neurons. The resulting model could reproduce phenomena such as the drop in membrane potential variability with stimulus onset, stimulus dependent membrane potential variability or the relationship between spontaneous, signal and noise correlations. Furthermore, Orbán et al. have replicated findings of earlier related models including non-classical receptive field effects and contrast invariant orientation tuning (Schwartz and Simoncelli, 2001; Anderson et al., 2000; Finn et al., 2007).

The contrast invariant property of tuning curves widths in primary visual cortex has been used to argue in favour of PPCs or natural parameter codes. Change in contrast is often used as a proxy for change in uncertainty, and it is reflected in log-linear codes as a multiplicative scaling of neural activities, thus producing contrast invariant tuning.

Contrast invariant tuning also emerges in the GSM model: the analysis of Orbán et al. shows that contrast invariance can be derived from properties of the generative model and is likely not specific to the neural representation of uncertainty as previously argued (Pouget et al., 2013).

A number of theoretical works have addressed technical challenges related to the neural circuit implementation of sampling. Hennequin et al. (2014); Aitchison and Lengyel (2016) studied how sampling can be efficiently implemented in recurrently connected excitatory-inhibitory networks. Buesing et al. (2011) has shown how sampling from binary variables can be implemented in a network of spiking neurons.

Kutschireiter et al. (2017) described how sampling could be potentially implemented by neural dynamics in a simple dynamic model by an approach

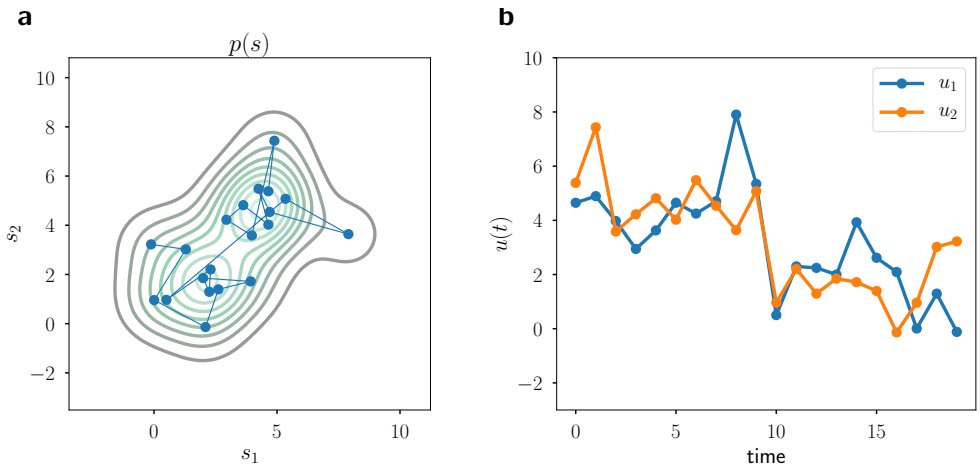


Figure 4.6: Sampling representation. (a) A 2-dimensional example of a bimodal distribution represented non-parametrically, as a set of samples (in blue). (b) According to the proposal of Orbán et al. (2016), the sampling process is reflected in the temporal fluctuations of the membrane potentials u_1 and u_2, coding for variables s_1 and s_2 of the distribution shown in (a).

derived from particle filtering.

4 Discussion of different proposals

There is a number of competing theories about how neural systems represent uncertainty information, however, experimental validation (or rather falsification) has remained difficult. One of the challenges in studying representations of uncertainty experimentally is the lack of single trial characterisation of neural activity. Trial-averaged data can often not distinguish between different proposals. For example, if neurons responded according to a tuning curve with a given width independently of the degree of uncertainty, the resulting trial-averaged tuning would still appear broader due to the variable (neural) point estimates of the stimulus value on each trial.

Neuronal variability plays different roles in the representations we have reviewed in the previous section. In the sampling representation variability codes directly for uncertainty, while in the case of PPCs, it is coupled to the mean activity and therefore conveys the same information. Deterministic codes

like KDEs, DDCs [2] or natural parameter codes are defined based on mean activity only, and don't prescribe any specific role or form of neuronal variability. Nevertheless, we do expect to see variability even if we assume that the brain is using a deterministic coding scheme. The representation of the posterior belief about some latent variable is dependent on the sensory observation received by the brain. Therefore, it is likely that variability in the sensory observations will induce variability in the representation as well. Moreover, when uncertainty captured by the posterior belief is due to variability in the generative model or sensory noise, the degree of variability in the representation will grow with uncertainty.

We can think of both PPCs and DDCs as representing exponential family distributions (by their natural or mean parameters, respectively), however, the relationship between the neuronal tuning curves and the encoded distributions is quite different. In the case of PPCs, Gaussian tuning curves would imply Gaussian distributions, while DDCs with Gaussian encoding functions can represent a broader family including multimodal distributions.

An aspect of probabilistic reasoning that has received little attention in the literature so far is how representations of uncertainty could be learned by neural systems. As we discussed in section 2 of this chapter, we need to consider how an internal model of the latent variables in the world is acquired and how a corresponding recognition model can be learned. Many of the existing works so far are limited by relying on supervised training (Beck et al., 2011; Orhan and Ma, 2017) or are restricted to models where learning and inference is analytically tractable (Makin et al., 2013, 2015; Raju and Pitkow, 2016). Learning with PPCs or natural parameter codes is non-trivial, as it is unclear how to handle distributions which are not from a tractable family (e.g. Gaussian), and therefore computing normalising constants is difficult.

While sampling is in principle well suited for dealing with intractable distributions, learning has been only explored in fairly limited settings with linear

[2]The original formulation of of DPCs and DDPCs did include Poisson variability but had no functional role in the representation and was treated as nuisance.

observation models and in the limit of small observation noise (Kutschireiter et al., 2017).

The DDC Helmholtz machine presented in chapter 2 provides a proof of concept that the DDC representation can support accurate learning in hierarchical generative models. The construction of the algorithm allows for simple biologically plausible learning rules that don't require propagating gradients across the whole hierarchy.

Finally, here we have focused on uncertainty induced by ambiguity in the generative process itself. This type of uncertainty, also called *aleatoric* uncertainty, is irreducible even when perfect knowledge of the generative process is assumed. *Epistemic* or model uncertainty, that captures uncertainty due to limited learning experience, has received much less attention in the neuroscience literature (see Pouget et al. (2013) for a brief discussion), but is also likely to be important for biological organisms.

D Appendix

D.1 Log-linear codes for uncertainty

Consider a generative model $p(x, z) = p(x|z)p(z)$, with a flat prior $p(z)$ and exponential family likelihood $p(x|z)$:

$$p(x|z) = \exp\big(S(x)^T\theta(z) - A(\theta(z))\big) \tag{4.25}$$

$$p(z) \propto 1 \tag{4.26}$$

One can show that if the posterior belief $q(z|x)$ is represented as a log-linear code (sec. 3.4) with sufficient statistic $\theta(z)$, i.e.

$$q(z|x) \propto \exp\big(\theta(z)^T r\big), \quad r = \rho(x) \tag{4.27}$$

the distribution describing the conditional variability in the firing rates, $p(r|z)$, is log-linear in r, just as in the case of PPCs (sec. 3.3).

$$p(r|z) = \int p(r|x)p(x|z)ds \tag{4.28}$$

$$= \int \delta(r - \rho(x))p(x|z)ds \tag{4.29}$$

$$= p(x|z)|_{x=\rho^{-1}(r)} \tag{4.30}$$

$$= \exp\big(S(\rho^{-1}(r))^T\theta(z) - A(\theta(z))\big) \tag{4.31}$$

If the log-normaliser $A(\theta(z))$ in eq. 4.25 does not depend on z (i.e. $A(\theta(z)) = A(\tilde{\theta})$) the natural parameter of the posterior $q(z|x)$ becomes: $\rho(x) = S(x)$. Thus, by further assuming that $S(x)$ is invertible we get:

$$p(r|z) \propto \exp\big(r^T\theta(z)\big)\big) \tag{4.32}$$

That is, in a special case we recover Poisson-like variability of neural responses.

Chapter 5

Distributed Distributional Codes in Reinforcement Learning

Animals need to devise strategies to maximise returns while interacting with their environment based on incoming noisy sensory observations. Task-relevant states, such as the agent's location within an environment or the presence of a predator, are often not directly observable but must be inferred using available sensory information. Successor representations (SR) have been proposed as a middle-ground between model-based and model-free reinforcement learning strategies, allowing for fast value computation and rapid adaptation to changes in the reward function or goal locations. Indeed, recent studies suggest that features of neural responses are consistent with the SR framework. However, it is not clear how such representations might be learned and computed in partially observed, noisy environments. Here, we introduce a neurally plausible model using *distributional successor features*, which builds on the distributed distributional code for the representation and computation of uncertainty, and which allows for efficient value function computation in partially observed environments via the successor representation. We show that distributional successor features can support reinforcement learning in noisy environments in which direct learning of successful policies is infeasible.

1 Introduction

Humans and other animals are able to evaluate long-term consequences of their actions and adapt their behaviour to maximize reward across different environments. This behavioural flexibility is often thought to result from the interaction of two adaptive systems implementing model-based and model-free reinforcement learning (RL).

Model-based learning allows for flexible goal-directed behaviour, acquiring an internal model of the environment which is used to evaluate the consequences of actions. As a result, an agent can rapidly adjust its policy to localized changes in the environment or in reward function. But this flexibility comes at a high computational cost, as optimal actions and value functions depend on expensive simulations in the model. Model-free methods, on the other hand, learn cached values for states and actions, enabling rapid action selection. However, this approach is particularly slow to adapt to changes in the task, as adjusting behaviour even to localized changes, e.g. in the placement of the reward, requires updating cached values at all states in the environment. It has been suggested that the brain makes use both of these complementary approaches, and that they may compete for behavioural control (Daw et al., 2005); indeed, several behavioural studies suggest that subjects implement a hybrid of model-free and model-based strategies (Daw et al., 2011; Gläscher et al., 2010).

Successor representations (SR; Dayan, 1993) augment the internal state used by model-free systems by the expected future occupancy of each world state. SRs can be viewed as a *precompiled* representation of the model under a given policy. Thus, learning based on SRs falls between model-free and model-based approaches and correspondingly can reproduce a range of behaviours (Russek et al., 2017). Recent studies have argued for evidence consistent with SRs in rodent hippocampal and human behavioural data (Stachenfeld et al., 2017; Momennejad et al., 2017).

