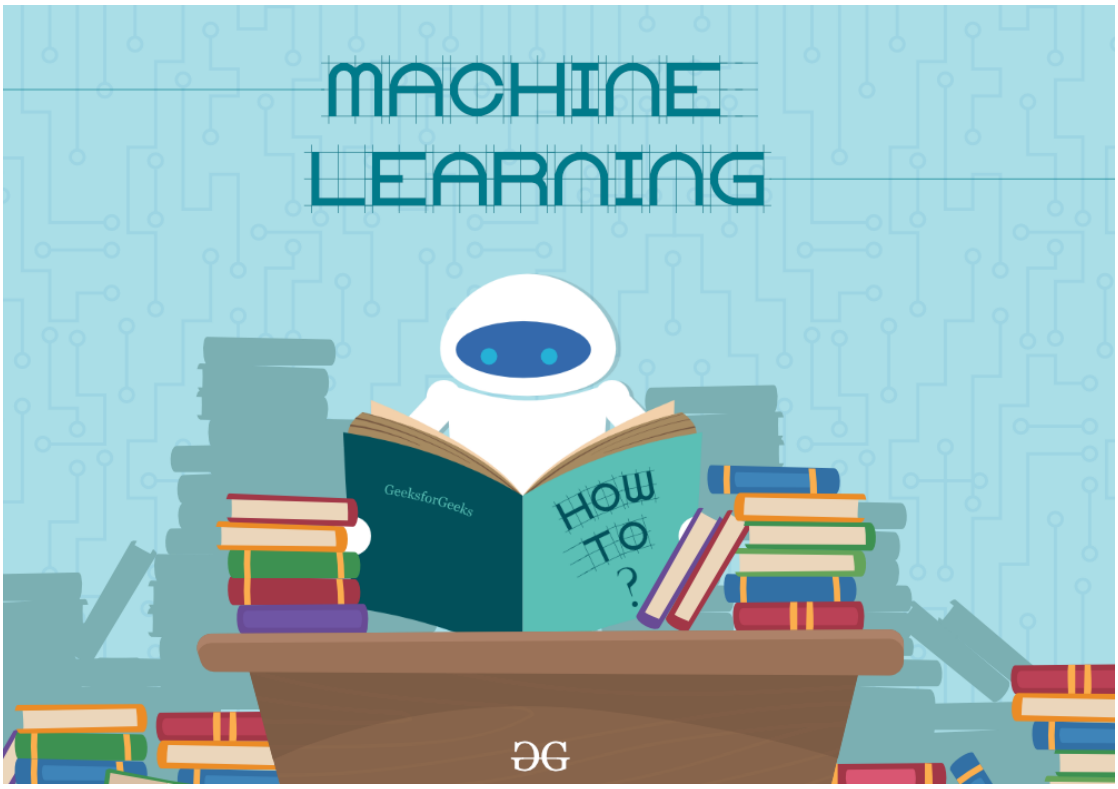
Learning and thermodynamics

Hidde Fokkema

December 27, 2021

Bachelor thesis Mathematics and Physics & Astronomy

Supervisor: dr. Bas Kleijn, dr. Greg Stephens



Institute of Physics
Korteweg-de Vries Institute for Mathematics
Faculty of Sciences
University of Amsterdam

Abstract

Machine learning is the study of algorithms that teach computers unknown processes from given data. One approach is to manipulate the input data in such a way that the algorithm only uses a useful, sometimes simpler representation. This is called representation learning. It is then possible to present how these algorithms work in a visual way. We argue that any representation learning algorithm has a general structure that can be represented by two such visual representations. The objective then becomes the minimisation of some sort of 'distance', called the KL-divergence, between the probability distributions associated with these so called graphical models. Assuming this objective, it is possible to find a calculable bound for the KL-divergence. This bound embodies the objectives of a whole class of representation learning algorithms. We will show this is true for supervised learning, Bayesian Neural Networks and the variational information bottle. This approach attempts to introduce a more principled approach into the machine learning landscape. Opening up a new way for analyses of many machine learning algorithms.

Title: Learning and thermodynamics
Author: Hidde Fokkema, hidde.fokkema@student.uva.nl, 11010673
Supervisors: dr. Bas Kleijn, dr. Greg Stephens
Second grader: dr. Bert van Es, dr. Edan Lerner
End date: December 27, 2021

Institute of Physics
University of Amsterdam
Science Park 904, 1098 XH Amsterdam
`http://www.iop.uva.nl`

Korteweg-de Vries Institute for Mathematics
University of Amsterdam
Science Park 904, 1098 XH Amsterdam
`http://www.kdvi.uva.nl`

Contents

Notation

Whenever we are dealing with vectors of multiple dimensions we will write them as **bold-face** characters: $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$. If an element is a scalar we will denote this with non bold-face characters: $x, y, z \in \mathbb{R}$. Integrating over a vector will mean that we integrate over all components

For indices, we will always use lower indices to index elements in a sequence or a data set, i.e. $(\mathbf{x}_i)_{i=1}^N$ or $(x_i)_{i=1}^N$. If we are using an upper index, then we are talking about the i'th element of a vector.

Random variables will be denoted by capital letters: X, Y. If we have a probability measure that is paramatrised by some parameter θ, then we will denote this as $p_\theta(x)$. Whenever we are integrating with respect to a probability measure $p(x)$ and for our analysis it does not matter if this is a discrete or continuous measure we will use the following notation:

$$\int \mathrm{d}P(x) = \begin{cases} \int p(x)\,\mathrm{d}x & \text{if } X \text{ is a continuous random variable,} \\ \sum_i p(x_i) & \text{if } X \text{ is a discrete random variable.} \end{cases} \tag{0.1}$$

When taking expectation values we will maintain the following convention: if there is ambiguity about which probability measure is used to calculate the expectation value, then we denote it $E_P[X]$ whenever we are using the p measure.

