Compiled Notes

Joseph Marino

January 2, 2018

Contents

Chapter 1

Intelligence

1.1 Intelligence

Intelligence is often vaguely defined. In many situations, we use the term to refer to the ability to perceive and understand certain concepts, such as math, art, politics, business, etc., but this narrow definition fails to capture the spectrum that intelligence occupies. Instead, I use the following definition:

Intelligence is the ability of a system to interact meaningfully within an environment.

This definition has a few components: A **system** is some collection of matter: a molecular structure, single-celled organism, animal, machine, computer, human, society, etc. **Interact meaningfully**, in general terms, refers to non-random interactions[1] with the environment, i.e. the distribution of actions given the environmental state has low entropy. *todo: refine this definition.* These actions are typically associated with achieving some goal or reward, although this is not strictly necessary. Finally, **within an environment** emphasizes that intelligence is specific to an environment, also referred to as the data distribution or domain. Intelligence is not a characteristic of a system in isolation; it is always conditioned on a particular context that the system is suited to handle.

1.2 Creating Intelligence

There are two ways in which to create an intelligent system: through **design** and through **learning**. In design, one intelligent system constructs another intelligent system. In learning, a system gains intelligence through interaction with the environment.

[1] We could argue whether or not a passive system that can only *perceive* aspects of its environment is intelligent, but such a system has no practical purpose, since it has no way of communicating this intelligence to the environment.

Chapter 2

Probability & Statistics

2.1 Probability Basics

2.1.1 Definitions

Law of the unconscious statistician

2.1.2 Distributions

Chapter 3

Information Theory

3.1 What is information?

Information is a reduction in uncertainty.

3.2 Basic Concepts

Entropy is a measure of the uncertainty of a random variable. It tends to agree with the notion of information. Let X be a random variable, with alphabet $\mathcal{X}$ and probability mass function $p(x) = \Pr(X = x)$, with $x \in \mathcal{X}$. The entropy of X is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{3.1}$$

The units of entropy depend on the base of the logarithm. If the base is 2, then the units are in bits. If the base is e, then the units are in nats. To convert between different units of entropy, use $H_b(X) = (\log_b a) H_a(X)$. Note that entropy can be interpreted as the expectation of $-\log p(x)$. Also, entropy is non-negative: $H(X) \geq 0$.

The **joint entropy** of multiple random variables is defined similarly:

$$H(X, Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y) \tag{3.2}$$

Likewise, the **conditional entropy** is defined as

$$\begin{aligned} H(Y|X) &= \sum_{x \in \mathcal{X}} p(x) H(Y|X = x) \\ &= -\sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} p(y|x) \log p(y|x) \\ &= -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(y|x). \end{aligned}$$

Note that

$$H(X, Y) = H(X) + H(Y|X). \tag{3.3}$$

In words, joint entropy = individual entropy + conditional entropy. This also implies that

$$H(X, Y|Z) = H(X|Z) + H(Y|X, Z). \tag{3.4}$$

Because conditional entropy involves the conditional probability, we see that $H(Y|X) \neq H(X|Y)$. However,

$$H(X) - H(X|Y) = H(Y) - H(Y|X). \tag{3.5}$$

Relative entropy is a measure of the distance between two distributions. Put differently, it is the inefficiency of assuming the distribution of the data is $q(x)$ when the true distribution is $p(x)$. This is also referred to as the **Kullback-Leibler distance** or **divergence**.

$$D_{KL}(p(x)||q(x)) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} = \mathbb{E}_{x \sim p(x)} \Big[\log \frac{p(x)}{q(x)} \Big] \tag{3.6}$$

Note that if there is a value $x \in \mathcal{X}$ such that $p(x) > 0$ and $q(x) = 0$, then $D_{KL} = \infty$. KL divergence is non-negative and is only zero when $p(x) = q(x)$. It is not symmetric, so it is not a proper distance measure.

Mutual Information is the amount of information that one random variable contains about another random variable. In other words, it is the reduction in the uncertainty of one random variable due to knowledge of the other. It is expressed as the relative entropy between the joint distribution $p(x, y)$ and the product of the marginals $p(x)p(y)$.

$$\begin{aligned} I(X;Y) &= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \\ &= D_{KL}(p(x, y)||p(x)p(y)) \\ &= \mathbb{E}_{x,y \sim p(x,y)} \Big[\log \frac{p(x, y}{p(x)p(y)} \Big]. \end{aligned}$$

This can also be written as

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X). \tag{3.7}$$

Thus, *X says as much about Y as Y says about X*. In words, mutual information is the amount of information in X minus the amount of information in X after observing Y (and vice versa). This corresponds to the amount of information in X that is explained by Y (and, again, vice versa). If Y perfectly explains X, then $H(X|Y) = 0$, so $I(X;Y) = H(X)$. At the other extreme, if Y says nothing about X, then $H(X|Y) = H(X)$, so $I(X;Y) = 0$. Also,

$$I(X;Y) = H(X) + H(Y) - H(X, Y), \tag{3.8}$$

and

$$I(X;X) = H(X) - H(X|X) = H(X). \tag{3.9}$$

Thus, entropy can be considered as the amount of self-information.

Chapter 4

Neuroscience

4.1 Neurons

4.2 Neural Circuits

4.2.1 Retinal Circuits

4.2.2 Neocortical Circuits

A review of various experimental findings on cortical circuits is presented in [1]. While there are differences in cortical structure and activity across different areas and species, many cortical circuit properties are conserved. [1] therefore argues that these differences are *quantitative* rather than *qualitative*, arising from differences in gene expression and inputs. They refer to these slight differences on a main conserved architecture as *serial homology*. The primary categorization of neocortical neurons is between excitatory cells (EC), which constitute roughly 80% of neurons, and interneurons, which constitute the remaining 20%.

A somewhat more simplistic view, with a focus on predictive coding, is presented in [2].

4.2.2.1 Excitatory Cells

Excitatory cells are classified into three main categories:

- **intratelencephalic (IT) neurons**: Found in layers 2-6 and project axons only within the telecephalon (neocortex, striatum, and corticoid structures such as amygdala and claustrum). They are the only ECs that project to contralateral cortex. There are many distinct subclasses of IT cells, such as those found in L4.
- **pyramidal tract (PT) neurons**: These are large pyramidal neurons found in layer 5B that project to subcerebral targets including brain stem, spinal cord and midbrain, as well as thalamus and striatum.
- **corticothalamic (CT) neurons**: Found in layer 6 and project primarily to ipsilateral thalamus.

Morphologies of these cell types are shown in Figure 4.1. It should be noted that all class of ECs form recurrent connections with local neurons of the same class. Across classes,

however, the connectivity is asymmetric, which has led to the hypothesis of a sequential circuit organization. This sequence is typically understood in terms of

$$\text{thalamus} \rightarrow \text{L4 IT neurons} \rightarrow \text{Other IT neurons} \rightarrow \text{PT neurons},$$

with the role of CT neurons within the circuit still largely unknown.

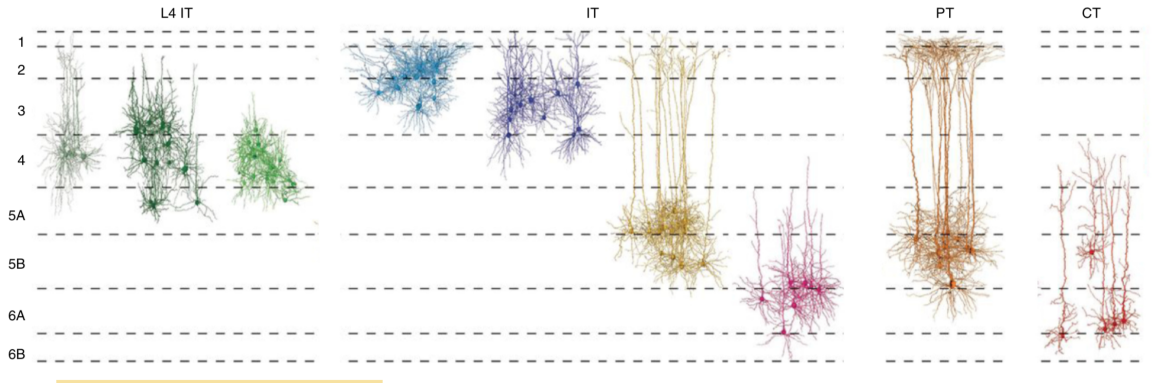

Figure 4.1: Morphologies of various classes of excitatory cells in neocortex. Reproduced from [1].

Most of the subcortical inputs to the cortex arise from thalamus and are roughly divided into "core" and "matrix" connections. The core relay neurons carry rapid sensory or motor information and are located in primary relay nuclei. These axons tend to form topographic connections in L4 or primary sensory areas. The matrix relay neurons are not well understood, but are typically found in higher order nuclei and project to L1 (and sometimes other layers) of one or more cortical areas. The core-matrix distinction applies most directly to primary sensory areas, but is also found in motor areas and possibly other areas. Higher order sensory cortex, for instance, receives predominantly matrix-type thalamocortical (TC) connections.

L4 neurons are a special case of IT neurons. They project asymmetrically to L2/3 and L5, and are therefore considered "upstream" of other circuit components. There are many morphological subclasses of L4 IT neurons, but they largely have similar circuit properties. Layer 4 is enlarged in primary sensory areas, so L4 neurons are thought to be specialized for sensory processing, with different sensory areas tuned appropriately. In higher-order sensory areas, L4 receives input from thalamus as well as lower areas of cortex. Although motor areas appear to lack a well defined L4, they may contain L4 type cells.

Other IT neurons receive input from L4 neurons and TC connections. Their outputs project to distant neocortical and striatum areas as well as locally to PT and CT neurons. L2 IT neurons receive matrix-type TC input and inputs from L5A and L4. L3 IT neurons receive similar inputs, with the addition of core-type TC inputs. These neurons tend to exhibit sparse firing and project to L5. The L5 IT neurons tend to be reciprocally connected to L2/3 and also have broad range connections, particularly to striatum. L6 IT neurons are less studied but tend to make distant connections.

PT neurons receive cortical (L2/3) and TC (core-type) input and project to specialized subcortical areas, such as spinal or tectal areas. They also make connections with ipsilateral cortex and thalamus. They tend to display bursting firing patterns, which provide a "dense code." This is perhaps an information-theoretic adaptation to broadcast cortical output through a small number of channels.