Motivated by both theoretical and experimental work arguing that neural RL systems operate over latent states and need to handle state uncertainty

(Dayan and Daw, 2008; Gershman, 2018; Starkweather et al., 2017), our work takes the successor framework further by considering partially observable environments. Adopting the framework of distributed distributional coding (Vértes and Sahani, 2018), we show how learnt latent dynamical models of the environment can be naturally integrated with SRs defined over the latent space. We begin with short overviews of reinforcement learning, the formalism of the partially observed setting (section 2) and the successor representation (section 3). In section 4, we describe how using DDCs in the generative and recognition models leads to a particularly simple algorithm for learning latent state dynamics and the associated SR.

2 Reinforcement learning – preliminaries

Generally speaking, the objective of a reinforcement learning agent is to learn to interact with its environment such that it maximises the long-run average reward it receives. A key component of most RL algorithms is that of computing the value function, defined as the expectation of the cumulative discounted future rewards under a given policy π:

$$V^{\pi}(s_t) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k}] \tag{5.1}$$

where γ is the discount factor determining the time horizon of the agent, and r_t is the (scalar) reward received at time t. Estimating the value function for an arbitrary policy is typically non-trivial, and is the focus of several existing algorithms ranging from model-free to model-based methods.

2.1 Temporal-difference learning

Temporal-difference learning (TD; Sutton, 1988) is a model-free approach for policy evaluation; that is, estimating the value function $V^{\pi}(s)$ while the agent follows a given policy π. TD learning relies on the recursive property of the

value function expressed by the Bellman equation:

$$V^{\pi}(s) = \mathbb{E}_{\pi(a|s)}[\mathbb{E}_{p(s',r|s,a)}[r + \gamma V^{\pi}(s')]] \tag{5.2}$$

By rearranging eq. 5.2, we can arrive at the following sample-based update rule which depends on the TD error (δ):

$$\delta = r + \gamma V^{\pi}(s') - V^{\pi}(s) \tag{5.3}$$

$$V^{\pi}(s) \leftarrow V(s)^{\pi} + \eta\delta \tag{5.4}$$

where r and s' are the reward and state observed after choosing an action according to the policy $\pi(a|s)$ and η is a learning rate.

TD learning is a blend of dynamic programming and Monte Carlo methods, as it uses the current estimate of the value function at s' as a target for updating the value for state s, together with replacing the expectations in eq. 5.2 by a single sample. Note that the TD update can be computed without explicit knowledge of the state transition and reward model $p(s', r|s, a)$ as the sampled values of s' and r are available through experience.

2.2 Partially observable Markov decision processes

Markov decision processes (MDP) provide a framework for modelling a wide range of sequential decision-making tasks relevant for reinforcement learning. An MDP is defined by a set of states $\mathcal{S}$ and actions $\mathcal{A}$, a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and a probability distribution $\mathcal{T}(s'|s, a)$ that describes the Markovian dynamics of the states conditioned on actions of the agent. For notational convenience we will take the reward function to be independent of action, depending only on state; but the approach we describe is easily extended to the more general case. A partially observable Markov decision process (POMDP) is a generalization of an MDP where the Markovian states $s \in \mathcal{S}$ are not directly observable to the agent. Instead, the agent receives observations ($o \in O$) that depend on the current *latent* state via an observation

process $\mathcal{Z}(o|s)$. Formally, a POMDP is a tuple: $(\mathcal{S},\ \mathcal{A},\ \mathcal{T},\ R,\ O,\ \mathcal{Z},\ \gamma)$, comprising the objects defined above and a discount factor γ. POMDPs can be defined over either discrete or continuous state spaces. Here, we focus on the more general continuous case, although the model we present is applicable to discrete state spaces as well.

3 The successor representation

As an agent explores an environment, the states it visits are ordered by the agent's policy and the transition structure of the world. State representations that respect this dynamic ordering are likely to be more efficient for value estimation and may promote more effective generalization. This may not be true of the observed state coordinates. For instance, a barrier in a spatial environment might mean that two states with adjacent physical coordinates are associated with very different values.

Dayan (1993) argued that a natural state space for model-free value estimation is one where distances between states reflect the similarity of future paths given the agent's policy. The successor representation (Dayan, 1993; SR) for state s_i is defined as the expected discounted sum of future occupancies for each state s_j, given the current state s_i:

$$M^{\pi}(s_i, s_j) = \mathbb{E}_{\pi}\Big[\sum_{k=0}^{\infty}\gamma^k \mathbb{I}[s_{t+k} = s_j] \mid s_t = s_i\Big]. \tag{5.5}$$

That is, in a discrete state space, the SR is a $N \times N$ matrix where N is the number of states in the environment. The SR depends on the current policy π through the expectation in the right hand side of eq. 5.5, taken with respect to a (possibly stochastic) policy $p^{\pi}(a_t|s_t)$ and environmental transitions $\mathcal{T}(s_{t+1}|s_t, a_t)$. The SR makes it possible to express the value function in a particularly simple form. Following from eq. 5.5 and the usual definition of the value function:

$$V^{\pi}(s_i) = \sum_j M^{\pi}(s_i, s_j)R(s_j)\,, \tag{5.6}$$

where $R(s_j)$ is the immediate reward in state s_j.

The successor matrix M^π can be learned by temporal difference (TD) learning (Sutton, 1988), in much the same way as TD is used to update value functions. In particular, the SR is updated according to a TD error:

$$\delta_t(s_j) = \mathbb{I}[s_t = s_j] + \gamma M^\pi(s_{t+1}, s_j) - M^\pi(s_t, s_j)\,, \tag{5.7}$$

which reflects errors in *state predictions* rather than rewards, a learning signal typically associated with model based RL.

As shown in eq. 5.6, the value function can be factorized into the SR—i.e., information about expected future states under the policy—and instantaneous reward in each state[1]. This modularity enables rapid policy evaluation under changing reward conditions: for a fixed policy only the reward function needs to be relearned to evaluate $V^\pi(s)$. This contrasts with both model-free and model-based algorithms, which require extensive experience or rely on computationally expensive evaluation, respectively, to recompute the value function.

3.1 Successor representation using features

The successor representation can be generalized to continuous states $s \in \mathcal{S}$ by using a set of feature functions $\{\psi_i(s)\}$ defined over $\mathcal{S}$. In this setting, the successor representation (also referred to as the successor feature representation or SF) encodes expected feature values instead of occupancies of individual states:

$$M^\pi(s_t, i) = \mathbb{E}_\pi\Big[\sum_{k=0}^{\infty} \gamma^k \psi_i(s_{t+k}) \mid s_t\Big] \tag{5.8}$$

Assuming that the reward function can be written (or approximated) as a linear function of the features: $R(s) = \boldsymbol{w}_{\text{rew}}^T \boldsymbol{\psi}(s)$ (where the feature values are collected into a vector $\boldsymbol{\psi}(s)$), the value function $V(s_t)$ has a simple form

[1] Alternatively, for the more general case of action-dependent reward, the expected instantaneous reward under the policy-dependent action in each state.

analagous to the discrete case:

$$V^{\pi}(s_t) = \boldsymbol{w}_{\text{rew}}^T M^{\pi}(s_t) \tag{5.9}$$

For consistency, we can use linear function approximation with the same set of features as in eq. 5.8 to parametrize the successor features $M^{\pi}(s_t, i)$.

$$M^{\pi}(s_t, i) \approx \sum_j U_{ij} \psi_j(s_t) \tag{5.10}$$

The form of the SFs, embodied by the weights U_{ij}, can be found by temporal difference learning:

$$\delta_i = \psi_i(s_t) + \gamma M(s_{t+1}, i) - M(s_t, i) \tag{5.11}$$

$$U_{ij} \leftarrow U_{ij} + \eta \delta_i \psi_j(s_t) \tag{5.12}$$

As we have seen in the discrete case, the TD error here signals prediction errors about features of state, rather than about reward.

4 Distributional successor representation

As discussed above, the successor representation can support efficient value computation by incorporating information about the policy and the environment into the state representation. However, in more realistic settings, the states themselves are not directly observable and the agent is limited to state-dependent noisy sensory information.

In this section, we lay out how the DDC representation for uncertainty allows for learning and computing with successor representations defined over latent variables. First, we describe an algorithm for learning and inference in dynamical latent variable models using DDCs. We then establish a link between the DDC and successor features (eq. 5.8) and show how they can be combined to learn what we call the *distributional successor features*. We discuss different algorithmic and implementation-related choices for the proposed scheme and

their implications.

4.1 Learning and inference in a state space model using DDCs

Here, we consider POMDPs where the state-space transition model is itself defined by a conditional DDC with means that depend linearly on the preceding state features. That is, the conditional distribution describing the latent dynamics implied by following the policy π can be written in the following form:

$$p^{\pi}(s_{t+1}|s_t) \Leftrightarrow \mathbb{E}_{s_{t+1}|s_t,\pi}[\boldsymbol{\psi}(s_{t+1})] = T^{\pi}\boldsymbol{\psi}(s_t) \tag{5.13}$$

where T^{π} is a matrix parametrizing the functional relationship between s_t and the expectation of $\boldsymbol{\psi}(s_{t+1})$ with respect to $p^{\pi}(s_{t+1}|s_t)$.

The agent has access only to sensory observations o_t at each time step, and in order to be able to make use of the underlying latent structure, it has to learn the parameters of generative model $p(s_{t+1}|s_t)$, $p(o_t|s_t)$ as well as learn to perform inference in that model.