1 Introduction

In recent years artificial intelligence and the study of learning resurfaced. Self driving cars; programmes that can write coherent stories by themselves; and intelligent agents that can help a doctor diagnosing a patient more accurately are some of the examples of advances that were developed over the last years. The amount of research that currently is being published is difficult to keep up with. The focus of research has mostly been on neural networks, Bayesian statistics and also information theory. This thesis will combine these research areas to present a new general framework to explain how a subset of machine learning algorithm works.

That these areas were the main focus of artificial intelligence was not always the case. Around the 1950s this field emerged because of multiple discoveries. Namely, neurons either fire an impulse or do not, there is no *soft* or *hard* impulse (just like computer bits), Shannon showed that every message signal consists of bits, and Alan Turing showed that all computations can be described digitally with bits. These three facts combined resulted in researchers thinking that the brain also works as a computer and thus could be mimicked. At this stage research mostly focused on calculating many possible outcomes of a scenario and then backtracking what the optimal solution was. This proved successful for a short while. Unfortunately, this method of optimising behaviour only works when the amount of possible outcomes is not too large. This is not the case for many real world problems, such as language processing or vision. Sometimes the number of outcomes grows even exponentially, resulting in programmes that are too slow for any real world purposes. Research reached a standstill for a while, until the 80s, when expert systems were developed. These are systems that are fed logical rules beforehand, either from experience or rule of thumb, to achieve certain tasks. This greatly improved the performance, but eventually these systems also ran into problems. Most notably, they behaved erratically when presented with unusual inputs.

Figure 1.1: A world we'd like to avoid, xkcd® .

The final advance that was made was largely thanks to *Judea pearl* in 1988 and his book *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Concepts such as Bayesian inference, stochastic modelling and information theory were introduced. This introduction allowed for more precise mathematical definitions and with that more precise proofs of how artificial intelligence could work theoretically. Combined with the fact that computers were at a stage where they could process the vast amounts of data needed in reasonable time. The field had some making up to do after almost 40 years of promises and slowly some progress is being made .

More recently this has resulted in studying the behaviour of many algorithms, how these algorithms converge and if we can learn something meaningful from the two combined. In mathematics and physics, the answers do not always lie in the field itself, but can be

found by making connections to other areas. That is also why in this thesis we will try to make a connection between the study of machine learning and thermodynamics. This connection was proposed by Alemi and Fisher in 2018 [1]. They showed that when picking suitable graphical representations of these models we can find calculable bounds on *the distance* between an ideal scenario and the real world. When optimising this distance a surprising connection between learning and thermodynamics was made through the use of information theory. Namely many of the quantities calculated in this approach find their origin in how entropy or other thermodynamical quantities are calculated.

The connection between machine learning and physics is not something new. On a more practical side machine learning has proved to be an incredibly useful tool in many areas in physics. Applications of machine learning techniques have found their way into the study of phase transitions in matter [28], as well as into the study of protein folding in biophysics [21] and even into the study of quantum measurements [15]. While the results of machine learning seem promising, the question of how a certain machine learning algorithms should be implement is mostly an engineering one. By drawing inspiration from physics and statistics we try to add more principled answers for this question to a largely trial and error based field.

In this thesis we explain the necessary background knowledge to understand the approach that Alemi and Fisher have taken in their paper. In the first chapter we introduce graphical models and get comfortable with them. The usefulness of working with such graphical representations is also explained. Finally, we present some well known algorithms as graphical models to drive this point home. In the next chapter we expand our knowledge on information theory and prove some interesting and useful properties of graphical models in relation to information theory. We conclude the chapter by deriving functional bounds on information theoretic properties of our framework. In the last chapter we return to the examples shown in the second chapter and show that our framework describes a general family of algorithms. From the reader we expect some familiarity with conditional probabilities, Bayesian statistics and some measure theory.

I would like to thank dr. Bas Kleijn and dr. Greg Stephens for their help and guidance throughout my bachelor thesis. They were able to steer my into the right direction and provided me with the right resources to move forward whenever I was stuck. Especially the feedback and comments that I got on my written pieces were much appreciated.

2 Graphical models and machine learning

2.1 Graphical models

The goal of machine learning and data analysis often is to find a certain underlying process which drives or generates the data. Important questions to ask are "what are the variables that drive are relevant to this process?" and "how do they depend on each other?". However, most processes in the real world are so complicated, sensitive, or show chaotic behaviour that we cannot know all the information needed to describe the system. Examples are biological systems[16], the Lorenz attractor [13] or climate [19]. The questions we saw earlier then translates to what parameters define our distribution and how can we find them from the data. This is where graphical models come in: a graphical model is *acyclical directed graph* $G = (V, E)$, that describes the dependent structure of multivariate data. In a directed graph the edges are given by ordered pairs of nodes (v, w), with $v, w \in V$, corresponding to an arrow from v to w. This arrow tells us we can go from v to w, but not the other way around. In theory there could be a way from w to v through a different sequence of nodes. Such a sequence $(w, v_1, \dots, v_n, v)$ is called a *path*. However, the requirement of an acyclic path means that no path from v to itself exists for all $v \in V$. This implies that there cannot be another path from w to v, otherwise we could add the edge (v, w) to that path and create a cycle. Which would be a contradiction with the acyclicality of the graph.

One of the defining properties of a graphical model is that every node v_i correspond to a random variable X_{v_i}, $v_i \in V$. The edges between nodes indicate which random variables are dependent on each other. Before we define a graphical model formally we need the definition of a certain type of independence.