CT neurons are found in L6, and tend to receive input from higher-order cortical areas, as well as some input from thalamus. Their projections to thalamic areas are slow and weak, so they are often considered as modulators instead of drivers. It is thought that CT neurons may integrate long-range cortical inputs to modulate TC activity, acting as a sort of gain control.

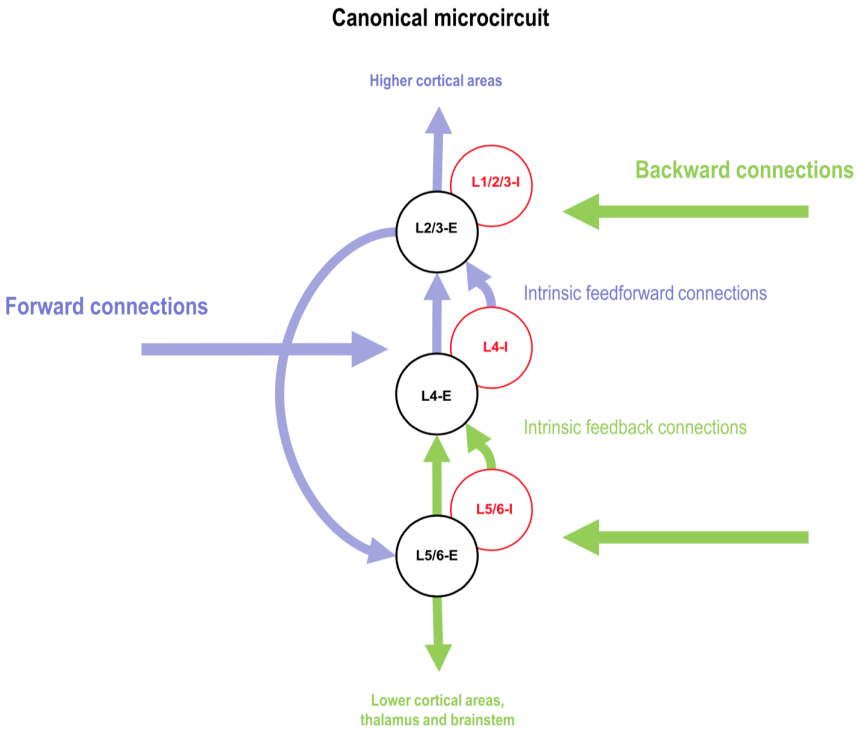

Figure 4.2: Connectivity diagram for a cortical microcircuit. Reproduced from [2].

4.2.2.2 Interneurons

Cortical interneurons are categorized into three classes based on gene expression:

- **Pvalb**
- **Sst**
- **Htr3a**

Interneurons contain many subclasses, which inhibit various local ECs and other interneurons. These inhibitory circuits may mediate diverse control of cortical processing during behavior, at least partially contributing to the quantitative differences between cortical circuits that perform different computations, yet have similar qualitative architectures.

4.3 Brain Structures

4.4 Canonical Neural Computations

4.4.1 Linear Weighting

4.4.2 Exponentiation

4.4.3 Normalization

[3] reviews various forms of normalization that appear in different brain areas and species. They define (divisive) normalization as *the computation in which the responses of neurons are divided by a common factor that typically includes the summed activity of a pool of neurons.* They point to examples of normalization in olfactory pathways, retinal contrast adaptation, V1, MT, V4, IT, medial superior temporal area (MST), auditory cortex, and lateral intraparietal area (LIP). They also point to a role of normalization in mediating attentional mechanisms, such as winner-take-all competition. The authors offer a number of reasons for the ubiquity of normalization, i.e. possible uses:

- **Maximizing Sensitivity**: adjusting the gain of neural responses to stay within the dynamic range.
- **Invariance**: discarding mean activation information and various irrelevant features
- **Neural Decoding**: treating a population of neurons as a normalized probability distribution
- **Stimuli Discrimination**: making discrimination easier for a linear classifier
- **Max Pooling**: biased and winner-take-all competition
- **Redundancy Reduction**: increasing the efficiency of neural representations by creating statistically independent activations.

Normalization is implemented in various circuits using various biophysical mechanisms, such as inhibition (polarization modulation), shunting inhibition (conductance modulation), and synaptic depression. Across many implementations, the overall computational scheme appears to rely on a "summation field" and a "suppression field," in which the activity of the summation field is divisively normalized according to the suppression field.

Chapter 5

Neural Networks

5.1 Basics

5.1.1 Linear Threshold Units

5.1.2 Multi-Layer Perceptrons

5.1.3 Backpropagation

Chapter 6

Graphical Models

6.1 Basics

6.2 Dynamical Directed Latent Variable Models

To model a sequence of T observations, $\mathbf{x}_{1:T}$, one can use a dynamical latent variable model, $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$, which models the joint distribution between $\mathbf{x}_{1:T}$ and a sequence of latent variables, $\mathbf{z}_{1:T}$, with parameters θ. Using a directed graphical model, $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$ can be factorized into conditional joint distributions, $p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})$, at each time step t, which are conditioned on all preceding variables:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) p_\theta(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}). \tag{6.1}$$

A plate notation diagram for the full graphical model is shown in Figure 6.1a. The distribution $p_\theta(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})$ is commonly referred to as the observation or measurement model, and $p_\theta(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})$ is the dynamics or transition model. Conditioning through a direct functional dependence on all past variables requires a method of storing and accessing this information, which becomes intractable, in terms of both memory and computation, as the sequence length grows. To overcome this intractability, the model is typically assumed to be Markovian, meaning that

- the states form a Markov sequence, $p_\theta(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) = p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1})$, and
- the observations are conditionally independent of all other latent variables and observations given the current latent variable, $p_\theta(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) = p_\theta(\mathbf{x}_t|\mathbf{z}_t)$.

In other words, it is assumed that the latent variable at a particular time step contains all of the information necessary to summarize the corresponding observation and all future and past latent variables. Eq. 6.1 then becomes:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{z}_{t-1}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{z}_t) p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}). \tag{6.2}$$

A plate notation diagram for the updated model is shown in Figure 6.1b.

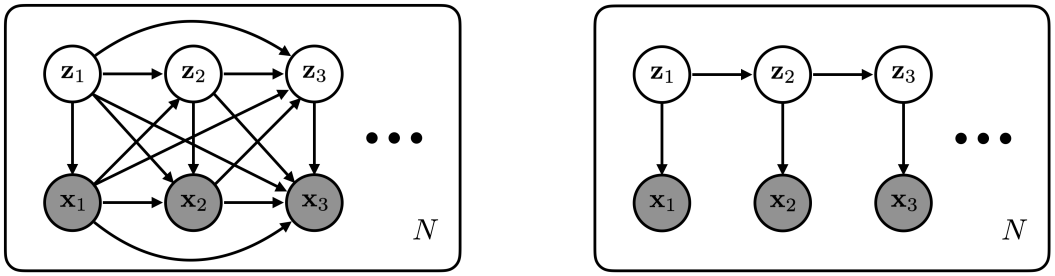

Figure 6.1: Plate notation diagrams for dynamical latent variable models with (a) all temporal dependencies and (b) the Markov assumption. Plates denote that there are N example sequences. Parameters θ have been omitted for clarity.

6.2.1 Bayesian Filtering

Dynamical directed latent variable models naturally lend themselves to a *filtering* setting [4], in which one computes the marginal posterior distribution, $p(\mathbf{z}_t|\mathbf{x}_{1:t})$, at each time step. This corresponds to estimating the current state of the system given all past observations. Under the Markov assumption (eq. 6.2), we start with a latent prior, $p_\theta(\mathbf{z}_1)$, and compute the marginal posterior at some time step t by recursively applying the following steps:

1. **Prediction**: use the Chapman-Kolmogorov equation to form a prior (i.e. prediction) on the latent variable at the next time step:

$$p(\mathbf{z}_t|\mathbf{x}_{1:t-1}) = \int p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}) p_\theta(\mathbf{z}_{t-1}|\mathbf{x}_{1:t-1}) d\mathbf{z}_{t-1}. \tag{6.3}$$

2. **Update**: use the prediction, along with Bayes' rule, to update the marginal posterior:

$$p(\mathbf{z}_t|\mathbf{x}_{1:t}) = \frac{p_\theta(\mathbf{x}_t|\mathbf{z}_t) p_\theta(\mathbf{z}_t|\mathbf{x}_{1:t-1})}{p_\theta(\mathbf{x}_{1:t})}, \tag{6.4}$$

   where $p_\theta(\mathbf{x}_{1:t})$ is calculated as

$$p_\theta(\mathbf{x}_{1:t}) = \int p_\theta(\mathbf{x}_t|\mathbf{z}_t) p_\theta(\mathbf{z}_t|\mathbf{x}_{1:t-1}) d\mathbf{z}_t. \tag{6.5}$$

When $p_\theta(\mathbf{x}_t|\mathbf{z}_t)$ and $p_\theta(\mathbf{z}_t|\mathbf{x}_{1:t-1})$ are parameterized as linear functions that take a Gaussian form, the resulting model is referred to as a Kalman filter [5]. However, in general, the prediction and update steps are both intractable, as they involve integrating over the space of latent variables. To overcome these intractabilities, one must turn to forms of approximate inference (see Section 7.3).

6.2.2 Beyond the Markov Assumption

While drastically simplifying the model, the Markov assumption is often invalid in practical settings, where one commonly encounters partial observations (i.e. hidden information) or correlated observation noise across time. In such cases, the "memoryless" or "static world" properties of the Markov assumption no longer hold, requiring that we expand the model to condition on past variables. This can be accomplished through

- conditioning on a concatenation of variables containing the recent trajectory,
- memory mechanisms that selectively store state information [6, 7, 8],
- modeling the observations and latent variables in generalized coordinates [9].