We consider online inference (filtering), i.e. at each time step t the recognition model produces an estimate $q(s_t|\mathcal{O}_t)$ of the posterior distribution $p(s_t|\mathcal{O}_t)$ given all observations up to time t: $\mathcal{O}_t = (o_1, o_2, \dots o_t)$. As in the DDC Helmholtz machine (Vértes and Sahani, 2018), these distributions are represented by a set of expectations—i.e., by a DDC:

$$\boldsymbol{\mu}_t(\mathcal{O}_t) = \mathbb{E}_{q(s_t|\mathcal{O}_t)}[\boldsymbol{\psi}(s_t)] \tag{5.14}$$

The filtering posterior $\boldsymbol{\mu}_t(\mathcal{O}_t)$ is computed iteratively, using the posterior in the previous time step $\boldsymbol{\mu}_{t-1}(\mathcal{O}_{t-1})$ and the new observation o_t. The Markovian structure of the state space model (see fig. 5.1) ensures that the recognition model can be written as a recursive function:

$$\boldsymbol{\mu}_t(\mathcal{O}_t) = f_W(\boldsymbol{\mu}_{t-1}(\mathcal{O}_{t-1}), o_t) \tag{5.15}$$

Algorithm 3 Wake-sleep algorithm in the DDC state-space model

Initialise T, W
while not converged **do**
  Sleep phase:
  sample: $\{s_t^{\text{sleep}}, o_t^{\text{sleep}}\}_{t=0\dots N} \sim p(\mathcal{S}_N, \mathcal{O}_N)$
  update W: $\Delta W \propto \sum_t \big(\boldsymbol{\psi}(s_t^{\text{sleep}}) - f_W(\boldsymbol{\mu}_{t-1}(\mathcal{O}_{t-1}^{\text{sleep}}), o_t^{\text{sleep}})\big) \nabla_W f_W$
  Wake phase:
  $\mathcal{O}_N \leftarrow \{\text{collect observations}\}$
  infer posterior DDC $\boldsymbol{\mu}_t(\mathcal{O}_t) = f_W(\boldsymbol{\mu}_{t-1}(\mathcal{O}_{t-1}), o_t)$
  update T: $\Delta T \propto (\boldsymbol{\mu}_{t+1}(\mathcal{O}_{t+1}) - T\boldsymbol{\mu}_t(\mathcal{O}_t))\boldsymbol{\mu}_t(\mathcal{O}_t)^T$
  update observation model parameters
end while

with a set of parameters W.

The recognition and generative models are updated using an adapted version of the wake-sleep algorithm (Hinton et al., 1995; Vértes and Sahani, 2018). In the following, we describe the two phases of the algorithm in more detail (see Algorithm 3).

Sleep phase

The aim of the sleep phase is to adjust the parameters of the recognition model given the current generative model. Specifically, the recognition model should approximate the expectation of the DDC encoding functions $\boldsymbol{\psi}(s_t)$ under the filtering posterior $p(s_t|\mathcal{O}_t)$. This can be achieved by moment matching, i.e., simulating a sequence of latent and observed states from the current model and minimizing the Euclidean distance between the output of the recognition model and the sufficient statistic vector $\boldsymbol{\psi}(.)$ evaluated at the latent state from the next time step.

$$W \leftarrow \underset{W}{\operatorname{argmin}} \sum_t \|\boldsymbol{\psi}(s_t^{\text{sleep}}) - f_W(\boldsymbol{\mu}_{t-1}(\mathcal{O}_{t-1}^{\text{sleep}}), o_t^{\text{sleep}})\|^2 \tag{5.16}$$

where $\{s_t^{\text{sleep}}, o_t^{\text{sleep}}\}_{t=0\dots N} \sim p(s_0)p(o_0|s_0) \prod_{t=0}^{N-1} p(s_{t+1}|s_t, T^\pi)p(o_{t+1}|s_{t+1})$.

This update rule can be implemented online as samples are simulated, and after a sufficiently long simulated sequence (or multiple sequences) $\{s_t^{\text{sleep}}, o_t^{\text{sleep}}\}_t$

the recognition model will learn to approximate expectations of the form: $f_W(\boldsymbol{\mu}_{t-1}(\mathcal{O}_{t-1}^{\text{sleep}}), o_t^{\text{sleep}}) \approx \mathbb{E}_{p(s_t|\mathcal{O}_t)}[\boldsymbol{\psi}(s_t)]$, yielding a DDC representation of the posterior.

Wake phase

In the wake phase, the parameters of the generative model are adapted such that it captures the sensory observations better. Here, we focus on learning the policy-dependent latent dynamics $p^{\pi}(s_{t+1}|s_t)$; the observation model can be learned by the approach of Vértes and Sahani (2018). Given a sequence of inferred posterior representations $\{\boldsymbol{\mu}_t(\mathcal{O}_t)\}$ computed using wake phase observations, the parameters of the latent dynamics T can be updated by minimizing a simple predictive cost function:

$$T \leftarrow \underset{T}{\operatorname{argmin}} \sum_t \|\boldsymbol{\mu}_{t+1}(\mathcal{O}_{t+1}) - T\boldsymbol{\mu}_t(\mathcal{O}_t)\|^2 \tag{5.17}$$

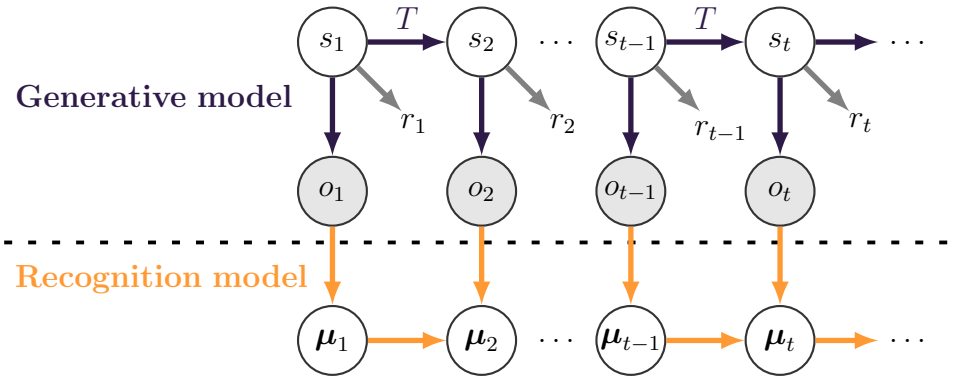
The intuition behind eq. 5.17 is that for the optimal generative model the latent dynamics satisfies the following equality: $T^*\boldsymbol{\mu}_t(\mathcal{O}_t) = \mathbb{E}_{p(o_{t+1}|\mathcal{O}_t)}[\boldsymbol{\mu}_{t+1}(\mathcal{O}_{t+1})]$. That is, the predictions made by combining the posterior at time t and the prior will agree with the average posterior at the next time step—making T^* a stationary point of the optimization in eq. 5.17. For further details on the nature of the approximation implied by the wake phase update and its relationship to variational learning, see the supplementary material. In practice, the update can be done online, using gradient steps analogous to prediction errors:

$$\Delta T \propto \big(\boldsymbol{\mu}_{t+1}(\mathcal{O}_{t+1}) - T\boldsymbol{\mu}_t(\mathcal{O}_t)\big)\boldsymbol{\mu}_t(\mathcal{O}_t)^T \tag{5.18}$$

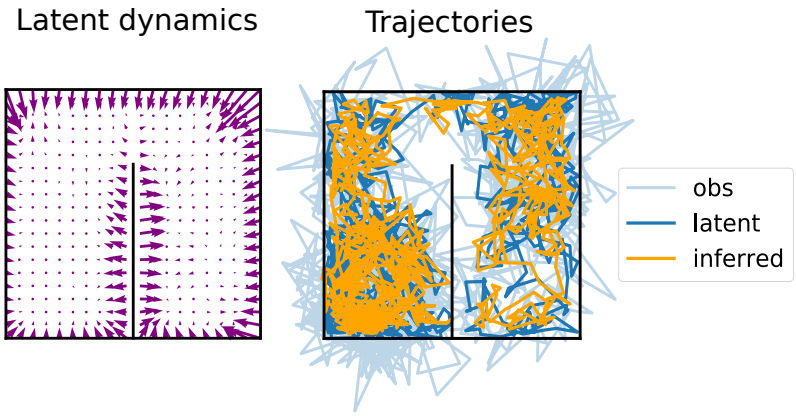
Figure 5.1 shows a state-space model corresponding to a random walk policy in the latent space with noisy observations, learned using DDCs (Algorithm 3). For further details of the experiment, see the supplementary material.

4.2 Learning distributional successor features

Next, we show how using a DDC to parametrize the generative model (eq. 5.13) makes it possible to compute the successor features defined in the latent space



(a) DDC state-space model



(b) Noisy 2D environment

Figure 5.1: Learning and inference in a state-space model parametrised by a DDC. (a) The structure of the generative and recognition models. (b) Visualization of the dynamics T learned by the wake-sleep (algorithm 3). Arrows show the conditional mean $\mathbb{E}_{s_{t+1}|s_t}[s_{t+1}]$ for each location. (c) Posterior mean trajectories inferred using the recognition model, plotted on top of true latent and observed trajectories.

in a tractable form, and how this computation can be combined with inference based on sensory observations.

Following the definition of the SFs (eq. 5.8), we have:

$$M(s_t) = \mathbb{E}_\pi\Big[\sum_{k=0}^{\infty} \gamma^k \boldsymbol{\psi}(s_{t+k}) | s_t\Big] \tag{5.19}$$

$$= \sum_{k=0}^{\infty} \gamma^k \mathbb{E}_\pi[\boldsymbol{\psi}(s_{t+k}) | s_t] \tag{5.20}$$

We can compute the conditional expectations of the feature vector ψ in

eq. 5.19 as a series of nested expectations:

$$\mathbb{E}_{s_{t+k}}[\psi(s_{t+k})|s_t] = \mathbb{E}_{s_{t+k-1}}[\mathbb{E}_{s_{t+k}}[\psi(s_{t+k})|s_{t+k-1}]|s_t] \tag{5.21}$$

$$= \mathbb{E}_{s_{t+k-1}}[T\psi(s_{t+k-1})|s_t] \tag{5.22}$$

$$= T^2\mathbb{E}_{s_{t+k-2}}[\psi(s_{t+k-2})|s_t] \tag{5.23}$$

$$\vdots$$

$$\mathbb{E}_{s_{t+k}}[\psi(s_{t+k})|s_t] = T^k\psi(s_t) \tag{5.24}$$

Thus, from equations 5.19 and 5.24 we have:

$$M(s_t) = \sum_{k=0}^{\infty} \gamma^k T^k \psi(s_t) \tag{5.25}$$

$$= (I - \gamma T)^{-1}\psi(s_t) \tag{5.26}$$

Equation 5.26 is reminiscent of the result for discrete observed state spaces $M(s_i, s_j) = (I - \gamma P)^{-1}_{ij}$ (Dayan, 1993), where P is a matrix containing Markovian transition probabilities between states. In a continuous state space, however, finding a closed form solution like eq. 5.26 is non-trivial, as it requires evaluating a set of typically intractable integrals. The solution presented here directly exploits the DDC parametrization of the generative model and the correspondence between the features used in the DDC and the SFs.