Definition 2.1.1. Let X, Y, Z be 3 random variables. We say that X, Y are *conditionally independent given* Z if,

$$p(X, Y|Z) = p(X|Z)p(Y|Z).$$

Commonly notated as $X \perp\!\!\!\perp Y \mid Z$

To illustrate that this is different from normal independence look at 2 dice throws. Let X be the number of eyes on the first throw and Y the number of eyes on the second throw. Let Z be the sum of both. The random variables X and Y are independent, but given Z they are not. For example, if we know that Z is even, then $p(X = 3, Y = 2|Z) = 0$, but we have that $p(X = 3|Z) \cdot p(Y = 2|Z) = \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$.

The graphical representation of this example model is given in figure 2.1. We will see later that this is a formal graphical model, as the total joint distribution is given by $p(X, Y, Z) = p(X)p(Y)p(Z|X, Y)$. This is the case because the two dice throws are independent and the sum is dependent on X and Y.

Another definition that we need before we can move on to actual examples of graphical models is *exchangeability*.

Definition 2.1.2. Consider an finite or infinite sequence of random variables $(X_1, X_2, \dots)$, with joint probability distributions $p(X_1, \dots, X_n)$ for the first n random variables for all

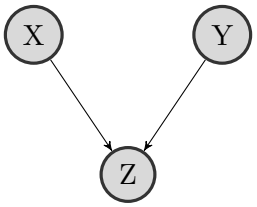


Figure 2.1: Graphical representation of two fair dice throws X, Y and their sum Z.

$n \in \mathbb{N}_{\geq 2}$. Then, this sequence is **exchangeable** if for all $n \in \mathbb{N}_{\geq 2}$ it holds that,

$$p(X_1, \ldots, X_n) = p(X_{\pi(1)}, \ldots, X_{\pi(n)}) \quad \pi \in S(n), \tag{2.1}$$

where $S(n)$ is the symmetry group of order n.

Sometimes the assumption of independent sampled data is too strong, but we still would like to reorder the data. Instead we can assume the data to be exchangeable, which would allow us to still use a familiar approach to analysing the data. Now we can introduce the graphical model.

Definition 2.1.3 (Graphical model)**.** A graphical model is an acyclic directed graph $\mathcal{G} = (V, E)$ and a collection of random variables $\{X_1, \ldots, X_n\}$ with the following properties:

- for each node $v_i \in V$ we have a random variable X_i. Notated as X_{v_i}
- The joint probability function is given by,

$$p(X_{v_1}, \ldots, X_{v_n}) = \prod_{i=1}^{n} p(X_{v_i} \mid \mathrm{Pa}(X_{v_i})). \tag{2.2}$$

  In other words, all random variables X_{v_i} are conditionally independent given its parent nodes.

This definition allows us to express statistical dependencies in a graphical way. We can also see why our example with the dice is a graphical model. The joint probability function factors into conditionally independent parts, namely $p(X, Y, Z) = p(X)p(Y)p(Z \,|\, X, Y)$. In general a given acyclical directed graph can give rise a whole range of probability distributions. If a distribution p is consistent with the given graph we will denote this by $p \models G$. Unfortunately, with large data sets these graphs can become incredibly big and messy as each data point would be represented by a new node. Therefore, we introduce the notion of *plate notation*

Definition 2.1.4 (Plate notation)**.** A *plate*, a box drawn around nodes in a graphical model, indicates a repeated dependence.

To illustrate how this simplifies the graphical models we will consider an example of generating N i.i.d. samples from a normal distribution with mean μ and variance σ^2.

Example 2.1.1. Let $x_i \sim N(\mu, \sigma^2)$ for all $i \in \{1, \ldots, N\}$. The joint probability function is then given by,

$$p(x_1, x_2, \ldots, x_N) = \prod_{i=1}^{N} p_{\mu,\sigma^2}(x_i) = \prod_{i=1}^{N} p(x_i \mid \mu, \sigma^2). \tag{2.3}$$

This example is graphically shown in figure 2.2. As we can see it greatly reduces the size and clumsiness of the former graphical representation.

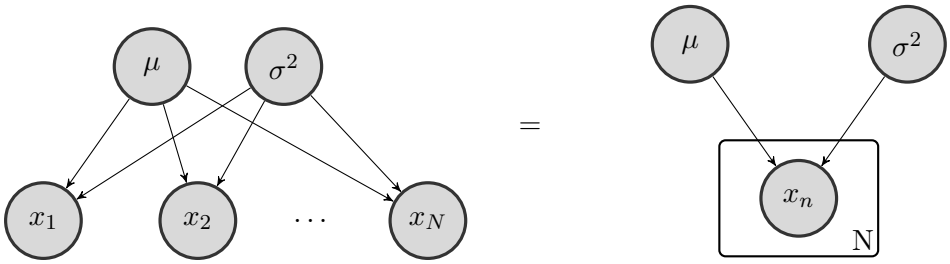


Figure 2.2: Two graphical models describing the same model. The right model uses the plate notation to make it more compact.

2.2 An illustrative example

To get more intuition on how these graphical models help us, let's look at an example. Specifically, an algorithm that is commonly used in almost every scientific field, such as artificial intelligence, econometric and biology.

Example 2.2.1 (Linear regression)**.** Consider the case of linear regression. Assume there are N data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with the following properties:

- $y_i = \mathbf{x}_i^T \beta + \epsilon_i$,
- $\epsilon_i \sim N(0, \sigma^2)$ for all $i = 1, \ldots, N$.