6.3 Transformation of Variables

Real-valued non-volume preserving (Real NVP) transformations provide a tractable method of performing learning and inference in a non-linear latent variable model [10]. With a bijective generative mapping g, latent variable $\mathbf{z} \sim p_z(\mathbf{z})$, and observation $\mathbf{x} \sim p_x(\mathbf{x})$, the density over observations can be modeled using the transformation of variables formula:

$$p_x(\mathbf{x}) = p_z(\mathbf{z}) \left| \det \left(\frac{\partial g(\mathbf{z})}{\partial \mathbf{z}^T} \right) \right|^{-1}. \tag{6.6}$$

Using the bijection $f = g^{-1}$, equation 6.6 can also be written as

$$p_x(\mathbf{x}) = p_z(f(\mathbf{x})) \left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^T} \right) \right| \tag{6.7}$$

$$\log\left(p_x(\mathbf{x})\right) = \log\left(p_z(f(\mathbf{x}))\right) + \log\left(\left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^T} \right) \right| \right). \tag{6.8}$$

Because the mapping between $\mathbf{x}$ and $\mathbf{z}$ is invertible, we can perform exact inference and evaluation with the model. Also note that evaluation of the log-likelihood, $\log\left(p_x(\mathbf{x})\right)$, does not require a fixed-form reconstruction cost, such as L^2 distance, because the terms on the right side of eq. 6.8 can be computed using the data, mapping, and prior. Inference and generation procedures are shown for a simple two dimensional example in Figure 6.2.

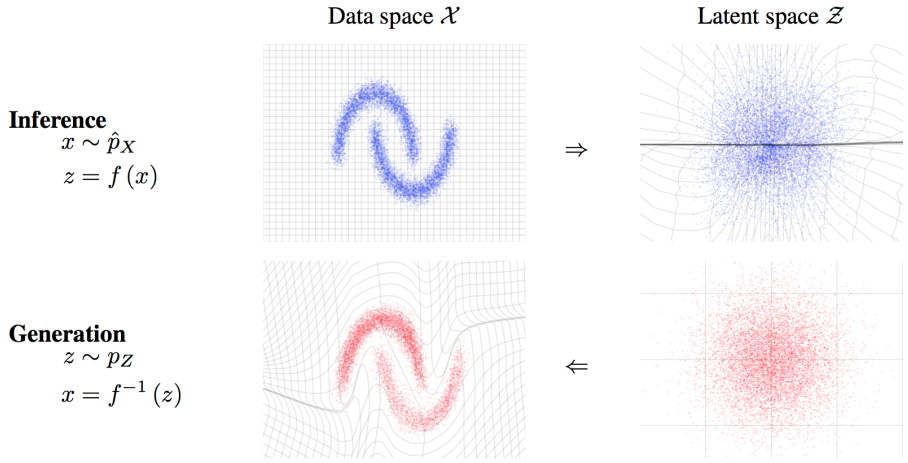

Figure 6.2: Inference and generation through transformation of variables in 2D. The invertible mapping between the observed and latent variables warps the respective spaces. Reproduced from [10].

In general, computing eq. 6.8 is intractable for high-dimensional variables, stemming from computing the log determinant of the Jacobian matrix. However, by carefully designing

the mapping, we can achieve a lower triangular Jacobian matrix, the determinant of which is simply the product of the diagonal elements. Given a D dimensional input $\mathbf{x}$ and $d < D$, an *affine coupling layer* with output $\mathbf{y}$ is defined as

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}, \tag{6.9}$$

$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d}), \tag{6.10}$$

where $s(\mathbf{x}_{1:d})$ and $t(\mathbf{x}_{1:d})$ are respectively *scale* and *translation* functions from $\mathbb{R}^d \to \mathbb{R}^{D-d}$. The Jacobian of this transformation is

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}^T} = \begin{bmatrix} \mathbf{I}_d & 0 \\ \frac{\partial \mathbf{y}_{d+1:D}}{\partial \mathbf{x}_{1:d}^T} & \operatorname{diag}\left(\exp\left(s(\mathbf{x}_{1:d})\right)\right) \end{bmatrix}. \tag{6.11}$$

Therefore, the determinant of the Jacobian can be easily computed as

$$\det\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}^T}\right) = \exp\left(\sum_j s(\mathbf{x}_{1:d})_j\right). \tag{6.12}$$

Note that s and t can be any arbitrary functions, as only their output values enter into inference and evaluation. Furthermore, the mapping in eqs. 6.9 and 6.10 can be inverted without computing the inverse of s or t:

$$\mathbf{x}_{1:d} = \mathbf{y}_{1:d}, \tag{6.13}$$

$$\mathbf{x}_{d+1:D} = (\mathbf{y}_{d+1:D} - t(\mathbf{x}_{1:d})) \odot \exp(-s(\mathbf{x}_{1:d})). \tag{6.14}$$

By composing multiple coupling layers, one can construct more expressive transformations. Successive coupling layers alternate between transforming different subsets of the input dimensions. The determinants of the Jacobians, as well as the inverse mappings, can still be computed easily. The authors of [10] use checkerboard and channel-wise masked convolutional networks to implement s and t, with a hierarchy of resolutions. Other possible transformations are possible instead of coupling layers, such as normalizing and inverse auto-regressive flow.

NICE: [11]

Chapter 7

Variational Inference

7.1 Inference as Optimization: Deriving the ELBO

Consider a model, $p(\mathbf{x}, \mathbf{z})$, that specifies a joint distribution over a latent variable $\mathbf{z}$ and observed variable $\mathbf{x}$. To infer the posterior distribution of $\mathbf{z}$ given $\mathbf{x}$, we can use the definition of conditional probability:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})}, \tag{7.1}$$

where the marginal likelihood (a.k.a. model evidence or partition function) $p(\mathbf{x})$ is calculated by marginalizing over the latent state space:

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}. \tag{7.2}$$

For large latent spaces and/or complicated models, performing the integration in eq. 7.2 is computationally intractable.

Variational inference transforms this intractable integration problem into an *optimization* problem by introducing an approximate posterior distribution, $q(\mathbf{z}|\mathbf{x})$, typically chosen from some simple family of distributions, such as independent Gaussians. We attempt to make $q(\mathbf{z}|\mathbf{x})$ as "close" as possible to $p(\mathbf{z}|\mathbf{x})$ by minimizing $D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$. Notice that the word close is in quotations because KL divergence is not a true distance measure; it is not symmetric. When the (non-negative) KL divergence between these distributions is zero, we recover the true posterior, $p(\mathbf{z}|\mathbf{x})$. We cannot evaluate $D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$ directly, as it includes the true posterior, which we cannot tractably compute. Instead, we will maximize a lower bound on $p(\mathbf{x})$, which will have the effect of minimizing $D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$, which we will now show. We start from the definition of KL divergence:

$$D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})}\right], \tag{7.3}$$

$$D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log q(\mathbf{z}|\mathbf{x})\right] - \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{z}|\mathbf{x})\right]. \tag{7.4}$$

Plugging in the definition of conditional probability into the second term yields:

$$D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log q(\mathbf{z}|\mathbf{x})\right] - \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})}\right], \tag{7.5}$$

which can be expanded into

$$D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log q(\mathbf{z}|\mathbf{x})\right] - \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x},\mathbf{z})\right] + \log p(\mathbf{x}). \tag{7.6}$$

Now, we'll define the following quantity, which we refer to as the *evidence lower bound (ELBO)* or *variational lower bound* or *negative free energy*:

$$\boxed{\mathcal{L} \equiv \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right]} \tag{7.7}$$

Plugging this definition back into eq. 7.6, we get

$$D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = \log p(\mathbf{x}) - \mathcal{L}. \tag{7.8}$$

Rearranging terms, we see

$$\log p(\mathbf{x}) = \mathcal{L} + D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})). \tag{7.9}$$

Since $D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$ is non-negative, $\mathcal{L}$ is a lower bound on $\log p(\mathbf{x})$. Further, since $\log p(\mathbf{x})$ is not dependent on $q(\mathbf{z}|\mathbf{x})$, we see that maximizing $\mathcal{L}$ with respect to $q(\mathbf{z}|\mathbf{x})$ must minimize $D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$. Thus, we can approximate the true posterior by optimizing $\mathcal{L}$ with respect to $q(\mathbf{z}|\mathbf{x})$.

Note that the ELBO can also be written in the following form:

$$\mathcal{L} = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right] \tag{7.10}$$

$$\mathcal{L} = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right] \tag{7.11}$$

$$\mathcal{L} = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] - D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{7.12}$$

This highlights that the ELBO specifies the optimal $q(\mathbf{z}|\mathbf{x})$ by trading off between representing the input, through the first term, and agreeing with the prior on the latent variables, through the second term. In other words, the first term attempts to fit the data, while the second term regularizes the representation.

7.2 Normalizing Flows

A common drawback of variational inference is that it is typically restricted to families of approximate posterior densities, e.g. factorized Gaussians. *Normalizing flows* [12] is a method for fitting flexible approximate posterior densities. It uses a sequence of invertible mappings, called a 'flow,' to transform simple posterior densities into arbitrary, flexible forms. If we start with a variable $\mathbf{z}$, drawn from a distribution $q(\mathbf{z})$, and apply an invertible, smooth mapping $f(\mathbf{z})$, then the change of variables formula allows us to write

$$q(\mathbf{z}') = q(\mathbf{z})\det\left|\frac{\partial f^{-1}}{\partial \mathbf{z}'}\right| = q(\mathbf{z})\det\left|\frac{\partial f}{\partial \mathbf{z}}\right|^{-1}. \tag{7.13}$$

Note that the mapping, f, can also depend on other inputs, such as observed variables $\mathbf{x}$ or other auxiliary inputs. By composing multiple mappings, we can construct arbitrarily

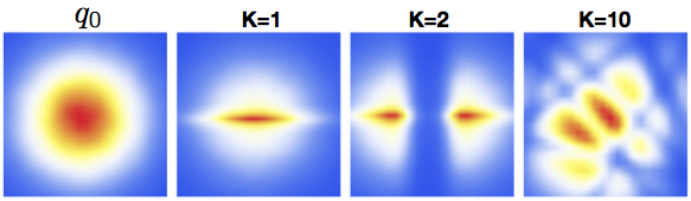

Figure 7.1: Normalizing flows applied to a 2D isotropic Gaussian, using transformations of the form given in eq. 7.15. Increasing the flow length K allows for more flexible approximate posterior distributions. Reproduced from [12].

complex densities. Thus, starting from an initial distribution of $q_0(\mathbf{z}_0)$, a flow of length K can be computed as

$$\log q_K(\mathbf{z}_K) = \log q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \log \det \left| \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|. \tag{7.14}$$

This sequence can be interpreted as performing expansions and contractions on the initial density to construct a more flexible form.