In this framework, we can not only compute the successor features in closed form in the latent space, but also evaluate the *distributional successor features*, the posterior expectation of the SFs given a sequence of sensory observations:

$$\mathbb{E}_{s_t|\mathcal{O}_t}[M(s_t)] = (I - \gamma T)^{-1}\mathbb{E}_{s_t|\mathcal{O}_t}[\boldsymbol{\psi}(s_t)] \tag{5.27}$$

$$= (I - \gamma T)^{-1}\boldsymbol{\mu}_t(\mathcal{O}_t) \tag{5.28}$$

The results from this section suggest a number of different ways the distributional successor features $\mathbb{E}_{s_t|\mathcal{O}_t}[M(s_t)]$ can be learned or computed.

4.2.1 Learning distributional SFs during *sleep phase*

The matrix $U = (I - \gamma T)^{-1}$ needed to compute distributional SFs in eq. 5.28 can be learned from temporal differences in feature predictions based on latent state sequences simulated in the *sleep phase* (see eq. 5.10-5.11). That is, given a sequence of latent states sampled from the dynamics model: $s_1, s_2, \ldots, s_T \sim \prod_t p(s_t|s_{t-1})p(s_0)$, U can be updated by TD learning:

$$\delta_i = \psi_i(s_t) + \gamma M^\pi(s_{t+1}, i) - M^\pi(s_t, i) \tag{5.29}$$

$$\Delta U_{ij} \propto \delta_i \psi_j(s_t) \tag{5.30}$$

Following a potential change in the dynamics of the environment, sleep phase learning allows for updating SFs and therefore cached values offline, without the need for further experience.

4.2.2 Computing distributional SFs by *dynamics*

Alternatively, eq. 5.28 can be implemented as a fixed point of a linear dynamical system, with recurrent connections reflecting the model of the latent dynamics:

$$\tau \dot{x} = -x + \gamma T x + \boldsymbol{\mu}_t(\mathcal{O}_t) \tag{5.31}$$

$$\Rightarrow x(\infty) = (I - \gamma T)^{-1} \boldsymbol{\mu}_t(\mathcal{O}_t) \tag{5.32}$$

In this case, there is no need to learn $(I - \gamma T)^{-1}$ explicitly but it is implicitly computed through dynamics. For this to work, there is an underlying assumption that the dynamical system in eq. 5.31 reaches equilibrium on a timescale (τ) faster than that on which the observations $\mathcal{O}_t$ evolve.

Both of these approaches avoid having to compute the matrix inverse directly and allow for evaluation of policies given by a corresponding dynamics matrix T^π offline.

4.2.3 Learning distributional SFs during *wake phase*

Instead of fully relying on the learned latent dynamics to compute the distributional SFs, we can use posteriors computed by the recognition model during the

wake phase, that is, using observed data. We can define the distributional SFs directly on the DDC posteriors: $\widetilde{M}(\mathcal{O}_t) = \mathbb{E}_\pi[\sum_k \gamma^k \boldsymbol{\mu}_{t+k}(\mathcal{O}_{t+k})|\boldsymbol{\mu}_t(\mathcal{O}_t)]$, treating the posterior representation $\boldsymbol{\mu}_t(\mathcal{O}_t)$ as a feature space over sequences of observations $\mathcal{O}_t = (o_1 \dots o_t)$. Analogously to section 3.1, $\widetilde{M}(\mathcal{O}_t)$ can be acquired by TD learning and assuming linear function approximation: $\widetilde{M}(\mathcal{O}_t) \approx U\boldsymbol{\mu}_t(\mathcal{O}_t)$. The matrix U can be updated online, while executing a given policy and continuously inferring latent state representations using the recognition model:

$$\delta_t = \boldsymbol{\mu}_t(\mathcal{O}_t) + \gamma M(\mathcal{O}_{t+1}) - M(\mathcal{O}_t) \tag{5.33}$$

$$\Delta U \propto \delta_t \boldsymbol{\mu}_t(\mathcal{O}_t)^T \tag{5.34}$$

It can be shown that $\widetilde{M}(\mathcal{O}_t)$, as defined here, is equivalent to $\mathbb{E}_{s_t|\mathcal{O}_t}[M(s_t)]$ if the learned generative model is optimal–assuming no model mismatch–and the recognition model correctly infers the corresponding posteriors $\boldsymbol{\mu}_t(\mathcal{O}_t)$ (see supplementary material). In general, however, exchanging the order of TD learning and inference leads to different SFs. The advantage of learning the distributional successor features in the wake phase is that even when the model does not perfectly capture the data (e.g. due to lack of flexibility or early on in learning) the learned SFs will reflect the structure in the observations through the posteriors $\boldsymbol{\mu}_t(\mathcal{O}_t)$.

4.3 Value computation in a noisy 2D environment

We illustrate the importance of being able to consistently handle uncertainty in the SFs by learning value functions in a noisy environment. We use a simple 2-dimensional box environment with continuous state space that includes an internal wall. The agent does not have direct access to its spatial coordinates, but receives observations corrupted by Gaussian noise. Figure 5.2 shows the value functions computed using the successor features learned in three different settings: assuming direct access to latent states, treating observations as though they were noise-free state measurements, and using latent state estimates inferred from observations. The value functions computed in the

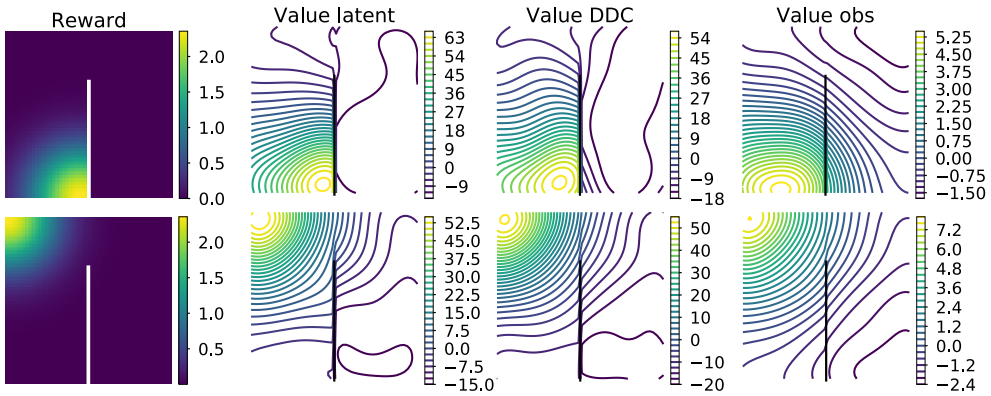


Figure 5.2: Value functions under a random walk policy for two different reward locations. Values were computed using SFs based on the latent, inferred DDC posterior or observed state variables.

latent space and computed from DDC posterior representations both reflect the structure of the environment, while the value function relying on SFs over the observed states fails to learn about the barrier.

To demonstrate that this is not simply due to using the suboptimal random walk policy, but persists through learning, we have learned successor features while adjusting the policy to a given reward function (see figure 5.3). The policy was learned by generalized policy iteration (Sutton and Barto, 1998), alternating between taking actions following a greedy policy and updating the successor features to estimate the corresponding value function.

The value of each state and action was computed from the value function $V(s)$ by a one-step look-ahead, combining the immediate reward with the expected value function having taken a given action:

$$Q(s_t, a_t) = r(s_t) + \gamma \mathbb{E}_{s_{t+1}|s_t,a_t}[V(s_{t+1})] \tag{5.35}$$

In our case, as the value function in the latent space is expressed as a linear function of the features $\boldsymbol{\psi}(s)$: $V(s) = \boldsymbol{w}_{\text{rew}}^T U \boldsymbol{\psi}(s)$ (eq. 5.9-5.10), the expectation

in 5.35 can be expressed as:

$$\mathbb{E}_{s_{t+1}|s_t,a_t}[V(s_{t+1})] = \boldsymbol{w}_{\text{rew}}^T U \cdot \mathbb{E}_{s'|s,a}[\boldsymbol{\psi}(s_{t+1})] \tag{5.36}$$

$$\approx \boldsymbol{w}_{\text{rew}}^T U \cdot P \cdot (\boldsymbol{\psi}(s_t) \times \boldsymbol{\phi}(a_t)) \tag{5.37}$$

Where P is a linear mapping, $P : \Psi \times \Phi \to \Psi$, that contains information about the distribution $p(s_{t+1}|s_t, a_t)$. More specifically, P is trained to predict $\mathbb{E}_{s_{t+1}|s_t,a_t}[\boldsymbol{\psi}(s_{t+1})]$ as a bilinear function of state and action features ($\boldsymbol{\psi}(s_t)$, $\boldsymbol{\phi}(a_t)$). Given the state-action value, we can implement a greedy policy by choosing actions that maximize $Q(s, a)$:

$$a_t \leftarrow \underset{a \in A}{\text{argmax}}\ Q(s_t, a) \tag{5.38}$$

$$= \underset{a \in A}{\text{argmax}}\ r(s_t) + \boldsymbol{w}_{\text{rew}}^T U \cdot P \cdot (\boldsymbol{\psi}(s_t) \times \boldsymbol{\phi}(a_t))) \tag{5.39}$$

$$= \underset{a \in A}{\text{argmax}}\ r(s_t) + \mathbf{b}(s_t)^T \boldsymbol{\phi}(a) \tag{5.40}$$

The argmax operation in eq. 5.40 (possibly over a continuous space of actions) could be biologically implemented by a ring attractor where the neurons receive state dependent input $\mathbf{b}(s)$ through feedforward weights reflecting the tuning $\boldsymbol{\phi}(a)$ of each neuron in the ring.

Just as in figure 5.2, we compute the value function in the fully observed case, using inferred states or using only the noisy observations. For the latter two, we replace $\boldsymbol{\psi}(s_t)$ in eq. 5.40 with the inferred state representation $\boldsymbol{\mu}(\mathcal{O}_t)$ and the observed features $\boldsymbol{\psi}(o_t)$, respectively. As the agent follows the greedy policy and it receives new observations the corresponding SFs are adapted accordingly. Figure 5.3 shows the learned value functions $V^\pi(s)$, $V^\pi(\boldsymbol{\mu})$ and $V^\pi(o)$ for a given reward location and the corresponding dynamics T^π. The agent having access to the true latent state as well as the one using distributional SFs successfully learn policies leading to the rewarded location. As before, the agent learning SFs purely based on observations remains highly sub-optimal.