The goal of machine learning would be to have an algorithm that finds a $\tilde{\beta}$ which minimises a cost function or error function. Most often the quadratic difference is taken:

$$c(\{(\mathbf{x}_i, y_i)\}_{i=1}^N; \tilde{\beta}) = \sum_{i=1}^{N} \|y_i - \mathbf{x}_i^T \tilde{\beta}\|^2. \tag{2.4}$$

We will not go into detail how to solve this problem, because there exists already plenty of literature on how to do this exactly. We only want to illustrate what a simple model would look like as a graphical model. If one would want to add any priors to this model, then that can easily be done by adding a row to figure 2.3 on top with lines drawn from the hyper parameters to the β and σ^2

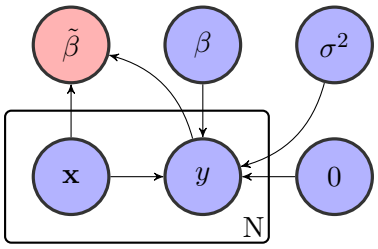


Figure 2.3: Graphical model of linear regression.

2.3 Interlude

We briefly interrupt the application of graphical models for specific algorithms and models by discussing two questions that may have come to mind. Namely, "are there distributions

that cannot be represented by a graphical model?" and "why would we use graphical models in the context of machine learning".

Let us start with the first question. The short answer is no, every joint probability distribution has a directed acyclic graph. To see why, we consider some joint probability distribution on n random variables $p(X_1, \ldots, X_n)$. In the worst case scenario, every random variable is dependent on every other random variables. Even in that case we can always factor the distribution into,

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i \mid X_{i+1}, \ldots, X_n). \tag{2.5}$$

Note however, that this factorisation is not unique and we could have done it in $n!$ ways depending on which order we choose to factor out random variables. This then gives us a way to construct a graph that would correspond with this distribution, by treating the dependencies as parent nodes. This graph will be acyclical, because whenever we factor out a random variable we essentially draw an arrow to every other random variable not factored out yet. This is a cascading effect that cannot result in cycles appearing, as arrows would need to point back towards a previous layer for that to happen. Of course, not all random variable should have to depend on every other random variable, but that would only decrease the dependencies in our factorisation and thus would lead to a simpler graph. This derivation does show that, given a joint distribution, we can create multiple graphs that would correspond to it. However, finding conditions as to when we can give a unique graph is out of the scope of this thesis.

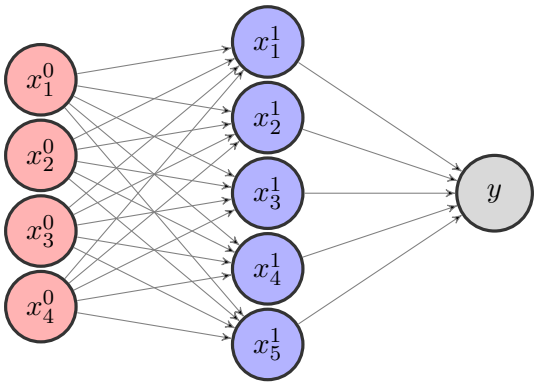


Figure 2.4: A simple neural network with one hidden layer showcasing its natural representation.

Continuing to the second question, it is still not entirely obvious why graphical models are used to represent distributions. This is the case because of the historical abundance of the use of neural networks in machine learning. These structures allow for a natural graphical representation where each layer is dependent only on the previous layer. See figure 2.4 for an example. A more informative answer is that with machine learning we are trying to find patterns and dependencies between an input and its output. Graphical models allow us to represent this causal aspect well.

2.4 Algorithms as graphical models

Now that we are familiar with graphical models and their usefulness, we will construct the graphical models of some existing machine learning algorithms. We will start in a

general setting with supervised learning and afterwards move to more specific examples. First, we will discuss Bayesian neural networks. Then we will move on to variational auto-encoders. Moving on from there, we will dive deeper by discussing the information bottleneck method, which has two variations: The variational information bottleneck and the deterministic information bottleneck.

2.4.1 Supervised learning

Before we start constructing our graphical model, we should define what supervised learning actually is. Informally, supervised learning is the process of learning a certain distribution by looking at examples. A more formal definition is given below.

Definition 2.4.1 (Supervised learning)**.** Given N labelled pairs of data points $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ (the training set), assumed to be generated by an unknown stochastic function $f : X \to Y$ with $\mathbf{y}_i = f(\mathbf{x}_i) + \epsilon_i$, *supervised learning* is the process of finding a function $\tilde{f}$ that approximates f in some appropriate metric [23] [3]. This process is entirely based on looking at the data points as examples.

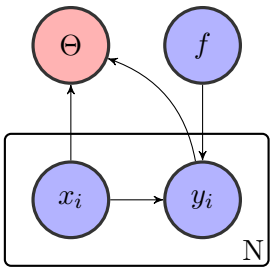


Figure 2.5: Supervised learning represented as a graphical model.

If Y is a space with an infinite amount of elements this process is commonly referred to as *regression*and if y is a finite space it is referred to as *classification*. We claim that this process is described by the following model.

Theorem 2.4.1. Let $\{(x_i, y_i)\}_{i=1}^N$ be N labelled pairs of data points generated by some function f. We assume these data points to be exchangeable. If $p_\Theta(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{x}, \Theta)$ is some conditional pdf that is obtained from a supervised learning algorithm then we can describe the process with figure 2.5

Before we move on towards the proof, we note that this figure resembles figure 2.3 when we let $\Theta = \tilde{\beta}$ and change f to our generating process depending on β and $N(0, \sigma^2)$.