To parameterize the mappings, [12] use transformations of the form

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^\intercal \mathbf{z} + b) \tag{7.15}$$

where $\mathbf{u}$, $\mathbf{w}$, and b are learned parameters and $h(\cdot)$ is a non-linear function. This class of normalizing flows is referred to as *planar flows*. The determinant of the Jacobian of this transformation is

$$\det \left| \frac{\partial f}{\partial \mathbf{z}} \right| = \left| \det \left(\mathbf{I} + \mathbf{u}(h'(\mathbf{w}^\intercal \mathbf{z} + b)\mathbf{w})^\intercal \right) \right| = \left| 1 + \mathbf{u}^\intercal h'(\mathbf{w}^\intercal \mathbf{z} + b)\mathbf{w} \right| \tag{7.16}$$

Finally, putting everything together, we can use normalizing flows to draw samples from the final, more flexible approximate posterior, $q(\mathbf{z}|\mathbf{x}) = q_K(\mathbf{z}_K)$. The variational lower bound becomes

$$\mathcal{L} = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z},\mathbf{x})} \left[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}|\mathbf{x}) \right] \tag{7.17}$$

$$\mathcal{L} = \mathbb{E}_{\mathbf{z}_0 \sim q(\mathbf{z}_0,\mathbf{x})} \left[\log p(\mathbf{x}, \mathbf{z}_K) - \log q(\mathbf{z}_0|\mathbf{x}) + \sum_{k=1}^{K} \log \left| 1 + \mathbf{u}_k^\intercal h_k'(\mathbf{w}_k^\intercal \mathbf{z}_{k-1} + b_k)\mathbf{w}_k \right| \right] \tag{7.18}$$

Note that in going to eq. 7.17 to eq. 7.18 we have used the *law of the unconscious statistician (LOTUS)*, which allows one to take expectations of a transformed variable (or any function thereof) using the original variable. If $\mathbf{z}' = f(\mathbf{z})$ and g is some arbitrary function of $\mathbf{z}'$, then this can be expressed as

$$\mathbb{E}_{\mathbf{z}' \sim q(\mathbf{z}')} \left[g(\mathbf{z}') \right] = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z})} \left[g(f(\mathbf{z})) \right]. \tag{7.19}$$

Thus, we can take expectations with respect to the final density, i.e. the approximate posterior, using samples drawn from the initial density. In an amortized inference setting, the inference model can output the parameters of the initial distribution as well as parameters for each stage of the flow. The flow parameters could also be learned as global parameters.

Another class of normalizing flows is *inverse auto-regressive flow (IAF)* [13]. These transformations take inspiration from auto-regressive models, which factor joint probability distributions into a series of conditional distributions using the chain rule of probability.

7.3 Variational Inference in Dynamical Models

We assume the dynamic directed latent variable model set-up given in Section 6.2. We can bound the marginal log-likelihood of the observation sequence by introducing an approximate posterior distribution, $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$, then using Jensen's inequality to write

$$\log p(\mathbf{x}_{1:T}) \geq \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left[\log \frac{p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})}{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\right] \equiv -\mathcal{F}_E, \tag{7.20}$$

where $\mathcal{F}_E$ is the total *variational free-energy*. For now, we assume the generative model takes the form given by eq. 6.1, and the approximate posterior takes a similar form:

$$q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}). \tag{7.21}$$

In other words, we assume we are in the online setting, and therefore use a *filtering* approximate posterior that does not condition on future observations or latent variables. Using eqs. 6.1 and 7.21, we can rewrite eq. 7.20 as

$$\log p(\mathbf{x}_{1:T}) \geq \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left[\sum_{t=1}^{T} \log \frac{p(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})p(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})}{q(\mathbf{z}_t|\mathbf{x}_{1:t}, \mathbf{z}_{1:t-1})}\right]. \tag{7.22}$$

Defining the term

$$C_t \equiv \log \frac{p(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})p(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})}{q(\mathbf{z}_t|\mathbf{x}_{1:t}, \mathbf{z}_{1:t-1})}, \tag{7.23}$$

then

$$\log p(\mathbf{x}_{1:T}) \geq \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left[\sum_{t=1}^{T} C_t\right] \tag{7.24}$$

$$= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x}_1)}\mathbb{E}_{q(\mathbf{z}_2|\mathbf{x}_{1:2}, \mathbf{z}_1)} \cdots \mathbb{E}_{q(\mathbf{z}_T|\mathbf{x}_{1:T}, \mathbf{z}_{1:T-1})}\left[\sum_{t=1}^{T} C_t\right]. \tag{7.25}$$

There are T terms within the sum, but because we do not condition on future variables, each C_t only depends on the expectations up to time t. This allows us to write:

$$\begin{aligned}
\log p(\mathbf{x}_{1:T}) \geq\ & \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x}_1)}\left[C_1\right] \\
& + \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x}_1)}\mathbb{E}_{q(\mathbf{z}_2|\mathbf{z}_1,\mathbf{x}_{1:2})}\left[C_2\right] \\
& + \dots \\
& + \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x}_1)}\mathbb{E}_{q(\mathbf{z}_2|\mathbf{z}_1,\mathbf{x}_{1:2})}\cdots\mathbb{E}_{q(\mathbf{z}_T|\mathbf{z}_{1:T-1},\mathbf{x}_{1:T})}\left[C_T\right] && (7.26) \\
& = \sum_{t=1}^{T}\mathbb{E}_{q(\mathbf{z}_{1:t}|\mathbf{x}_{1:t})}\left[C_t\right] && (7.27) \\
& = \sum_{t=1}^{T}\mathbb{E}_{\prod_{\tau=1}^{t}q(\mathbf{z}_\tau|\mathbf{x}_{1:\tau},\mathbf{z}_{1:\tau-1})}\left[C_t\right] && (7.28) \\
& = \sum_{t=1}^{T}\mathbb{E}_{\prod_{\tau=1}^{t-1}q(\mathbf{z}_\tau|\mathbf{x}_{1:\tau},\mathbf{z}_{1:\tau-1})}\left[\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{1:t},\mathbf{z}_{1:t-1})}\left[C_t\right]\right] && (7.29)
\end{aligned}$$

The total variational free-energy is thus the sum of per-time-step variational free-energies, with expectations taken w.r.t. all past latent variables:

$$\mathcal{F}_E = -\sum_{t=1}^{T}\mathbb{E}_{\prod_{\tau=1}^{t-1}q(\mathbf{z}_\tau|\mathbf{x}_{1:\tau},\mathbf{z}_{1:\tau-1})}\left[\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{1:t},\mathbf{z}_{1:t-1})}\left[C_t\right]\right]. \tag{7.30}$$

Evaluating these outer expectations becomes computationally intractable as the sequence length grows. When approximating each of the T expectations with m Monte Carlo samples, evaluating the summation requires drawing $\mathcal{O}(m^T)$ samples. For $m > 1$, the number of samples, and therefore the amount of computation, blows up as $T \to \infty$. Thus, to retain tractability, we can set $m = 1$ for expectations over all past time steps, thereby evaluating the path summation of the per time step variational free-energies, defined as the *variational free-action*, $\mathcal{F}_A$:

$$\mathcal{F}_A \equiv -\sum_{t=1}^{T}\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{1:t},\mathbf{z}_{1:t-1})}\left[C_t\right]\Big|_{\mathbf{z}_{1:t-1}} \tag{7.31}$$

where $\mathbf{z}_{1:t-1}$ is the sampled trajectory of the past latent variables. The free-action evaluates a single path through the latent space, whereas the free-energy evaluates all possible paths. However, the free-action still provides a lower bound on the marginal log-likelihood of the sequence. Need to show this...

Chapter 8

Representation Learning

8.1 Introduction

Representation learning is the process of taking data, $\mathbf{x}$, and forming a representation, $\mathbf{z}$, that captures the "meaningful" information contained in the data. How do we define what constitutes meaningful information? This question exposes a fundamental aspect of representation learning: it is inherently ill-defined. There is no way in which to truly assess the quality of a representation, i.e. which aspects of the data are meaningful, without defining a *task*, $\mathbf{y}$, which one hopes to accomplish with the representation. In Section 8.2, we discuss the information bottleneck theory [14], an information theoretic framework for representation learning. Then, in Section 8.3, we discuss learning representations in the absence of a clearly defined task, where we outline a variety of heuristics and priors.

8.2 The Information Bottleneck

The **information bottleneck** is a compression method that squeezes the information that an input variable $\mathbf{x} \in X$ (data) contains about some other variable $\mathbf{y} \in Y$ (task) through a 'bottleneck' formed by a representation $\mathbf{z} \in Z$ [14]. The approach is motivated by rate distortion theory [15], in which the quality of a representation is determined by

1. the *information rate*, i.e. the average amount of information required to specify $\mathbf{z}$ without confusion

2. the *distortion function* d, which implicitly specifies the most relevant aspects of the input X.

A good representation is thus one that maximally compresses the meaningful information in the data. We could solve for the representation by introducing a Lagrange multiplier β and use variational methods to solve the following equation for $p(\mathbf{z}|\mathbf{x})$:

$$\mathcal{F}[p(\mathbf{z}|\mathbf{x})] = I(Z;X) + \beta\langle d(\mathbf{z},\mathbf{x})\rangle_{p(\mathbf{z}|\mathbf{x})}, \tag{8.1}$$

where I denotes the mutual information. However, we rarely have access to the correct distortion function. The information bottleneck method instead uses another variable Y to specify the relevant aspects of X. The tradeoff between compression and information preservation is expressed by the functional

$$\mathcal{L}[p(\mathbf{z}|\mathbf{x})] = I(Z;X) - \beta I(Z;Y). \tag{8.2}$$

An exact formal solution can be derived for this functional. One can solve the free-energy functional

$$\mathcal{F}[p(\mathbf{z}|\mathbf{x}), p(\mathbf{z}), p(\mathbf{y}|\mathbf{z})] = I(Z;X) + \beta\langle D_{KL}(p(\mathbf{y}|\mathbf{x})||p(\mathbf{y}|\mathbf{z}))\rangle_{p(\mathbf{x},\mathbf{z})} \tag{8.3}$$

for $p(\mathbf{z}|\mathbf{x})$, $p(\mathbf{z})$, and $p(\mathbf{y}|\mathbf{z})$, using a deterministic annealing approach for β. Building on this work, in [16], an ideal representation is defined as one that is:

1. **sufficient**: capturing the information contained in $\mathbf{x}$ that is sufficient to estimate $\mathbf{y}$, i.e. $I(Y;Z) = I(Y;X)$.
2. **minimal**: retains as little information about $\mathbf{x}$ as possible, making the task easier, i.e. $I(Z;X)$ is minimized.
3. **invariant**: nuisances n have no effect on the representation, which helps to prevent overfitting i.e. $I(Z;n) = 0$.
4. **disentangled**: the total correlation, defined as the dissimilarity between the representation and the product of its marginals, $TC(\mathbf{z}) \equiv D_{KL}(p(\mathbf{z})||p(\mathbf{z}_1)p(\mathbf{z}_2)\dots p(\mathbf{z}_L))$, is minimized.