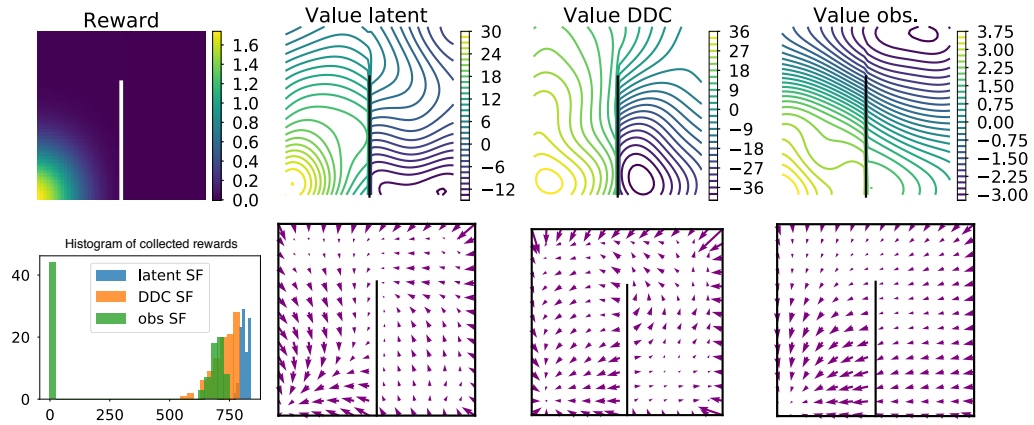


Figure 5.3: Value functions computed using SFs under the learned policy. Top row shows reward and value functions learned in the three different conditions. Bottom row shows histogram of collected rewards from 100 episodes with random initial states, and the learned dynamics T^{π} visualized as in fig. 5.1.

5 Discussion and related work

We have shown that the DDC represention of uncertainty over latent variables can be naturally integrated with representations of uncertainty about future states, and thus offers a natural generalisation of SRs to more realistic environments with partial observability. The proposed algorithm jointly tackles the problem of learning the latent variable model and learning to perform online inference by filtering. Distributional SFs are applicable to POMDPs with continuous or discrete variables and leverage a flexible posterior approximation, not restricted to a simple parametric form, that is represented in a population of neurons in a distributed fashion.

While parametrising the latent dynamics with DDCs is attractive as it makes computing the SFs in the latent space analytically tractable and allows for computing distributional SFs by recurrent dynamics (sec. 4.2.2), it is so far unclear how sampling from such a model might be implemented by neural circuits. Alternatively, one can consider a standard exponential family parametrisation which remains compatible with sleep and wake phase TD learning of distributional SFs.

Earlier work on biological reinforcement learning in POMDPs was restricted

to the case of binary or categorical latent variables where the posterior beliefs can be computed analytically (Rao, 2010). Furthermore the transition model of the POMDP was assumed to be known, rather than learned as in the present work.

Here, we have defined distributional SFs over states, using single step look-ahead to compute state-action values (eq. 5.35). Alternatively, SFs could be defined directly over both states and actions (Kulkarni et al., 2016; Barreto et al., 2017) whilst retaining the distributional development presented here. Barreto et al. (2017, 2019) have shown that successor representations corresponding to previously learned tasks can be used as a basis to construct policies for novel tasks, enabling generalization. Our framework can be extended in a similar way, eliminating the need to adapt the SFs as the policy of the agent changes.

In section 4.2, we proposed a number of different schemes for learning SFs in the latent space that interpolate between model-based and model-free approaches. These are not mutually exclusive but could be applicable in different stages of familiarity with an environment. Learning distributional SFs during the wake phase is advantageous when the model of the environment is not fully learned yet, as it does not rely on simulating experiences from the latent dynamics but uses it only indirectly for filtering given actual observations. Whereas after sufficient exposure to the environment, updating SFs using *sleep phase* simulations from the model—analogously to the Dyna algorithm (Sutton, 1990)—allows for evaluating policies offline and makes learning more data efficient.

The way we presented here, the recognition model produces posterior expectations of the DDC features $\psi(s)$ and thereby captures uncertainty about the latent sate s. The SF are then computed based on these posteriors, exploiting the correspondence between the DDC and successor features. An alternative is for the recognition model to be trained to update the distributional SFs directly. As they are linearly related to the DDCs (eq. 5.28), the distributional SFs capture uncertainty about the latent state s in a similar fashion.

The distributional SFs generalise SFs to handle partially observability, while preserving the predictions made by the existing framework considering fully observable environments. In the limiting case of no posterior uncertainty, i.e. when $p(s_t|\mathcal{O}_t) = \delta(s_t - \hat{s}_t)$, the distributional successor features relax back to the SFs defined over fully observed sates: $\mathbb{E}_{\delta(s_t-\hat{s}_t)}[M(s_t)] = (I - \gamma T)^{-1}[\psi(\hat{s}_t)]$.

The neurotransmitter dopamine has long been hypothesised to signal reward prediction errors (RPE) and thus to play a key role in temporal difference learning (Schultz et al., 1997). More recently, it has been argued that dopamine activity is consistent with RPEs computed based on belief states rather than sensory observations directly (Babayan et al., 2018; Lak et al., 2017; Sarno et al., 2017). Thus dopamine is well suited to carry the information necessary for learning value functions under state uncertainty. In another line of experimental work, dopamine has been found to signal *sensory* prediction errors even if the absence of an associated change in value (Takahashi et al., 2017), suggesting a more general role of dopamine in learning (Gershman, 2018; Gardner et al., 2018). Gardner et al. have proposed that dopamine—signalling prediction error over features of state—may provide the neural substrate for the error signals necessary to learn successor representations. Distributional SFs unify these two sets of observations and their theoretical implications in a single framework. They posit that prediction errors are computed over the posterior belief about latent states (represented as DDCs), and that these predictions are defined over a set of non-linear features of the hidden state rather than reward.

The framework for learning distributional successor features presented here also provides a link between various intriguing and seemingly disparate experimental observations in the hippocampus. The relationship between hippocampal place cell activity and (non-distributional) SRs has been explored previously (e.g., Stachenfeld et al., 2014; 2017) providing an interpretation for phenomena such as "splitter" cells, which show spatial tuning that depends on the whole trajectory (i.e. policy) traversed by the animal not just on its current position (Grieves et al., 2016). However, as discussed earlier, relevant states for

a given reinforcement learning problem (in this case states over which the SR should be learned) cannot be assumed to be directly available to the agent but must be inferred from observations. The hypothesis that hippocampal place cell activity encodes *inferred* location, with its concomitant uncertainty, has also been linked to experimental data (Madl et al., 2014). Thus, our approach connects these two separate threads in the literature and thereby encompasses both groups of experimental results.

Hollup et al. (2001) observed that place fields distributed asymmetrically around the rewarded location in an annular water maze. The animals were trained to swim in a given direction towards a platform hidden under water. They found that the segment of the maze containing the platform had the highest density of place fields followed by the segment that preceded the platform according to the trajectory of the animal. Stachenfeld et al. (2017) showed in simulations that SRs together with uncertainty about location can account for this asymmetry in the firing field distribution. To mimic uncertainty, they convolved the SR by a Gaussian kernel, leading to a shift in the peak of the firing fields due to the asymmetry in the SR. Importantly, convolving with a Gaussian kernel is equivalent to taking an expectation with respect to a corresponding Gaussian distribution, therefore, it is tantamount to the distributional SFs introduced here—assuming a Gaussian posterior distribution over the location of the animal.

Lastly, the framework helps to link simulation of an internal model to learning. Acquisition of the inference model in our framework requires simulating experience (sleep samples) from the agent's current model of the environment, to provide the basis for an update of the recognition model. The sleep samples reflect the agent's knowledge of the environmental dynamics but they need not correspond exactly to a previously experienced trajectory. This is reminiscent of hippocampal "replay" which does not just recapitulate previous experience, but often represents novel trajectories not previously experienced by the animal (Gupta et al., 2010; Ólafsdóttir et al., 2015; Stella et al., 2019). Relatedly, Liu

et al. (2019) recently observed that replay events in humans reflect abstract structural knowledge of a learned task. Our model suggests a novel functional interpretation of these replayed trajectories; namely, that they may play an important role in *learning to infer* relevant latent states from observations. This accords with the observation that experimental interference with replay events impedes learning in contexts where optimal actions depend on history-based inference (Jadhav et al., 2012).

In sum, distributional SFs provide interpretation for a variety of experimental observations and a step towards algorithmic solutions for flexible decision making in realistic and challenging problem settings animals face, i.e. under state uncertainty.

E Appendix

E.1 Wake phase update for distributional SFs

Here, we give some additional insights into the nature of the approximation implied by the wake phase update for the DDC state-space model and discuss its link to variational methods.

According to the standard M step in variational EM, the model parameters are updated to maximize the expected log-joint of the model under the approximate posterior distributions:

$$\Delta\theta \propto \nabla_\theta \sum_t \mathbb{E}_{q(s_t, s_{t+1}|\mathcal{O}_{t+1})}[\log p_\theta(s_{t+1}|s_t)] \tag{5.41}$$

$$= \nabla_\theta \sum_t -\int q(s_t, s_{t+1}|\mathcal{O}_{t+1})(\log p_\theta(s_{t+1}|s_t) + \log q(s_t|\mathcal{O}_{t+1}))d(s_t, s_{t+1}) \tag{5.42}$$

$$= \nabla_\theta \sum_t -KL[q(s_t, s_{t+1}|\mathcal{O}_{t+1})\|p_\theta(s_{t+1}|s_t)q(s_t|\mathcal{O}_{t+1})] \tag{5.43}$$

After projecting the distributions appearing in the KL divergence (eq. 5.43) into the joint exponential family defined by sufficient statistics $[\psi(s_t), \psi(s_{t+1})]$, they can be represented using the corresponding mean parameters:

$$q(s_t, s_{t+1}|\mathcal{O}_{t+1}) \overset{\mathcal{P}}{\Longrightarrow} \begin{bmatrix} \mathbb{E}_{q(s_t, s_{t+1}|\mathcal{O}_{t+1})}[\psi(s_t)] \\ \mathbb{E}_{q(s_t, s_{t+1}|\mathcal{O}_{t+1})}[\psi(s_{t+1})] \end{bmatrix} = \begin{bmatrix} \mu_t(\mathcal{O}_{t+1}) \\ \mu_{t+1}(\mathcal{O}_{t+1}) \end{bmatrix} \tag{5.44}$$

$$p_\theta(s_{t+1}|s_t)q(s_t|\mathcal{O}_{t+1}) \overset{\mathcal{P}}{\Longrightarrow} \begin{bmatrix} \mathbb{E}_{p_\theta(s_{t+1}|s_t)q(s_t|\mathcal{O}_{t+1})}[\psi(s_t)] \\ \mathbb{E}_{p_\theta(s_{t+1}|s_t)q(s_t|\mathcal{O}_{t+1})}[\psi(s_{t+1})] \end{bmatrix} = \begin{bmatrix} \mu_t(\mathcal{O}_{t+1}) \\ T\mu_t(\mathcal{O}_{t+1}) \end{bmatrix} \tag{5.45}$$

To restrict ourselves to online inference, we can make a further approximation: $\mu_t(\mathcal{O}_{t+1}) \approx \mu_t(\mathcal{O}_t)$. Thus, the wake phase update can be thought of as replacing the KL divergence in equation 5.43 by the Euclidean distance

between the (projected) mean parameter representations in eq. 5.44-5.45.

$$\sum_t \|\mu_{t+1}(\mathcal{O}_{t+1}) - T\mu_t(\mathcal{O}_t)\|^2 \tag{5.46}$$

Note that this cost function is directly related to the maximum mean discrepancy (Gretton et al. (2012); MMD)–a non-parametric distance metric between two distributions–with a finite dimensional RKHS.