Proof. By definition of the setup we instantly have that all the $\mathbf{y}_i$'s are all only dependent on f and $\mathbf{x}_i$, $p(\mathbf{x}_i, \mathbf{y}_i, f) = p(\mathbf{y}_i|\mathbf{x}_i, f)$. Similarly, supervised learning uses only the $\mathbf{x}_i$'s and $\mathbf{y}_i$'s to estimate an optimal Θ by definition. Combining everything, we get:

$$p(\{(x_i, y_i)\}_{i=1}^N, \Theta, f) = p(f) \prod_{i=1}^N p(x_i, y_i, \Theta \mid f) = p(f) \prod_{i=1}^N p(\mathbf{x}_i) p(\mathbf{y}_i|\mathbf{x}_i, f) p(\Theta|\mathbf{x}_i, \mathbf{y}_i), \quad (2.6)$$

which aligns with the graphical model in 2.5. □

2.4.2 Bayesian neural networks

As promised, we will look at a more specific example now. However, before we define a Bayesian neural network we need to know exactly what a normal neural network is.

Definition 2.4.2 (Neural network)**.** An n layered neural network is a a function from $\mathbb{R}^p \to \mathbb{R}^d$ consisting of the following elements:

- Each layer consists of n_i neurons, where i indicates the layer it belongs to.
- Each neuron carries an individual weight parameter w_j^i, where i indicates the layer and j which node of that layer it belongs to. The complete set of weights will be referred to as $\mathbf{w}$.
- An activation function $\phi : \mathbb{R} \to \mathbb{R}_{\geq 0}$.
- We let $\mathbf{x}_0 \in \mathbb{R}^p$ be the input and $\mathbf{x}_{n+1} \in \mathbb{R}^d$ the output, then the output is calculated in steps as follows,

$$x_{i+1}^j = \varphi(\sum_{k=1}^{n_i} w_k^i x_i^k). \tag{2.7}$$

These networks are used to approximate functions or probability distributions. A labelled data set $\{(X_i, Y_i)\}_{i=1}^N$ is used to train the network. To see how the optimising works in practise, we refer to Bishop with his excellent book *Pattern Recognition and Machine Learning* [3] or any other book on machine learning.

This is not a Bayesian neural network however. For that we refer to the model proposed by Blundell et al. in [4]. The main difference between a normal neural network and a Bayesian neural network is that we do not have fixed weights, but consider each weight as having its own independent distribution. The goal of the optimising scheme then is to calculate the posterior distribution $p_\theta(\mathbf{w} \mid \{(X_i, Y_i)\}_{i=1}^N)$. Once you have such a posterior we can calculate the predictive posterior by integrating out the weight dependence, given some observed data point $\mathbf{x}$ we get:

$$p_\theta(\hat{\mathbf{y}}|\mathbf{x}) = \int p(\hat{\mathbf{y}} \mid \mathbf{w}, \mathbf{x}) \, \mathrm{d}P_\theta(\mathbf{w} \mid \{(X_i, Y_i)\}_{i=1}^N). \tag{2.8}$$

Blundel et al. argue that using these kind of networks requires less weights than a normal neural network. The intuitive explanation for this is that by 'averaging out' all the possible values of the weights you account for the uncertainty in the weights of a normal neural network. The possibility of using fewer nodes is their appeal, but the expression in 2.8 is rarely tractable. To still be able to calculate some sort of solution we consider a different distribution $q_\psi(\mathbf{w} \mid \theta)$ to approximate the prior $p_\theta(\mathbf{w})$. This is called an variational approximation. Of course, we would like the new distribution to lie close to the true distribution. We will see in chapter 4 how this is done.

For now we need to know how the graphical model of the whole algorithm looks. One might think that this model would look similar to the one in figure 2.4. However, that is a graphical model of how to calculate an output given an input. If we look at how the model is trained we get something different. For that we first look at the following dependencies:

- The data is generated by some unknown process Φ and we assume that Y_i is dependent of X_i.
- The parameters $\mathbf{w}$ is dependent on some labelled data $\{(X_i, Y_i)\}_{i=1}^N$ and some parameter θ.

The eventual graph is described here in figure 2.6.

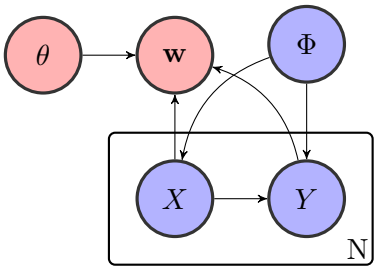


Figure 2.6: The graphical model that describes the algorithm behind the Bayesian neural network.

2.4.3 Variational auto-encoders

Neural networks as we have seen them so far work great when you have labelled data, but you run into trouble when you do not and you still want to do some analysis with this setup. For that purpose auto-encoders were introduced. We will look at a special case of these auto-encoders introduced by Kingma and Welling in [18]. The essential idea is to project your data to some *latent variable* and then to reconstruct your original data point.

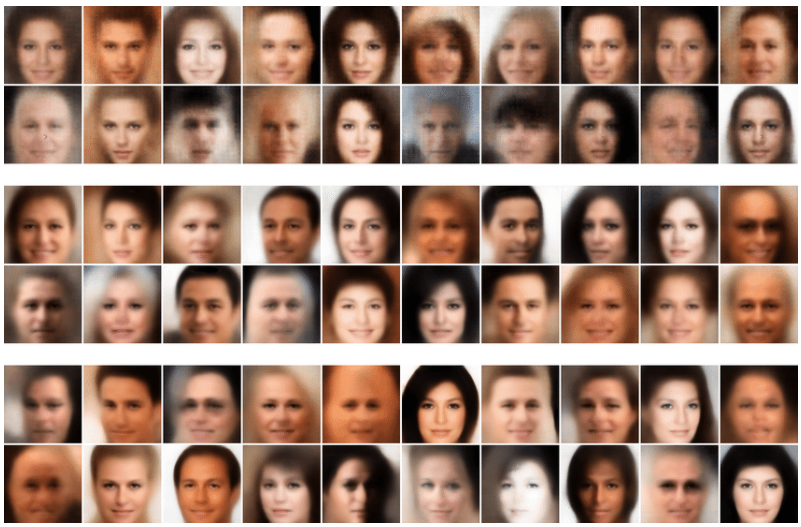


Figure 2.7: Some examples of generated faces by different kinds of auto-encoders [5].