It is shown that only (1) and (2) need to be enforced, as (3) and (4) naturally follow.

8.3 Representation Learning Without a Task

Often, we do not have a well-defined task, however, we would still like to learn a representation that captures aspects of the data that will facilitate rapid transfer learning to future tasks. This is referred to as *unsupervised learning*. The lack of task implies that we no longer have a measure of sufficiency, meaning that we must define an alternative quantity to optimize.

[17] identify an alternative set of *priors* for ideal representations:

1. **smoothness**: the representation should be a *smooth* mapping of the data, i.e. if $\mathbf{x}^{(1)} \approx \mathbf{x}^{(2)}$, then $\mathbf{z}^{(1)} \approx \mathbf{z}^{(2)}$.
2. **multiple explanatory factors**: the representation is made up of a set of disentangled factors that explain the data. This is also sometimes referred to as a distributed representation.
3. **hierarchical organization**: the representation is arranged in a hierarchy from lower level to more abstract factors.
4. **semi-supervised**: the factors necessary for modeling the data $\mathbf{x}$ are useful for modeling some task labels $\mathbf{y}$. This is related to sufficiency mentioned above, but does not place importance on the task first and foremost.
5. **shared factors across tasks**: explanatory factors within the representation should be useful more multiple tasks
6. **manifolds**: probability mass should be concentrated near regions that are much smaller than the entire dimensionality of the data. The data does not occupy the input space uniformly.

7. **natural clustering**: different explanatory factors may lie on different manifolds.

8. **temporal and spatial coherence**: inputs that are nearby in space and time should be similar (although this can be violated)

9. **sparsity**: only a small number of factors within the representation should be active for any particular data example.

10. **simplicity of factor dependencies**: the higher level factors should have simple (i.e. linear) dependencies.

8.4 Disentangled Representations

Define disentanglement. Why are disentangled representations helpful? Trade off between disentanglement and faithfully representing the data. How do we learn disentangled representations, while at the same time, respecting the structure of the latent space? Importance of priors.

Two random variables are **independent** if their joint probability can be expressed as the product of their marginals:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}), \tag{8.4}$$

and they are **conditionally independent** if their conditional joint probability can be expressed as

$$p(\mathbf{x}, \mathbf{y}|\mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{y}|\mathbf{z}). \tag{8.5}$$

Independence is related to covariance, but is a stronger property. Two variables that are independent have zero covariance and two variables that have non-zero covariance are dependent. Zero covariance implies that the variables have no *linear* dependence. Independence implies that the variables also have no *non-linear* dependence. That is, it is possible for two dependent variables to have zero covariance.

8.5 Other

Notes:

[18] propose using a cross-covariance penalty term in the setting of semi-supervised learning of the form:

$$C(\hat{\mathbf{y}}^{1\ldots N}, \mathbf{z}^{1\ldots N}) = \frac{1}{2}\sum_{ij}\left[\frac{1}{N}\sum_{n}(\hat{y}_i^n - \bar{\hat{y}}_i)(z_j^n - \bar{z}_j)\right]^2, \tag{8.6}$$

where $\hat{\mathbf{y}}$ is a vector of (one-hot) inferred labels and $\mathbf{z}$ is a latent representation. N is the batch size.

[19] propose to use a regularization weight in a VAE setting to "upweight" the amount of regularization as a means of promoting **redundancy reduction and independence** among the latent representation. They also argue for the importance of **dense sampling of the (continuous) data manifold** for disentanglement. Sparse sampling of the data manifold results in ambiguity is the manifold interpretation, requiring additional supervision for disentanglement.

[20] use supervision to impose structure on part of the latent representation in a VAE.

Dropout [21] is a technique for preventing "fragile coadaptation" between units within a representation, effectively enforcing that they represent different (independent) quantities. This technique also results in redundancy within the representation.

Chapter 9

Reinforcement Learning

9.1 The RL Problem Setting

TODO: image of agent environment interaction.

9.1.1 The Basics

Reinforcement learning involves an **agent** interacting within an **environment**. The agent selects actions, and the environment responds with new states and rewards. Sutton & Barto [22] identify four additional sub-elements to reinforcement learning systems:

- a **policy**,
- a **reward signal**,
- a **value function**,
- and, optionally, a **model** of the environment.

A *policy* defines a mapping from (perceived) states to output actions to be taken. A *reward signal* is a real-valued number that the agent receives from the environment. The agent tries to maximize its total reward, which is referred to as value. The *value function* specifies the expected value, starting from a particular state. Thus, it is ultimately value that we try to optimize. Finally, a *model* of the environment is the agent's simulation of the environment, allowing the agent to plan actions and predict outcomes (states, rewards, etc.).

The boundary between agent and environment is often not clearly defined. Sutton & Barto place the agent-environment boundary at the limit of the agent's control, not at the physical boundary. For instance, the limbs of the agent, its internal energy reserves, and even the reward mechanism are considered to be part of the environment, since the agent does not have absolute control over these aspects.

We assume that time unfolds in a series of discrete time-steps, $t = 1, 2, 3, \ldots$ At each time step, the agent is in some state $S_t \in \mathcal{S}$ and chooses some action $A_t \in \mathcal{A}(S_t)$. Here, $\mathcal{S}$ denotes the space of possible states and $\mathcal{A}(S_t)$ denotes the space of possible actions from state S_t. The action is chosen based on the agent's policy $\pi_t(a|s)$, which probabilistically maps states $S_t = s$ to actions $A_t = a$. At the next time step, the agent receives a real-valued reward, $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$, and arrives in the next state, S_{t+1}.

The **return**, G_t, is the (discounted) sum of rewards starting at time t. For **episodic tasks**, in which the sequence has a finite length, T, this is defined as

$$G_t \triangleq R_{t+1} + R_{t+2} + \cdots + R_T = \sum_{k=0}^{T-t-1} R_{t+k+1}, \tag{9.1}$$

However, for **continuing tasks**, in which the time sequence length can be infinite, we include a **discount rate**, $\gamma \in [0, 1]$, to prevent the return from going to infinity:

$$G_t \triangleq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{9.2}$$

The **value function** or **state-value function**, $v_\pi(s)$, is defined as the expected return from being in state s when using policy π:

$$v_\pi(s) \triangleq \mathbb{E}_\pi \left[G_t | S_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right]. \tag{9.3}$$

Note that here we are assuming that the task respects the Markov property; the current state, along with the policy, contains all of the information to determine the value function. We can also define an **action-value function**, $q_\pi(s, a)$, which specifies expected returns for each action a taken from state s using policy π:

$$q_\pi(s, a) \triangleq \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right]. \tag{9.4}$$

Using the definition of the value function, we can derive a recursive relationship known as the **Bellman equation**. Starting from eq. 9.3:

$$\begin{aligned}
v_\pi(s) &\triangleq \mathbb{E}_\pi \left[G_t | S_t = s \right] && (9.5)\\
&= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right] && (9.6)\\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \middle| S_t = s \right] && (9.7)\\
&= \sum_{a,s',r} \pi(a|s) p(s', r|s, a) \left[r + \gamma \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \middle| S_{t+1} = s' \right] \right] && (9.8)\\
&= \sum_{a,s',r} \pi(a|s) p(s', r|s, a) \left[r + \gamma v_\pi(s') \right]. && (9.9)
\end{aligned}$$

To go from eq. 9.6 to eq. 9.7, we split the first term out of the sum. To arrive at 9.8, we move the expectation to the future summation, re-expressing the expectation over the first step using summations to weight each return by the appropriate probability of the action a, state s', and reward r. Finally, to arrive at 9.9, we notice that the expectation is the value of state s' under our policy, i.e. $v_\pi(s')$. Thus, we see that the value function has the recursive definition:

$$v_\pi(s) = \sum_{a,s',r} \pi(a|s) p(s', r|s, a) \left[r + \gamma v_\pi(s') \right]. \tag{9.10}$$

A similar recursive definition can be derived for $q_\pi(s, a)$.

The **optimal value function** $v_*(s)$

9.1.2 Summary of Notation

9.2 Value Based Methods

9.2.1 Multi-Armed Bandits

The multi-armed bandit problem is an example of a *non-associative setting*, in which the agent only learns to act in one situation. This simplifies much of the reinforcement learning problem. In particular, the **multi-armed bandit** problem involves a situation with k possible actions (sometimes referred to as arms, in analogy to slot machines), each with an associated value. The value of action a is denoted as $q_*(a)$, the **action-value function**. The interaction between the agent and the environment consists of a series of episodes, each of duration 1 time step. At time step t, the agent takes **action** A_t and receives **reward** R_t. Thus, for some action $A_t = a$, the action-value function is defined as

$$q_*(a) \triangleq \mathbb{E}\left[R_t | A_t = a\right]. \tag{9.11}$$

Note that rewards can be a stochastic function of the action chosen. $q_*(a)$ encodes the *mean* reward of choosing action a. In multi-armed bandit problems, the action-value function is initially unknown. To optimize total reward, we must estimate the action-value function to find the action or actions with the highest reward. Selecting this action will then optimize reward. We refer to the **action-value estimates** at time step t as $Q_t(a)$.

At any time step t, we can either **exploit** our current knowledge by choosing the action with the optimal action-value estimate, or we can **explore** other actions with non-optimal action-value estimates. Exploitation is a greedy process, which will likely yield higher rewards in the short-term. Exploration, on the other hand, may find better actions, which can later be exploited, allowing it to likely yield higher rewards in the long-term.