E.2 Equivalence of sleep and wake phase TD

Here, we show that the posterior expectation of the SFs learned in the latent space during sleep phase ($\mathbb{E}_{p(s_t|\mathcal{O}_t)}[M(s_t)]$) is equivalent to the SFs learned during wake phase ($\widetilde{M}(\mu_t(\mathcal{O}_t))$) if the generative model matches the data distribution and the recognition model produces the exact posterior.

Wake phase TD learns to approximate the following expression:

$$\widetilde{M}(\mu_t(\mathcal{O}_t)) = \mathbb{E}_{p(\mathcal{O}_{>t}|\mathcal{O}_t)}[\sum_{k=0}^{\infty} \gamma^k \mu_{t+k}(\mathcal{O}_{t+k})] \tag{5.47}$$

Where

$$\begin{aligned}
\mathbb{E}_{p(\mathcal{O}_{>t}|\mathcal{O}_t)}[\mu_{t+k}(\mathcal{O}_{t+k})] &= \mathbb{E}_{p(\mathcal{O}_{t+1:t+k}|\mathcal{O}_t)}[\mu_{t+k}(\mathcal{O}_{t+k})] \qquad (5.48)\\
&= \int d\mathcal{O}_{t+1:t+k}\ p(\mathcal{O}_{t+1:t+k}|\mathcal{O}_t) \int ds_{t+k}\ p(s_{t+k}|\mathcal{O}_{t+k})\psi(s_{t+k}) \\
&= \int d\mathcal{O}_{t+1:t+k}\ p(\mathcal{O}_{t+1:t+k}|\mathcal{O}_t) \int ds_{t+k}\ \frac{p(s_{t+k}, \mathcal{O}_{t+1:t+k}|\mathcal{O}_t)}{p(\mathcal{O}_{t+1:t+k}|\mathcal{O}_t)}\psi(s_{t+k}) \\
&= \int ds_{t+k} \int d\mathcal{O}_{t+1:t+k}\ p(s_{t+k}, \mathcal{O}_{t+1:t+k}|\mathcal{O}_t)\psi(s_{t+k}) \\
&= \int ds_{t+k} p(s_{t+k}|\mathcal{O}_t)\psi(s_{t+k}) \qquad (5.49)\\
&= T^k\mu_t
\end{aligned}$$

Thus, we have:

$$\begin{aligned}\widetilde{M}(\mu_t(\mathcal{O}_t)) &= \sum_{k=0}^{\infty} \gamma^k T^k \mu_t \\ &= (I - \gamma T)^{-1} \mu_t \\ &= \mathbb{E}_{p(s_t|\mathcal{O}_t)}[M(s_t)]\end{aligned} \tag{5.50}$$

E.3 Further experimental details

Figure 1: Learning and inference in the DDC state-space model

The generative model corresponding to a random walk policy:

$$\begin{aligned}p(s_{t+1}|s_t) &= [s_t + \tilde{\eta}]_{\text{WALLS}}, \\ p(o_t|s_t) &= s_t + \xi\end{aligned} \tag{5.51}$$

Where $[.]_{\text{WALLS}}$ indicates the constraints introduced by the walls in the environment (outer walls are of unit length). $\eta \sim \mathcal{N}(0, \sigma_s = 1.), \tilde{\eta} = 0.06 * \eta / \|\eta\|$, $\xi \sim \mathcal{N}(0, \sigma_o = 0.1)$, $s_t, o_t \in \mathbb{R}^2$

We used $K = 100$ Gaussian features with width $\sigma_\psi = 0.3$ for both the latent and observed states. A small subset of features were truncated along the internal wall, to limit the artifacts from the function approximation. Alternatively, a features with various spatial scales could also be used. The recursive recognition model was parametrized linearly using the features:

$$f_W(\mu_{t-1}, o_t) = W[T\mu_{t-1}; \psi(o_t)] \tag{5.52}$$

As sampling from the DDC parametrized latent dynamics is not tractable in general, in the sleep phase, we generated approximate samples from a Gaussian distribution with consistent mean.

The generative and recognition models were trained through 50 wake-sleep cycles, with $3 \cdot 10^4$ sleep samples, and $5 \cdot 10^4$ wake phase observations.

The latent dynamics in Fig.1b is visualized by approximating the mean

dynamics as a linear readout from the DDC: $\mathbb{E}_{s_{t+1}|s_t}[s_{t+1}] \approx \alpha T \psi(s_t)$ where $s \approx \alpha \psi(s)$.

Figure 2 Values under a random walk policy To compute the value functions under the random walk policy we computed the SFs based on the latent ($\psi(s)$), inferred (μ) or observed ($\psi(o)$) features, with discount factor $\gamma = 0.99$. In each case, we estimated the reward vector w_{rew} using the available state information.

Figure 3 Values under a learned policy To construct the state-action value function, we used 10 features over actions $\phi(a)$, von Mises functions ($\kappa = 2.$) arranged evenly on $[0, 2\pi]$. The policy iteration was run for 500 cycles, and in each cycle an episode of 500 steps was collected according to the greedy policy. The visited latent, inferred of observed state sequences were used to update the corresponding SFs to re-evaluate the policy. To facilitate faster learning, only episodes with positive returns were used to update the SFs.

Part III

General Conclusions

A concept reoccurring throughout this thesis is the idea of encoding uncertainty about unobservable variables as a set of expectation of non-linear features, a representation we called distributed distributional codes (DDC). As we have seen in multiple examples, lot of the computations involving uncertainty take the form of an expectation with respect to the corresponding distribution. The computational appeal of DDCs is that they are very well suited to such computations, and translate the often intractable integrals to learning simple linear mappings. We have demonstrated that when combined with the wake-sleep algorithm the resulting DDC Helmholtz machine can learn hierarchical latent variable models with greater accuracy than variational methods that employ rigid posterior approximations.

We have also shown how DDCs can lead to a natural generalisation of successor features to partially observable settings by relying on the conjugacy of the two representations. In distributional successor features the DDC expectations play a dual role: they serve as belief states and provide a basis set for a linear decomposition of the reward function. This is not only a computational convenience but also accords with a number different experimental observations.

Unsupervised learning approaches, like latent variable models, are explicitly trained to capture structure in the outside world and produce posterior beliefs over latent variables. The resulting representations, at least in an idealised case, correctly handle uncertainty about unobserved quantities and facilitate generalisation or transfer to novel tasks. Such flexibility has been observed in animal and human behaviour and it is a desideratum for artificial intelligent systems as well. Supervised training with a range of auxiliary tasks or multi-task learning can in principle discover similar representations. Furthermore, in a neural network trained in a supervised multi-task setting, one can expect representations akin to DDCs to emerge, due to the analogy to sleep phase learning where the supervision is provided by the generative model. This parallel also prompts questions relevant for neuroscience: to what extent early sensory representations are shaped by the tasks performed or by pure unsupervised

learning and how these two might interact.

An example of a useful auxiliary task is the distributional objective in reinforcement learning, that has recently received some empirical support in neuroscience (Dabney et al., 2020), and that aids representation learning by separating states that predict the same expected reward but different reward distributions. It is likely that similar benefits can be ascribed to learning distributional representations over other task-relevant variables, even in tasks where information about uncertainty is not directly probed.

There is a consensus in the literature about the idea that the brain is able to perform some forms of probabilistic computation, largely originating from behaviour evidence. This also suggests that uncertainty about relevant variables needs to be represented in neural systems, at least implicitly. However, how brains learn to perform these computations or how uncertainty is reflected in neural activity is much less clear. Here, we reviewed a number of theoretical proposals and attempted to clarify some of the conceptual issues regarding studying representations of uncertainty experimentally. We hope that our discussion will facilitate future research on the topic and help gain insights into how brains learn to reason accurately in the face of uncertainty.

Another open question about Bayesian inference and learning in the brain is whether there is an explicit representation of the generative model as in the DDC Helmholtz machine and other probabilistic unsupervised learning algorithms. Exceptions include a family of methods that rely on a contrastive loss function, an approach that has received renewed attention in machine learning (Hyvarinen and Morioka, 2016; Oord et al., 2019; Hénaff et al., 2019). These methods aim to extract representations by exploiting temporal structure in the data, looking for representations that are predictive of the future. Related ideas have been present in neuroscience as well: slow feature analysis (Wiskott and Sejnowski, 2002) finds components in the data that change slowly over time, and has been proposed as a model of learning in the early visual cortex. Temporal consistency of our sensory experience is a powerful prior and could

be leveraged in neural systems as well to learn probabilistic models where the generative process is only implicit.

Colophon

This document was set in the Latin Modern Roman typeface using LaTeX and BibTeX, composed in the editor Texpad. Figures were produced using matplotlib, seaborn, and tikz packages.

Bibliography

M. B. Ahrens and M. Sahani. Observers exploit stochastic models of sensory change to help judge the passage of time. *Curr. Biol.*, 21(3):200–206, 2011.

L. Aitchison and M. Lengyel. The Hamiltonian Brain: Efficient Probabilistic Inference with Excitatory-Inhibitory Neural Circuit Dynamics. *PLOS Computational Biology*, 12(12):e1005186, 2016.

D. Alais and D. Burr. The ventriloquist effect results from near-optimal bimodal integration. *Curr. Biol.*, 14(3):257–262, 2004.