Definition 2.4.3 (latent variable)**.** A *latent variable*, commonly notated with a $\mathbf{z}$, is a variable that is assumed to exist in a model or algorithm, but is not necessarily observed [17].

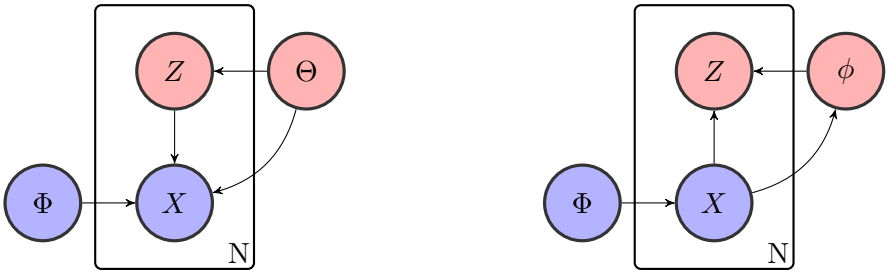
The reason for such a term is that we can rewrite $p_\Theta(\mathbf{x}_i)$ as,

$$p_\Theta(\mathbf{x}_i) = \int \mathrm{d}P_\Theta(\mathbf{x}_i, \mathbf{z}_i). = \int p_\Theta(\mathbf{x}_i \,|\, \mathbf{z}_i)\, \mathrm{d}P_\Theta(\mathbf{z}_i). \tag{2.9}$$

This might not seem so advantageous, but even when the joint distributions $p(\mathbf{x}_i, \mathbf{z}_i)$ are rather simple, the marginal can become a rather complex distributions. This also allows us to generate data from the likelihood $p_\Theta(\mathbf{x}_i \,|\, \mathbf{z}_i)$ by sampling from the prior distribution $p_\Theta(z_i)$. This is called a generative model or a generative process. This aspect of auto-encoders has shown to be very useful, as it is fairly difficult to make a machine draw a

picture of something specific, for example. Various auto-encoding models have shown to be quite effective in generating samples. A different model that does this incredibly well as well are *Generative Adversial Nets* [14].

To return to the latent variable, we have already seen such variables implicitly. Namely, we can see the intermediate values that are calculated in the intermediate layers of neural networks as latent variables. In the variational auto-encoder a different kind of neural network is used to make this latent variable more explicit. To do this, you take the input variable X_i, as both the input and as the to be target output. That way you train your network to reconstruct X_i from the latent variable. In the end, this will result in a posterior $p_\theta(z \mid x)$ that will be able to do this *encoding*. We can represent this in a fairly simple diagram in the diagram of figure 2.8b.



(a) generative model (b) variational approximation

Figure 2.8: A diagram showing the variational auto-encoder framework.

We will look more in debt what this model is optimising in a later chapter. For now, we can shed some light in the diagrams in 2.8. The marginal in equation 2.9 is in general not tractable. Again, a variational approach is adopted. We introduce the distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$ to approximate the posterior distribution. Note however, that we are now dealing with two separate processes. The process of finding the likelihood $p_\Theta(\mathbf{x}|\mathbf{z})$ and optimising the variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$. Kingma and Welling showed that you can find an objective that does both tasks simultaneously. We will see later what this objective is.

2.4.4 Information Bottleneck

It is interesting to see that VAE models generate positive results. What is it about projecting our input onto a latent variable that gets us any further? In 2000 Tishby offered an answer to a slightly different, but not entirely irrelevant, question by introducing his *information bottleneck* (IB) technique[27]. Informally said, he argued that these models are compressing the information that we have about the data into a representation that is useful for our algorithm, leaving only the useful information.

Variational information bottleneck

Tishby only spoke about the generative model and did not incorporate the variational distribution in his research. This was later done by Alemi et al. in 2016 [2], calling their approach the *Variational Information bottleneck*. They also showed that the variational auto-encoder discussed earlier was a specific case of their result. The basic idea behind the technique is that when we have some labelled data $\{(X_i, Y_i)\}_{i=1}^N$, then we want to find an encoding of X_i, called Z_i, that compresses the information of X_i maximally while also keeping as much information about the dependence between Y_i and X_i. In the case of

unlabelled data $\{X_i\}_{i=1}^N$ we can replace the Y_i by X_i and then we get a technique that is similar to what auto-encoders do. How you would calculate these information quantities will be discussed in the next chapter. In practise, this is done by projecting your high dimensional data $\{X_i\}_{i=1}^N$ down to a lower dimensional space and then reconstructing it again. The reconstruction error is then minimised for some metric of your choice. In this framework we assume that our Z is dependent X and our Y is dependent on X, and X, Y are both generated by some data generating process. We want to find an *encoder* dependent on our data that does this transformation between X and Y. In the most general case we can represent this as in figure 2.9.

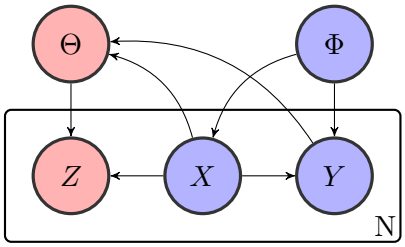


Figure 2.9: The variational information bottleneck algorithm represented as graphical model.