We can select actions according to different strategies. For instance, we could select the action with the optimal action-value estimate at each time step:

$$A_t = \text{argmax}_a Q_t(a) \tag{9.12}$$

However, this policy will never explore other actions, and may therefore perform poorly in the long run. Another strategy is to select the action with the optimal action-value estimate with probability $1 - \epsilon$ and select a random action with probability ϵ. This strategy is referred to as ϵ**-greedy**.

9.2.2 Dynamic Programming Methods

9.2.2.1 Policy Iteration

9.2.2.2 Value Iteration

9.2.3 Monte Carlo Methods

9.2.4 TD Learning

9.2.4.1 Q Learning

9.2.4.2 SARSA

9.3 Policy Gradient Methods

9.3.1 REINFORCE

9.3.2 Actor-Critic Methods

9.4 Intrinsic Motivation

Intrinsic motivation is to reinforcement learning as unsupervised learning is to supervised learning. By learning from general purpose signals that connect the perception-action loop, intrinsic motivation can allow an agent to learn how to perceive and act even in the presence of a sparse or non-existent reward signal. This can dramatically speed up learning when an environmental reward signal is encountered. In [23], the authors characterize intrinsic motivation using the concept of *collative variables* [24], which refer to various measures of subjective uncertainty, such as novelty, surprise, ambiguity, complexity, etc., all of which can be expressed in terms of information theoretic quantities. The computational landscape described in [23] consists of:

1. **Knowledge based models** involve comparisons between predicted sensorimotor states from an internal model and the actual states experienced
2. **Competence based models** involve comparisons between self-generated goals and the extent to which they are actually reached
3. **Morphological models** involve finding relationships between multiple sensorimotor channels which are not based on the agent's long-term knowledge or competence

A majority of intrinsic motivation approaches fall into the first category, which is the focus of this section. This includes measures such as Bayesian surprise [25], curiosity [26], prediction errors [27, 28], and information theoretic measures [29, 30, 31].

9.4.1 Empowerment

We can consider an agent's perception-action loop as a noisy communication channel: an agent performs actions, which can affect the environment and transmit information to the agent's sensory states. Ideally, an agent should be able to control and predict aspects of its environment, effectively communicating with itself through its own actions. **Empowerment** formalizes this notion, corresponding to the maximum mutual information (i.e.

channel capacity) between an agent's actions and its sensory states [29]. Denoting a sequence of n actions starting at time step t as $a_t^n \in A_t^n$ and the state at time t as $S_t = s_t$, empowerment is expressed as

$$\mathcal{E}(s_t) \equiv \max_{p(a_t^n|s_t)} I(A_t^n; S_{t+n}). \tag{9.13}$$

That is, given the distribution of all possible n-step action sequences from the current state, empowerment is the maximum mutual information between an action sequence and the agent's corresponding sensory state n time steps later. The conditional distribution $p(a_t^n|s_t)$ is the agent's policy. From the definition of mutual information, we can also write this as:

$$\mathcal{E}(s_t) = \max_{p(a_t^n|s_t)} \mathbb{E}_{p(a_t^n, s_{t+n}|s_t)} \left[\log \left(\frac{p(a_t^n, s_{t+n}|s_t)}{p(a_t^n|s_t)p(s_{t+n}|s_t)} \right) \right] \tag{9.14}$$

Note that empowerment is only concerned with the *capacity* to affect one's own sensory states, not the actual action sequence performed.

From eqs. 9.13 and 9.14, there are three methods by which an agent can increase empowerment:

1. attempt to reach states where $\mathcal{E}(s_t)$ is high, i.e. where $p(s_{t+n}|a_t^n, s_t)$ is less noisy,
2. modify one's sensors or perception model to improve empowerment, i.e. forming a better state representation where different environmental states can be discriminated,
3. modify one's actuators or policy model to improve empowerment, i.e. taking better actions that result in specific environmental states.

The first method operates over shorter time horizons and is similar to *inference*, in which an agent changes its current states (and/or actions). The second and third methods operate over longer time horizons and are similar to *learning*, in which an agent changes its model parameters.

9.4.1.1 Variational Lower Bound on Empowerment

Rewriting $p(a_t^n, s_{t+n}|s_t) = p(s_{t+n}|a_t^n, s_t)p(a_t^n|s_t)$, we see that the expectation in eq. 9.14 involves marginalizing over the environment dynamics $p(s_{t+n}|a_t^n, s_t)$. Even if this dynamics model were known, computing the mutual information would be intractable for long time horizons n or complex environments with many states and actions. Instead, we can use a variational distribution over actions, $q(a_t^n|s_{t+n}, s_t)$, to induce a lower bound on the mutual information, then maximize this lower bound [32]. The mutual information can be rewritten as

$$I(A_t^n; S_{t+n}) = H(A_t^n) - H(A_t^n|S_{t+n}). \tag{9.15}$$

The conditional entropy is bounded by

$$H(A_t^n|S_{t+n}) \leq -\mathbb{E}_{p(a_t^n, s_{t+n}|s_t)} \left[\log q(a_t^n|s_{t+n}, s_t) \right]. \tag{9.16}$$

Therefore, the mutual information is bounded by

$$I(A_t^n; S_{t+n}) \geq H(A_t^n) + \mathbb{E}_{p(a_t^n, s_{t+n}|s_t)} \left[\log q(a_t^n|s_{t+n}, s_t) \right], \tag{9.17}$$

$$I(A_t^n; S_{t+n}) \geq H(A_t^n) + \mathbb{E}_{p(s_{t+n}|a_t^n, s_t)p(a_t^n|s_t)} \left[\log q(a_t^n|s_{t+n}, s_t) \right]. \tag{9.18}$$

The set-up has an intuitive interpretation [32, 33]. The marginal action sequence distribution $p(a_t^n|s_t)$ can be regarded as a *source* or *exploration* distribution over which we draw actions, i.e. the policy. Upon selecting an action sequence, the distribution $p(s_{t+n}|a_t^n, s_t)$ can be regarded as an *encoder* or *transition* distribution that maps an action sequence to a future state. The variational distribution $q(a_t^n|s_{t+n}, s_t)$ then acts as a *decoder* or *planning* distribution, mapping some future state back to an action sequence.

Having derived a lower bound on the mutual information, empowerment is approximately calculated by maximizing over the parameters of the distributions $p(a_t^n|s_t)$ and $q(a_t^n|s_{t+n}, s_t)$:

$$\mathcal{E}(s_t) = \max_{p(a_t^n|s_t), q(a_t^n|s_{t+n}, s_t)} \Big(H(A_t^n) + \mathbb{E}_{p(a_t^n, s_{t+n}|s_t)} \left[\log q(a_t^n|s_{t+n}, s_t) \right] \Big) . \tag{9.19}$$

When performed using alternating optimization, this is referred to as the information maximization (IM) algorithm [32]. It is noted in [33] that eq. 9.19 can be difficult to optimize. The authors address this by placing a constraint on $H(A_t^n)$, essentially re-weighting the entropy term.

9.4.2 Uncertainty Motivation

9.4.3 Information Gain Motivation

9.4.4 Predictive Novelty Motivation

Predictive novelty motivation assigns an intrinsically motivated reward that is proportional to the agent's prediction error on an observed sensorimotor state. This can be expressed in probabilistic or non-probabilistic terms. The reward of the novel sensorimotor state will drive the agent to repeatedly explore that state, while at the same time, the agent's internal model improves to minimize prediction error, thereby reducing the novelty of the state. This approach, along with the option framework, is used in [28] to learn a hierarchical collection of skills, allowing the agent to more effectively optimize a sparse extrinsic reward signal. It should also be noted that one must be careful in assigning rewards to prediction errors, as extreme novelty (i.e. a random, unpredictable environment) should have an aversive effect on the agent.

9.4.5 Learning Progress Motivation

[34] introduce variational intrinsic control within the option framework.

Chapter 10

Predictive Coding

10.1 Introduction

Predictive coding, in its current form, was formalized by [35]. This theoretical neuroscience model posited that sensory processing is a filtering problem, in which latent states underlying the environmental stimuli are estimated according to their agreement with the observed input as well as prior beliefs about these quantities. This offered an explanation for "extra-classical" receptive field effects in visual processing, in which stimuli outside the receptive field of a particular cortical neuron are able to affect that neuron's activity. This was explained as the result top-down prior beliefs.

However, the notion of perception as a generative process dates back to at least [36]. *TODO: mention other early work before Rao and Ballard with similar ideas*

Predictive coding claims that sensory processing, i.e. perception, is fundamentally about constructing a generative model of the input sensory signal. To perform *inference* in this model (to perceive), the model uses its current estimate of the latent variables underlying the environment to generate reconstructions or predictions of the input. Using the residual (error) from this reconstruction or prediction, along with residuals from prior beliefs, the model updates its estimate of these latent variables. *Learning* corresponds to updating the parameters of the generative model to improve these overall residuals.

10.1.1 Literature Summary

[37] provides a survey of the area of predictive coding

[38, 39, 40] presents a free energy formulation of predictive coding that uses expectation maximization. Primarily focuses on the static setting. Puts forth a theory of cortical processing as carrying out predictive coding.

[41] presents *dynamic causal modeling.*

[42] presents *variational filtering.*

[43] presents *dynamic expectation maximization* (DEM).

[9] presents *generalized filtering*, which does not make the mean-field assumption and instead treats all unknown variables as conditionally dependent. This differs from variational filtering [42] in that it assumes the parameters and precisions also change with time, with a prior on smoothness of these changes.

[44] claims that attention can be formulated as the adjustment of prior precisions in the predictive coding model.

10.2 The Static Setting

10.2.1 With a free-form distribution

We start with the general presentation from [43]. We have some latent parameters ϑ, which through some model m, generate observations $\mathbf{x}$. Variational inference allows us to express the log-evidence as

$$\log p(\mathbf{x}|m) = D_{KL}(q(\vartheta)||p(\vartheta|\mathbf{x}, m)) + \mathcal{F}, \tag{10.1}$$

where $q(\vartheta)$ is an approximate posterior distribution, $p(\vartheta|\mathbf{x}, m)$ is the true posterior, and $\mathcal{F}$ is the (negative) free energy, which can be expressed as

$$\mathcal{F} \equiv \mathbb{E}_{q(\vartheta)} \left[\log p(\mathbf{x}, \vartheta) - \log q(\vartheta)\right]. \tag{10.2}$$

Because KL-divergence is non-negative, $\mathcal{F}$ is a lower bound on the log-evidence. This bound is tight if and only if $q(\vartheta)$ is the true posterior. That is, because the log-evidence does not depend on $q(\vartheta)$, we can minimize the KL-divergence term, thereby forming a closer approximation to the true posterior, by maximizing $\mathcal{F}$ w.r.t. $q(\vartheta)$.