C. H. Anderson. Basic elements of biological computional systems. *International Journal of Modern Physics C*, 5, 1994.

C. H. Anderson and D. C. V. Essen. *Neurobiological Computational Systems*. 1994.

J. S. Anderson, I. Lampl, D. C. Gillespie, and D. Ferster. The Contribution of Noise to Contrast Invariance of Orientation Tuning in Cat Visual Cortex. *Science*, 290(5498):1968–1972, 2000.

B. M. Babayan, N. Uchida, and S. J. Gershman. Belief state representation in the dopamine system. *Nat Commun*, 9(1):1891, 2018.

A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. Mankowitz, A. Žídek, and R. Munos. Transfer in Deep Reinforcement Learning Using Successor Features and Generalised Policy Improvement. *arXiv:1901.10964 [cs]*, 2019.

P. W. Battaglia and P. R. Schrater. Humans trade off viewing time and movement duration to improve visuomotor accuracy in a fast reaching task. *J. Neurosci.*, 27(26):6984–6994, 2007.

J. M. Beck, P. E. Latham, and A. Pouget. Marginalization in Neural Circuits with Divisive Normalization. *J. Neurosci.*, 31(43):15310–15319, 2011.

A. J. Bell and T. J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

P. Berkes, G. Orbán, M. Lengyel, and J. Fiser. Spontaneous Cortical Activity Reveals Hallmarks of an Optimal Internal Model of the Environment. *Science*, 331(6013):83–87, 2011.

R. Blake and N. K. Logothetis. Visual competition. *Nature Reviews Neuroscience*, 3(1):13–21, 2002.

W. Bounliphone, E. Belilovsky, M. B. Blaschko, I. Antonoglou, and A. Gretton. A Test of Relative Similarity For Model Selection in Generative Models. *arXiv:1511.04581 [cs, stat]*, 2015.

L. Buesing, J. Bill, B. Nessler, and W. Maass. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS computational biology*, 7(11):e1002211, 2011.

Y. Burda, R. Grosse, and R. Salakhutdinov. Importance Weighted Autoencoders. *arXiv:1509.00519 [cs, stat]*, 2015.

N. Chater, J. B. Tenenbaum, and A. Yuille. Probabilistic models of cognition: conceptual foundations. *Trends Cogn. Sci. (Regul. Ed.)*, 10(7):287–291, 2006.

W. Dabney, Z. Kurth-Nelson, N. Uchida, C. K. Starkweather, D. Hassabis, R. Munos, and M. Botvinick. A distributional code for value in dopamine-based reinforcement learning. *Nature*, 577(7792):671–675, 2020.

I. Dasgupta, E. Schulz, N. D. Goodman, and S. J. Gershman. Remembrance of inferences past: Amortization in human hypothesis generation. *Cognition*, 178:67–81, 2018.

N. D. Daw, Y. Niv, and P. Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8(12):1704, 2005.

N. D. Daw, S. J. Gershman, B. Seymour, P. Dayan, and R. J. Dolan. Model-Based Influences on Humans' Choices and Striatal Prediction Errors. *Neuron*, 69(6):1204–1215, 2011.

P. Dayan. Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation*, 5(4):613–624, 1993.

P. Dayan. *Helmholtz Machines and Wake-Sleep Learning, Handbook of Brain Theory and Neural Networks, 2.* 2000.

P. Dayan and N. D. Daw. Decision theory, reinforcement learning, and the brain. *Cognitive, Affective, & Behavioral Neuroscience*, 8(4):429–453, 2008.

P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

A. F. Dean. The variability of discharge of simple cells in the cat striate cortex. *Exp Brain Res*, 44(4):437–440, 1981.

M. O. Ernst and M. S. Banks. Humans integrate visual and haptic information in a statistically optimal fashion. *Nature*, 415(6870):429, 2002.

C. R. Fetsch, A. H. Turner, G. C. DeAngelis, and D. E. Angelaki. Dynamic reweighting of visual and vestibular cues during self-motion perception. *J. Neurosci.*, 29(49):15601–15612, 2009.

I. M. Finn, N. J. Priebe, and D. Ferster. The Emergence of Contrast-Invariant Orientation Tuning in Simple Cells of Cat Visual Cortex. *Neuron*, 54(1): 137–152, 2007.

K. Fukumizu, F. R. Bach, and M. I. Jordan. Dimensionality Reduction for Supervised Learning With Reproducing Kernel Hilbert Spaces:. Technical report, Defense Technical Information Center, Fort Belvoir, VA, 2003.

M. P. H. Gardner, G. Schoenbaum, and S. J. Gershman. Rethinking dopamine as generalized prediction error. *Proc. Biol. Sci.*, 285(1891), 2018.

S. J. Gershman. The Successor Representation: Its Computational Logic and Neural Substrates. *J. Neurosci.*, 38(33):7193–7200, 2018.

S. J. Gershman and N. D. Goodman. Amortized Inference in Probabilistic Reasoning. In *CogSci*, 2014.

S. Ghadimi and G. Lan. Stochastic First- and Zeroth-order Methods for Nonconvex Stochastic Programming. *arXiv:1309.5549 [cs, math, stat]*, 2013.

J. Gläscher, N. Daw, P. Dayan, and J. P. O'Doherty. States versus Rewards: Dissociable Neural Prediction Error Signals Underlying Model-Based and Model-Free Reinforcement Learning. *Neuron*, 66(4):585–595, 2010.

A. Grabska-Barwińska, S. Barthelmé, J. Beck, Z. F. Mainen, A. Pouget, and P. E. Latham. A probabilistic approach to demixing odors. *Nature Neuroscience*, 20(1):98–106, 2017.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13:723–773, 2012.

R. M. Grieves, E. R. Wood, and P. A. Dudchenko. Place cells on a maze encode routes rather than destinations. *eLife*, 5:e15986, 2016.

S. Grünewälder, G. Lever, L. Baldassarre, S. Patterson, A. Gretton, and M. Pontil. Conditional mean embeddings as regressors. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 2: 1823–1830, 2012.

Y. Gu, D. E. Angelaki, and G. C. Deangelis. Neural correlates of multisensory cue integration in macaque MSTd. *Nat. Neurosci.*, 11(10):1201–1210, 2008.

A. S. Gupta, M. A. A. van der Meer, D. S. Touretzky, and A. D. Redish. Hippocampal replay is not a simple function of experience. *Neuron*, 65(5): 695–705, 2010.

H. v. Helmholtz. *Handbuch der physiologischen Optik*. L. Voss, 1867.

O. J. Hénaff, A. Srinivas, J. De Fauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. v. d. Oord. Data-Efficient Image Recognition with Contrastive Predictive Coding. *arXiv:1905.09272 [cs]*, 2019.

G. Hennequin, L. Aitchison, and M. Lengyel. Fast Sampling-Based Inference in Balanced Neuronal Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2240–2248. Curran Associates, Inc., 2014.

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

S. A. Hollup, S. Molden, J. G. Donnett, M. B. Moser, and E. I. Moser. Accumulation of hippocampal place fields at the goal location in an annular watermaze task. *J. Neurosci.*, 21(5):1635–1644, 2001.

P. O. Hoyer and A. Hyvärinen. Interpreting Neural Response Variability as Monte Carlo Sampling of the Posterior. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 293–300. MIT Press, 2003.

A. Hyvarinen and H. Morioka. Unsupervised Feature Extraction by Time-Contrastive Learning and Nonlinear ICA. *arXiv:1605.06336 [cs, stat]*, 2016.

S. Ikeda, S.-i. Amari, and H. Nakahara. Convergence of the Wake-Sleep Algorithm. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 239–245. MIT Press, 1999.

R. A. Jacobs. Optimal integration of texture and motion cues to depth. *Vision Research*, 39(21):3621–3629, 1999.

S. P. Jadhav, C. Kemere, P. W. German, and L. M. Frank. Awake Hippocampal Sharp-Wave Ripples Support Spatial Memory. *Science*, 336(6087):1454–1458, 2012.

W. Jitkrittum, Z. Szabo, K. Chwialkowski, and A. Gretton. Interpretable Distribution Features with Maximum Testing Power. *arXiv:1605.06796 [cs, stat]*, 2016.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2): 183–233, 1999.

D. Kingma, Welling, M., and Amsterdam Machine Learning lab (IVI, FNWI). Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations (ICLR2014)*. arXiv.org, 2014.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, 2014.

D. C. Knill. Surface orientation from texture: ideal observers, generic observers and the information content of texture cues. *Vision Research*, 38(11):1655–1682, 1998.

D. C. Knill and W. Richards. *Perception as Bayesian Inference*. Cambridge University Press, 1996.

K. P. Körding and D. M. Wolpert. Bayesian integration in sensorimotor learning. *Nature*, 427(6971):244, 2004.

T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman. Deep Successor Reinforcement Learning. *arXiv:1606.02396 [cs, stat]*, 2016.

A. Kutschireiter, S. C. Surace, H. Sprekeler, and J.-P. Pfister. Nonlinear Bayesian filtering and learning: a neuronal dynamics for perception. *Scientific Reports*, 7(1):8722, 2017.

A. Lak, K. Nomoto, M. Keramati, M. Sakagami, and A. Kepecs. Midbrain Dopamine Neurons Signal Belief in Choice Accuracy during a Perceptual Decision. *Curr. Biol.*, 27(6):821–832, 2017.

Y. Liu, R. J. Dolan, Z. Kurth-Nelson, and T. E. J. Behrens. Human Replay Spontaneously Reorganizes Experience. *Cell*, 178(3):640–652.e14, 2019.

N. K. Logothetis and J. D. Schall. Neuronal correlates of subjective visual perception. *Science*, 245(4919):761–763, 1989.

W. J. Ma, J. M. Beck, P. E. Latham, and A. Pouget. Bayesian inference with probabilistic population codes. *Nature Neuroscience*, 9(11):1432–1438, 2006.

P. R. MacNeilage, M. S. Banks, D. R. Berger, and H. H. Bülthoff. A Bayesian model of the disambiguation of gravitoinertial force by visual cues. *Exp Brain Res*, 179(2):263–290, 2007.

T. Madl, S. Franklin, K. Chen, D. Montaldi, and R. Trappl. Bayesian integration of information in hippocampal place cells. *PLoS ONE*, 9(3):e89762, 2014.