This is also how Tishby presented his framework. However, we run into a similar problem as the issues described in the previous models, when actually trying to do any meaningful inference. Namely, calculating the *decoder* $p_\theta(\mathbf{y} \mid \mathbf{z})$ is in most cases not possible. Instead, we can show how you would go about calculating this distribution formally.

Lemma 2.4.1. The *decoder* $p_\theta(\mathbf{y} \mid \mathbf{z})$ in the information bottleneck method is given by,

$$p_\theta(\mathbf{y} \mid \mathbf{z}) = \int \frac{p_\theta(\mathbf{y} \mid \mathbf{x}) p_\theta(\mathbf{z} \mid \mathbf{x})}{p_\theta(\mathbf{z})} p_\theta(\mathbf{x}) \mathrm{d}\mathbf{x}. \tag{2.10}$$

Proof. From the diagram we can see that $p_\theta(\mathbf{y} \mid \mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{y} \mid \mathbf{x})$. With that knowledge we can write,

$$p_\theta(\mathbf{y} \mid \mathbf{z}) = \int p_\theta(\mathbf{y}, \mathbf{x} \mid \mathbf{z}) \mathrm{d}\mathbf{x} = \int p_\theta(\mathbf{y} \mid \mathbf{x}) p(\mathbf{z} \mid \mathbf{x}) \mathrm{d}\mathbf{x} = \int \frac{p_\theta(\mathbf{y} \mid \mathbf{x}) p_\theta(\mathbf{z} \mid \mathbf{x})}{p_\theta(\mathbf{z})} p_\theta(\mathbf{x}) \mathrm{d}\mathbf{x}. \tag{2.11}$$

□

As we have seen before this decoder will not be tractable for a general case. Tishby then continues his analysis in a different setting than ours. However, Alemi et al. adopt a now familiar method by introducing a variational approximation $q_\psi(\mathbf{y} \mid \mathbf{z})$ that should lie close to $p_\theta(\mathbf{y} \mid \mathbf{z})$ in some sense. We will see later what objective emerges from this approach.

Deterministic Information bottleneck

It is also interesting to note the *deterministic information bottleneck*, introduced in 2016 by Schwab and Strouse [26]. The model still follows the outline given in figure 2.9. The difference with the variational information bottleneck and the information bottleneck is that they choose to work with a slightly different objective, which still resembles to the objective of the variational information bottleneck. Surprisingly, this objective resulted in a decoder that was not stochastic in nature, but deterministic, as the name also implies.

Again, we will see why this is the case in a later stage. Looking at this model is interesting for two reasons according to Schwab and Strouse. The first reason being that it performs better in some cases than the original information bottleneck method. Furthermore, they argue that having a deterministic outcome is also advantageous, as it leaves no ambiguity in how to choose a specific outcome. A problem you have in a stochastic approach.

2.5 World P vs world Q

Thus far we have seen how we can represent certain machine learning algorithms as graphical models. The attentive reader might also have noticed that we have a general pattern for how to make these algorithms work. This suggest that there exists a general framework when dealing with this specific kind of learning. The type of learning here is called *representation learning* and we want to cast *all* algorithms connected with this type of learning into a general framework.

Alemi and Fischer propose to do this in the following way as presented in figure 2.10 [1]. On the left we see what they call world P, which is what we can achieve. Note that this is identical to how the variational information bottleneck method is presented. Here we can see that the labelled data $\{(X_i, Y_i)\}_{i=1}^N$ determines the parameters in Θ. These parameters are optimised by the algorithm of choice. Each latent variable Z_i is in turn dependent on these parameters and and the corresponding X_i. That brings us to the Φ in our model. It is some data generating process of which we would like to learn the relationship between X_i and Y_i. In other words, we would like to know the underlying true distribution $p(\mathbf{y} \mid \mathbf{x})$.

This description is world P, but this is not the ideal situation. The latent variable Z_i should be able to describe the complete relationship between X_i and Y_i in an ideal situation. If that is true, then the VAE would be able to generate perfect samples, that would be indistinguishable from the data in the training set. That ideal scenario is what world Q represents. The goal of any representation algorithm would then be to find a projection of a distribution p from world P unto world Q that remains close to true distribution. This would be our variation approximation q, that we have seen many times now. To do this we will have to find a suitable definition for "distance". The next section will focus on that aspect.

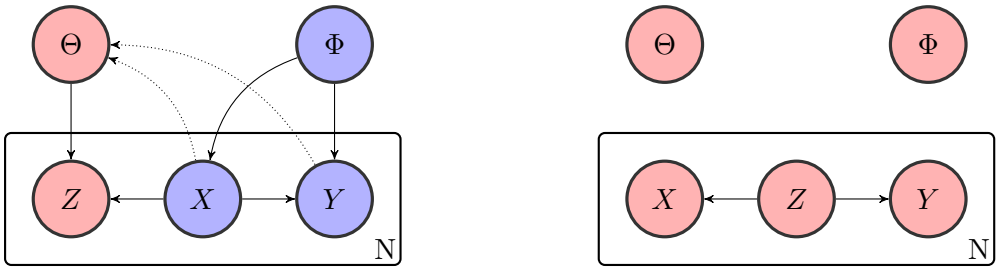


Figure 2.10: The two graphical models of world P (l) and world Q (r) as described in [1].