Often, we assume that the approximate posterior fractorizes in some way,

$$q(\vartheta) = \prod_i q(\vartheta^i). \tag{10.3}$$

This factorization introduces independencies over states, parameters, noise variables, etc. With a free-form approximate posterior, the free energy is maximized w.r.t. $q(\vartheta^i)$ when

$$q(\vartheta^i) = \frac{1}{\mathcal{Z}^i} \exp(V(\vartheta^i)), \tag{10.4}$$

where the variational energy $V(\vartheta^i)$ is defined as

$$V(\vartheta^i) \equiv \mathbb{E}_{q(\vartheta^{\neg i})} \left[p(\mathbf{x}, \vartheta)\right], \tag{10.5}$$

and where $\vartheta^{\neg i}$ denotes the set of parameters not in the i^{th} partition. $\mathcal{Z}^i$ is the normalization factor for the i^{th} partition. Equation 10.4 can be shown as follows. The Fundamental Lemma of variational calculus states that $\mathcal{F}$ is maximized w.r.t. $q(\vartheta^i)$ if and only if

$$\delta_q(\vartheta^i)\mathcal{F} = 0 \Leftrightarrow \partial_{q(\vartheta^i)} f^i = 0 \tag{10.6}$$

where

$$\int f^i d\vartheta^i = \mathcal{F} \tag{10.7}$$

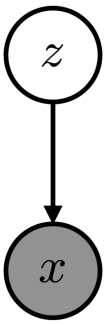

Figure 10.1: A graphical model denoting a latent variable model in which a single latent variable z generates a single observed variable x. In predictive coding, we typically assume that $p(z)$ and $p(x|z)$ are Gaussian densities, and the generative mapping from z to x is a non-linear function.

10.2.2 MAP State Estimation in a One–Level Model

Consider a situation in which the value of a single latent variable z must be inferred from a single observed variable x. This is represented by the graphical model in figure 10.1. Let g denote a non-linear function defining how z generates x. Assume that the generated output takes the form of a normal distribution with mean $\mu_x = g(z)$ and constant variance σ_x^2:

$$p(x|z) = \mathcal{N}(x; g(z), \sigma_x^2). \tag{10.8}$$

Recall that in one dimension, a normal distribution takes the form

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{10.9}$$

We will place a prior on z, which we also assume is a normal distribution with constant mean μ_p and constant variance σ_p^2:

$$p(z) = \mathcal{N}(z; \mu_p, \sigma_p^2). \tag{10.10}$$

To infer the posterior distribution of z, we can use Bayes' rule to invert the generative model:

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}, \tag{10.11}$$

where the marginal distribution in the denominator is given as

$$p(x) = \int p(x|z)p(z)dz. \tag{10.12}$$

In general, computing the exact posterior distribution is computationally intractable due to the integration over z in eq. 10.12. Instead, we'll resort to variational inference[1] to find the maximum (mode) of the posterior, otherwise known as the maximum a posteriori

[1]We are not actually using variational inference here, since the maximum of $p(z|x)$ must also be the maximum of $p(x, z)$ through the definition of conditional probability: $p(z|x) = \frac{p(x,z)}{p(x)}$, since $p(x)$ does not depend on the value of z.

or MAP estimate. This is the *most likely* estimate of the value of z. We will define our approximate distribution, a point mass estimate at μ_q, as $q(z|x) = \delta(z = \mu_q)$, with the MAP estimate denoted as $\hat{\mu}_q$.

We want to maximize the evidence lower bound (ELBO), $\mathcal{L}$:

$$\mathcal{L} = \mathbb{E}_{z\sim q(z|x)}\left[\log p(x,z) - \log q(z|x)\right] = \mathbb{E}_{z\sim q(z|x)}\left[\log p(x|z) + \log p(z) - \log q(z|x)\right] \tag{10.13}$$

$$\mathcal{L} = \log\left(\frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{(x-g(\mu_q))^2}{2\sigma_x^2}}\right) + \log\left(\frac{1}{\sqrt{2\pi\sigma_p^2}} e^{-\frac{(\mu_q-\mu_p)^2}{2\sigma_p^2}}\right) \tag{10.14}$$

$$\mathcal{L} = -\frac{1}{2}\log(2\pi\sigma_x^2) - \frac{(x-g(\mu_q))^2}{2\sigma_x^2} - \frac{1}{2}\log(2\pi\sigma_p^2) - \frac{(\mu_q-\mu_p)^2}{2\sigma_p^2} \tag{10.15}$$

$$\mathcal{L} = \frac{1}{2}\left(-\log(\sigma_x^2) - \frac{(x-g(\mu_q))^2}{\sigma_x^2} - \log(\sigma_p^2) - \frac{(\mu_q-\mu_p)^2}{\sigma_p^2}\right) + \text{const.} \tag{10.16}$$

In going from eq. 10.13 to eq. 10.14, we used the fact that $q(z|x)$ is a delta function, so the expectation becomes an evaluation at a single point, $z = \mu_q$. The expectation of $q(z|x)$ at this point is 1, so $\log q(z|x)$ evaluates to 0. To find the MAP estimate, we must solve the following optimization problem:

$$\hat{\mu}_q = \text{argmax}_{\mu_q}\mathcal{L} = \text{argmax}_{\mu_q}\frac{1}{2}\left(-\log(\sigma_x^2) - \frac{(x-g(\mu_q))^2}{\sigma_x^2} - \log(\sigma_p^2) - \frac{(\mu_q-\mu_p)^2}{\sigma_p^2}\right). \tag{10.17}$$

To use first-order optimization methods, we must find the partial derivative of $\mathcal{L}$ w.r.t. μ_q:

$$\frac{\partial\mathcal{L}}{\partial\mu_q} = \frac{x-g(\mu_q)}{\sigma_x^2}\frac{dg}{d\mu_q} + \frac{\mu_q-\mu_p}{\sigma_p^2} \tag{10.18}$$

We see that we have two terms: **the first term moves the estimate toward agreement with the observation**, and **the second term moves the estimate toward agreement with the prior**. Each of these terms are weighted by their corresponding variances. In other words, inference (i.e. finding the optimal estimate of μ_q) involves a weighted combination of *bottom-up* and *top-down* information. By repeatedly moving our estimate μ_q along this gradient, we can hopefully arrive at the MAP estimate $\hat{\mu}_q$. Note that we may not find the true value $\hat{\mu}_q$ if the optimization surface is non-convex.

We would like to extend this single dimensional model to handle latent and observed variables of arbitrary size. In this setting, $\mathbf{x}$ and $\mathbf{z}$ are now vectors. The conditional likelihood, $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_\mathbf{x} = g(\mathbf{z}), \boldsymbol{\Sigma}_\mathbf{x})$ is now a multi-variate Gaussian distribution. The general form for this distribution is

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\intercal\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}, \tag{10.19}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix, $|\boldsymbol{\Sigma}|$ is the determinant of $\boldsymbol{\Sigma}$, and n is the dimensionality of the vector $\mathbf{x}$. The prior on $\mathbf{z}$ is also a multi-variate Gaussian: $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$. The MAP estimate is now a vector, $\hat{\boldsymbol{\mu}}_q$, of the most likely estimate of $\boldsymbol{\mu}_q$. To find this estimate,

we must again optimize $\mathcal{L}$ w.r.t. $\boldsymbol{\mu}_q$. Repeating the steps above, the gradient, $\nabla_{\boldsymbol{\mu}_q}\mathcal{L}$, is given as:

$$\nabla_{\boldsymbol{\mu}_q}\mathcal{L} = \left(\frac{dg}{d\boldsymbol{\mu}_q}\right)^{\intercal} \boldsymbol{\Sigma}_{\mathbf{x}}^{-1}(\mathbf{x} - g(\boldsymbol{\mu}_q)) + \boldsymbol{\Sigma}_p^{-1}(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p). \tag{10.20}$$

10.3 Predictive Coding in Dynamical Models

Predictive coding in dynamical latent variable models is described in detail in [42, 43, 9]. Friston distinguishes between various approaches:

- **Variational Filtering** [42] uses a free-form approximate posterior, modeled as an ensemble of particles. Parameters are assumed to be stationary.
- **Dynamical Expectation Maximization (DEM)** [43] uses a fixed-form approximate posterior, making the Laplace approximation. Parameters are again assumed to be stationary.
- **Generalized Filtering** [9] also makes the Laplace approximation, but the parameters are now estimated online.

The main technical details behind these approaches are largely the same. Here, we reproduce the overall presentation given in [43].

10.3.1 Derivation

10.3.2 Models

10.4 Attention

10.5 Neural Implementation

10.5.1 Correspondence with Cortical Mircrocircuits

[2] outlines ideas about correspondence between predictive coding and cortical microcircuits.

[45] proposes that the pulvinar region of the thalamus is involved in setting precisions of predictions.

[46] gives an introductory tutorial to predictive coding with ideas about implementing predictive coding in neocortex.

[47] draws comparisons between backprop and predictive coding.

Chapter 11

Active Inference

[48], [49] , [50] [51], [52] discuss *active inference*.

Chapter 12

Experiment Best Practices

This chapter contains some helpful practices for carrying out (machine learning) research projects.

12.1 GitHub

GitHub is an online platform for project source control. It is essential when collaborating on projects, as it allows you to meticulously track changes and manage conflicts between multiple versions of the code. But GitHub is also helpful when working alone on a project across multiple machines or if you just want to open source your code. The following are the basic commands for using GitHub:

Clone a repository:

```
git clone <repository address>
```

Add a file to the repository:

```
git add <filename>
```

Or to add everything to the repository:

```
git add .
```

or

```
git add -A
```

Commit changes to the repository:

```
git commit -m "<message>"
```

or

```
git commit
```

Note that if you run the second command, you will enter a vi interface where you can enter a multi-line message. To exit, type esc :wq

Push all committed changes to the repository:

```
git push
```

Pull the repository:

```
git pull
```

To create a new branch:

```
git branch <branch name>
```

Switch current branch:

```
git checkout <branch name>
```

Merge branch changes back into master (while currently checking out other branch):

```
git merge master
```

You can send pull requests on GitHub.com. After the branch is merged, you can delete the branch on the website.