J. G. Makin, M. R. Fellows, and P. N. Sabes. Learning Multisensory Integration and Coordinate Transformation via Density Estimation. *PLOS Computational Biology*, 9(4):e1003035, 2013.

J. G. Makin, B. K. Dichter, and P. N. Sabes. Learning to Estimate Dynamical State with Probabilistic Population Codes. *PLOS Computational Biology*, 11 (11):e1004554, 2015.

N. Metropolis and S. Ulam. The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Comput*, 17(1):177–204, 2005.

T. Minka. Divergence Measures and Message Passing. *Microsoft Research*, 2005.

A. Mnih and D. Rezende. Variational Inference for Monte Carlo Objectives. In *PMLR*, pages 2188–2196, 2016.

I. Momennejad, E. M. Russek, J. H. Cheong, M. M. Botvinick, N. D. Daw, and S. J. Gershman. The successor representation in human reinforcement learning. *Nature Human Behaviour*, 1(9):680, 2017.

M. L. Morgan, G. C. Deangelis, and D. E. Angelaki. Multisensory integration in macaque visual cortex depends on cue reliability. *Neuron*, 59(4):662–673, 2008.

K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf. Kernel Mean Embedding of Distributions: A Review and Beyond. *FNT in Machine Learning*, 10(1-2):1–141, 2017.

K. P. Murphy. *Machine learning: a probabilistic perspective*. Cambridge, MA, 2012.

R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56(1):71–113, 1992.

H. F. Ólafsdóttir, C. Barry, A. B. Saleem, D. Hassabis, and H. J. Spiers. Hippocampal place cells construct reward related sequences through unexplored space. *Elife*, 4:e06063, 2015.

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

A. v. d. Oord, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv:1807.03748 [cs, stat]*, 2019.

G. Orbán, P. Berkes, J. Fiser, and M. Lengyel. Neural Variability and Sampling-Based Probabilistic Representations in the Visual Cortex. *Neuron*, 92(2): 530–543, 2016.

A. E. Orhan and W. J. Ma. Efficient probabilistic inference in generic neural networks trained with non-probabilistic feedback. *Nature Communications*, 8(1):138, 2017.

E. Parzen. On Estimation of a Probability Density Function and Mode. *Ann. Math. Statist.*, 33(3):1065–1076, 1962.

A. Pouget, P. Dayan, and R. S. Zemel. Inference and computation with population codes. *Annual review of neuroscience*, 26(1):381–410, 2003.

A. Pouget, J. M. Beck, W. J. Ma, and P. E. Latham. Probabilistic brains: knowns and unknowns. *Nat. Neurosci.*, 16(9):1170–1178, 2013.

A. Rahimi and B. Recht. Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 555–561, 2008.

R. V. Raju and Z. Pitkow. Inference by Reparameterization in Neural Population Codes. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2029–2037. Curran Associates, Inc., 2016.

R. Ranganath, L. Tang, L. Charlin, and D. Blei. Deep exponential families. In *Artificial Intelligence and Statistics*, pages 762–771, 2015.

R. P. N. Rao. Decision Making Under Uncertainty: A Neural Model Based on Partially Observable Markov Decision Processes. *Front. Comput. Neurosci.*, 4, 2010.

D. J. Rezende and S. Mohamed. Variational Inference with Normalizing Flows. *arXiv:1505.05770 [cs, stat]*, 2015.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv:1401.4082 [cs, stat]*, 2014.

M. Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *Ann. Math. Statist.*, 27(3):832–837, 1956.

E. M. Russek, I. Momennejad, M. M. Botvinick, S. J. Gershman, and N. D. Daw. Predictive representations can link model-based reinforcement learning to model-free mechanisms. *PLOS Computational Biology*, 13(9):e1005768, 2017.

M. Sahani and P. Dayan. Doubly Distributional Population Codes: Simultaneous Representation of Uncertainty and Multiplicity. *Neural Computation*, 15 (10):2255–2279, 2003.

R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pages 872–879. ACM, 2008.

E. Salinas and L. F. Abbott. Vector reconstruction from firing rates. *J Comput Neurosci*, 1(1):89–107, 1994.

M. Sanjabi, J. Ba, M. Razaviyayn, and J. Lee. On the Convergence and Robustness of Training GANs with Regularized Optimal Transport. 2018.

S. Sarno, V. de Lafuente, R. Romo, and N. Parga. Dopamine reward prediction error signal codes the temporal evaluation of a perceptual decision report. *Proc. Natl. Acad. Sci. U.S.A.*, 114(48):E10494–E10503, 2017.

W. Schultz, P. Dayan, and P. R. Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

O. Schwartz and E. P. Simoncelli. Natural signal statistics and sensory gain control. *Nat. Neurosci.*, 4(8):819–825, 2001.

H. S. Seung and H. Sompolinsky. Simple models for reading neuronal population codes. *PNAS*, 90(22):10749–10753, 1993.

L. Shams, W. J. Ma, and U. Beierholm. Sound-induced flash illusion as an optimal percept. *Neuroreport*, 16(17):1923–1927, 2005.

R. Singh, M. Sahani, and A. Gretton. Kernel Instrumental Variable Regression. *arXiv:1906.00232 [cs, econ, math, stat]*, 2019.

A. Smola, A. Gretton, L. Song, and B. Schölkopf. A Hilbert Space Embedding for Distributions. In M. Hutter, R. A. Servedio, and E. Takimoto, editors, *Algorithmic Learning Theory*, Lecture Notes in Computer Science, pages 13–31. Springer Berlin Heidelberg, 2007.

H. P. Snippe. Parameter extraction from population codes: a critical assessment. *Neural Comput*, 8(3):511–529, 1996.

C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder Variational Autoencoders. *arXiv:1602.02282 [cs, stat]*, 2016.

L. Song, J. Huang, A. Smola, and K. Fukumizu. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 961–968, 2009.

L. Song, A. Gretton, and C. Guestrin. Nonparametric tree graphical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 765–772, 2010.

B. Sriperumbudur, K. Fukumizu, and G. Lanckriet. On the relation between universality, characteristic kernels and RKHS embedding of measures. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 773–780, 2010.

K. L. Stachenfeld, M. Botvinick, and S. J. Gershman. Design Principles of the Hippocampal Cognitive Map. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2528–2536. Curran Associates, Inc., 2014.

K. L. Stachenfeld, M. M. Botvinick, and S. J. Gershman. The hippocampus as a predictive map. *Nature Neuroscience*, 20(11):1643–1653, 2017.

C. K. Starkweather, B. M. Babayan, N. Uchida, and S. J. Gershman. Dopamine reward prediction errors reflect hidden-state inference across time. *Nat. Neurosci.*, 20(4):581–589, 2017.

F. Stella, P. Baracskay, J. O'Neill, and J. Csicsvari. Hippocampal Reactivation of Random Trajectories Resembling Brownian Diffusion. *Neuron*, 2019.

M. Steyvers, T. L. Griffiths, and S. Dennis. Probabilistic inference in human semantic memory. *Trends Cogn. Sci. (Regul. Ed.)*, 10(7):327–334, 2006.

A. A. Stocker and E. P. Simoncelli. Noise characteristics and prior expectations in human visual speed perception. *Nature Neuroscience*, 9(4):578, 2006.

R. S. Sutton. Learning to predict by the methods of temporal differences. *Mach Learn*, 3(1):9–44, 1988.

R. S. Sutton. Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In B. Porter and R. Mooney, editors, *Machine Learning Proceedings 1990*, pages 216–224. Morgan Kaufmann, San Francisco (CA), 1990.

R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Y. K. Takahashi, H. M. Batchelor, B. Liu, A. Khanna, M. Morales, and G. Schoenbaum. Dopamine Neurons Respond to Errors in the Prediction of Sensory Features of Expected Rewards. *Neuron*, 95(6):1395–1405.e3, 2017.

J. B. Tenenbaum and T. L. Griffiths. Theory-based Causal Inference. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, pages 43–50, Cambridge, MA, USA, 2002. MIT Press.

E. Todorov. Optimality principles in sensorimotor control. *Nat. Neurosci.*, 7 (9):907–915, 2004.

D. J. Tolhurst, J. A. Movshon, and A. F. Dean. The statistical reliability of signals in single neurons in cat and monkey visual cortex. *Vision Res.*, 23 (8):775–785, 1983.

J. Trommershauser, K. Kording, and M. S. Landy, editors. *Sensory Cue Integration*. Computational Neuroscience Series. Oxford University Press, Oxford, New York, 2011.

R. E. Turner and M. Sahani. Two problems with variational expectation maximisation for time-series models. In *Bayesian Time series models*, pages 109–130. Cambridge University Press, 2011.

R. J. van Beers, A. C. Sittig, and J. J. D. v. d. Gon. Integration of Proprioceptive and Visual Position-Information: An Experimentally Supported Model. *Journal of Neurophysiology*, 81(3):1355–1364, 1999.

J. H. van Hateren and A. van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proc Biol Sci*, 265(1394):359–366, 1998.

E. Vértes and M. Sahani. Flexible and accurate inference and learning for deep generative models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4166–4175. Curran Associates, Inc., 2018.

E. Vértes and M. Sahani. A neurally plausible model learns successor representations in partially observable environments. In H. Wallach, H. Larochelle,

A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 13692–13702. Curran Associates, Inc., 2019.

M. J. Wainwright and M. I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305, 2008.

M. J. Wainwright and E. P. Simoncelli. Scale Mixtures of Gaussians and the Statistics of Natural Images. In S. A. Solla, T. K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 855–861. MIT Press, 2000.

E. Y. Walker, R. J. Cotton, W. J. Ma, and A. S. Tolias. A neural basis of probabilistic computation in visual cortex. *bioRxiv*, page 365973, 2019.

L. Whiteley and M. Sahani. Implicit knowledge of visual uncertainty guides decisions with asymmetric outcomes. *Journal of vision*, 8(3):2–2, 2008.

L. Wiskott and T. J. Sejnowski. Slow Feature Analysis: Unsupervised Learning of Invariances. *Neural Computation*, 14(4):715–770, 2002.

D. M. Wolpert, Z. Ghahramani, and M. I. Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

T. Yang and M. N. Shadlen. Probabilistic reasoning by neurons. *Nature*, 447 (7148):1075–1080, 2007.

R. S. Zemel and P. Dayan. Distributional Population Codes and Multiple Motion Models. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 174–182. MIT Press, 1999.

R. S. Zemel, P. Dayan, and A. Pouget. Probabilistic interpretation of population codes. *Neural computation*, 10(2):403–430, 1998.