To conclude, a good workflow practice is to (1) pull the repository at the beginning of each work session, (2) push any incremental changes at regular intervals, and (3) create new branches for any major changes to be implemented.

12.2 Experiment Logging

Maintaining proper records of experiments is an essential part of conducting good research. Scientific notebooks (preferably in a digital format) are often helpful for keeping high-level notes about experiments and methodologies, but one must keep experimental logs to track the different experimental set-ups that have been tested and the results. These are vital for surveying your project (what set-ups work? what set-ups should we consider trying?) and eventually communicating your results. Implementing and keeping track of these logs can seem like a hassle, but they are just as important as the experimental code itself.

A basic experiment logging pipeline looks something like this:

1. At the beginning of the experiment, **create a log directory dedicated to this experiment**. This log directory should have a unique name. An easy choice is the date and time of the start of the experiment, which allows one to sort all of the log directories in chronological order.

2. **Copy the source code into a subdirectory in the log directory**. This is absolutely vital, as it allows one to maintain the code that led to the result of that experiment, ensuring repeatability. If you are modifying the source code while running experiments, reproducing earlier experimental conditions can be a nightmare.

   The following code creates a new log folder for the experiment and saves the files from the current directory into a subdirectory called “source.”

```
import os
from time import strftime

def init_log(log_root):
    log_dir = strftime("%b_%d_%Y_%H_%M_%S") + '/'
    os.makedirs(log_path)
    os.system("rsync -au --include '*/' --include '*.py' --
        exclude '*' . " + log_path + "source")
```

3. In addition to maintaining source code of the experiment, it's important to **log the experimental results**. At the bare minimum, one should log train and validation performance metrics. Unless these metrics are logged, you have no means of comparing different experiments after they have run. There is an unending list of other results that can be logged, such as: outputs of the model, activations within the model, distributions of different metrics and parameters as they evolve during training, etc. The downside of logging this information is that it can slow down training and take up disc space. Often, it's best to set certain logging or visualization flags within your code. This allows you to log and understand what is happening within your model during the early experimental phase/debugging, then turn off this functionality during extended training sessions.

4. **Save the model weights**. This one seems obvious, but it is easy to be lazy and write code in which you cannot save and load previous models. If you want to keep training your model, perform further evaluation, or just run and demonstrate it's capabilities, it is essential that you build this functionality into your code. Many basic libraries will make this easy, but it's up to you to add this to your experimental code to make this seamless.

There are some upfront costs associated with setting up these practices within your code, but much of this code base can also be reused for each project. You have no excuse for not keeping proper records of your experiments!

12.3 Plotting

Bibliography

[1] K. D. Harris and G. M. Shepherd, "The neocortical circuit: themes and variations," *Nature neuroscience*, vol. 18, no. 2, pp. 170–181, 2015.

[2] A. M. Bastos, W. M. Usrey, R. A. Adams, G. R. Mangun, P. Fries, and K. J. Friston, "Canonical microcircuits for predictive coding," *Neuron*, vol. 76, no. 4, pp. 695–711, 2012.

[3] M. Carandini and D. J. Heeger, "Normalization as a canonical neural computation," *Nature Reviews Neuroscience*, vol. 13, no. 1, pp. 51–62, 2012.

[4] S. Särkkä, *Bayesian filtering and smoothing*, vol. 3. Cambridge University Press, 2013.

[5] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Journal of basic engineering*, vol. 83, no. 1, pp. 95–108, 1961.

[6] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Advances in neural information processing systems*, pp. 2980–2988, 2015.

[7] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther, "Sequential neural models with stochastic layers," in *Advances in Neural Information Processing Systems*, pp. 2199–2207, 2016.

[8] M. Gemici, C.-C. Hung, A. Santoro, G. Wayne, S. Mohamed, D. J. Rezende, D. Amos, and T. Lillicrap, "Generative temporal models with memory," *arXiv preprint arXiv:1702.04649*, 2017.

[9] K. Friston, K. Stephan, B. Li, and J. Daunizeau, "Generalised filtering," *Mathematical Problems in Engineering*, vol. 2010, 2010.

[10] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[11] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv preprint arXiv:1410.8516*, 2014.

[12] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," *arXiv preprint arXiv:1505.05770*, 2015.

[13] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," 2016.

[14] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.

[15] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[16] A. Achille and S. Soatto, "On the emergence of invariance and disentangling in deep representations," *arXiv preprint arXiv:1706.01350*, 2017.

[17] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[18] B. Cheung, J. A. Livezey, A. K. Bansal, and B. A. Olshausen, "Discovering hidden factors of variation in deep networks," *arXiv preprint arXiv:1412.6583*, 2014.

[19] I. Higgins, L. Matthey, X. Glorot, A. Pal, B. Uria, C. Blundell, S. Mohamed, and A. Lerchner, "Early visual concept learning with unsupervised deep learning," *arXiv preprint arXiv:1606.05579*, 2016.

[20] N. Siddharth, B. Paige, A. Desmaison, V. de Meent, F. Wood, N. D. Goodman, P. Kohli, P. H. Torr, *et al.*, "Inducing interpretable representations with variational autoencoders," *arXiv preprint arXiv:1611.07492*, 2016.

[21] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[23] P.-Y. Oudeyer and F. Kaplan, "How can we define intrinsic motivation?," in *Proceedings of the 8th International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, Lund University Cognitive Studies, Lund: LUCS, Brighton*, Lund University Cognitive Studies, Lund: LUCS, Brighton, 2008.

[24] D. E. Berlyne, "Structure and direction in thinking.," 1965.

[25] L. Itti and P. F. Baldi, "Bayesian surprise attracts human attention," in *Advances in neural information processing systems*, pp. 547–554, 2006.

[26] J. Schmidhuber, "Formal theory of creativity, fun, and intrinsic motivation (1990–2010)," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 3, pp. 230–247, 2010.

[27] J. D. Nelson, "Finding useful questions: On bayesian diagnosticity, probability, impact, and information gain," *Psychological review*, vol. 112, no. 4, 2005.

[28] A. G. Barto, S. Singh, and N. Chentanez, "Intrinsically motivated learning of hierarchical collections of skills," in *Proceedings of the 3rd International Conference on Development and Learning*, pp. 112–19, 2004.

[29] A. S. Klyubin, D. Polani, and C. L. Nehaniv, "Empowerment: A universal agent-centric measure of control," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 1, pp. 128–135, IEEE, 2005.

[30] D. Y. Little and F. T. Sommer, "Learning and exploration in action-perception loops," *Frontiers in neural circuits*, vol. 7, 2013.

[31] A. D. Wissner-Gross and C. E. Freer, "Causal entropic forces," *Physical review letters*, vol. 110, no. 16, p. 168702, 2013.

[32] D. Barber and F. Agakov, "The im algorithm: a variational approach to information maximization," in *Proceedings of the 16th International Conference on Neural Information Processing Systems*, pp. 201–208, MIT Press, 2003.

[33] S. Mohamed and D. J. Rezende, "Variational information maximisation for intrinsically motivated reinforcement learning," in *Advances in neural information processing systems*, pp. 2125–2133, 2015.

[34] K. Gregor, D. J. Rezende, and D. Wierstra, "Variational intrinsic control," *arXiv preprint arXiv:1611.07507*, 2016.

[35] R. P. Rao and D. H. Ballard, "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects," *Nature neuroscience*, vol. 2, no. 1, pp. 79–87, 1999.

[36] H. Von Helmholtz, *Handbuch der physiologischen Optik*, vol. 9. Voss, 1867.

[37] A. Clark, "Whatever next? predictive brains, situated agents, and the future of cognitive science," *Behavioral and Brain Sciences*, vol. 36, no. 3, pp. 181–204, 2013.

[38] K. Friston, "Functional integration and inference in the brain," *Progress in neurobiology*, vol. 68, no. 2, pp. 113–143, 2002.

[39] K. Friston, "Learning and inference in the brain," *Neural Networks*, vol. 16, no. 9, pp. 1325–1352, 2003.

[40] K. Friston, "A theory of cortical responses," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 360, no. 1456, pp. 815–836, 2005.

[41] K. J. Friston, L. Harrison, and W. Penny, "Dynamic causal modelling," *Neuroimage*, vol. 19, no. 4, pp. 1273–1302, 2003.

[42] K. J. Friston, "Variational filtering," *NeuroImage*, vol. 41, no. 3, pp. 747–766, 2008.

[43] K. J. Friston, N. Trujillo-Barreto, and J. Daunizeau, "Dem: a variational treatment of dynamic systems," *Neuroimage*, vol. 41, no. 3, pp. 849–885, 2008.

[44] H. Feldman and K. J. Friston, "Attention, uncertainty, and free-energy," *Frontiers in human neuroscience*, vol. 4, 2010.

[45] R. Kanai, Y. Komura, S. Shipp, and K. Friston, "Cerebral hierarchies: predictive processing, precision and the pulvinar," *Phil. Trans. R. Soc. B*, vol. 370, no. 1668, p. 20140169, 2015.

[46] R. Bogacz, "A tutorial on the free-energy framework for modelling perception and learning," *Journal of Mathematical Psychology*, vol. 76, pp. 198–211, 2017.

[47] J. C. Whittington and R. Bogacz, "An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity," *Neural computation*, 2017.

[48] K. Friston, J. Mattout, and J. Kilner, "Action understanding and active inference," *Biological cybernetics*, vol. 104, no. 1, pp. 137–160, 2011.

[49] K. Friston, P. Schwartenbeck, T. FitzGerald, M. Moutoussis, T. Behrens, and R. J. Dolan, "The anatomy of choice: active inference and agency," *Frontiers in human neuroscience*, vol. 7, 2013.

[50] K. Friston, F. Rigoli, D. Ognibene, C. Mathys, T. Fitzgerald, and G. Pezzulo, "Active inference and epistemic value," *Cognitive neuroscience*, vol. 6, no. 4, pp. 187–214, 2015.

[51] K. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, and G. Pezzulo, "Active inference: A process theory," *Neural computation*, 2016.

[52] K. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, G. Pezzulo, *et al.*, "Active inference and learning," *Neuroscience & Biobehavioral Reviews*, vol. 68, pp. 862–879, 2016.

Appendix A

Appendix 1