3 Information theory and entropy

We now have two graphical models, one representing the world that we have to work with, and one the world that we would like to achieve. For the world P we choose a distribution p that is conditional on the data through a learning algorithm . The world Q can have any distribution consistent with its graphical model [1] and we would like to find the distribution that lies closest to our distribution p in some sense. To do this we will introduce something called the *Kullback-Leibler* divergence (KL-divergence), which can be seen as a notion of the distance between two distributions [12]. However, this is not a formal metric and its use does require some care from a measure theoretic viewpoint. Before we can introduce the KL-divergence we need to introduce some concepts related to information and entropy.

3.1 Shannon information

It was Shannon, who first was able to give a quantitative description of what information is in 1948[25]. He was looking to find theoretical bounds on how much information can be retained when sending a message through a communication channel. In that time we did not have the luxury of modern day technology where data can be send easily from one agent to another without any problems. Communication channels were prone to lose bits and pieces of a message whenever the size of the message was too big. Knowing what messages could go through without a loss of quality was thus very important at the time. However, that did require a formal definition of what information is and how to calculate it. Before Shannon derived his definition, he wrote down 4 requirements this definition should have. For a discrete random variable X with probabilities p_i for $i \in \{1, \ldots, n\}$ we denote this quantity as $H(X)$. The properties it should have are:

- The quantity H should be continuous in the probabilities p_i for all $i \in \{1, \ldots, n\}$.
- Whenever we have that $p_i = \frac{1}{n}$ for all i, then H should be monotonically increasing with respect to n.
- Whenever two events, X and Y, are independent we should be able to calculate the information by $H(X, Y) = H(X) + H(Y)$.
- Every event with probability 1 and probability 0 should have information 0.

From this list of properties Shannon showed that this information function can only have the form of definition 3.1.1. For a proof of this fact we refer to Shannon's original paper [25].

Definition 3.1.1. Let X be a discrete random variable with values $\{x_1, x_2, \ldots x_n\}$ and probability mass function $p(x)$, then the Shannon information is given by,

$$H(X) = \mathbb{E}[I(X)] = \mathbb{E}[-\log(p(X))] = -\sum_{i=1}^{n} p(x_i) \log(p(x_i)). \tag{3.1}$$

Here $I(X)$ is called the information content of X. We can also define a similar quantity for continuous random variables by replacing the sum with an integral and replacing the probability mass function with the respective density. However, more care has to be used when using this definition, as they are not completely equivalent. For example a continuous random variable can have negative information. The latter is also referred to as the differential entropy, to distinguish the two cases.

It must be noted that the base for the logarithm in this context is commonly taken to be 2 as we are talking about *bits* of information. As these are almost always encoded in a base-2 system it is natural to take the base-2 logarithm. In the case of using the natural logarithm, we talk about *nats* of information.

This was the first time that the amount of information within in a message could be quantified. The inspiration for this definition came from thermodynamics where the Gibbs entropy for an ensemble with discrete energy states is defined as,

$$S(X) = -k_B \sum_{i=1}^{n} p_i \ln(p_i). \tag{3.2}$$

Here k_B is the Boltzmann constant and the sum goes over all energy states of the ensemble, each with probability p_i. For now we will leave this connection to thermodynamics be, but we will get back to it when we are more familiar the concept of information.

3.2 Predictive information

This description of information looks rather simple, but still proved to be very useful. With machine learning we are trying to predict a certain outcome based on features that can be measured. Thus what do the features $X = (X_1, X_2, \ldots, X_n)$ tell us about property Y. In terms of information we would like to know how much relevant information is stored in the vector X about Y. Unfortunately, Shannon information falls short in this aspect. As it can only tell us something about the general information stored, not the information stored about something else. Luckily, Shannon and Weaver gave a definition called *mutual information* that does tell us about the information between to random variables [24].

Definition 3.2.1. Let X, Y be two random variables, then the mutual information is given by,

$$I(X;Y) = \mathbb{E}\left[\log \frac{p(X,Y)}{p(X)p(Y)}\right]. \tag{3.3}$$

A sanity check would be to see what happens whenever both variables are independent. In that case, no information is contained in X about Y and the mutual information should be 0. If we consider that situation we get the following result for the predictive information,

$$I(X;Y) = \mathbb{E}\left[\log \frac{p(X,Y)}{p(X)p(Y)}\right] = \mathbb{E}\left[\log \frac{p(X)p(Y)}{p(X)p(Y)}\right] = \mathbb{E}[\log 1] = 0, \tag{3.4}$$

which luckily confirms our sanity check.

It is of course natural to ask what the Shannon information and mutual information for joint and conditional random variables are.

Definition 3.2.2 (Conditional mutual information)**.** let X, Y and Z be random variables, we can define the condition mutual information by,

$$I(X;Y \mid Z) = H(X \mid Z) - H(X \mid Y, Z) \tag{3.5}$$

From these definitions of information we can deduce some nice identities that will prove to be useful.

Proposition 3.2.1 (Joint and conditional information properties)**.** Let X, Y and Z be random variables. For the conditional and joint random variables we have the following identities:

1. $H(X,Y) = H(X) + H(Y \mid X) = H(Y) + H(X \mid Y)$,
2. $I(X;Y) = H(X) - H(X|Y)$,
3. $I(X,Y;Z) = I(X;Z) + I(Y;Z \mid X)$, this is also referred to as the Chain rule of information.

Proof. For the first identity we just write out the definition of $H(X,Y)$:

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

All operations that were used are completely symmetric in X and Y, giving us the second equality immediately as well. Continuing to the second identity we get,

$$\begin{aligned} I(X;Y) &= -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log\frac{p(x)p(y)}{p(x,y)} = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log\frac{p(x)}{p(x \mid y)} \\ &= -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log p(x) + \sum_{x\in X}\sum_{y\in Y} p(x,y)\log p(x \mid y) \\ &= -\sum_{x\in X} p(x)\log p(x) + \sum_{x\in X}\sum_{y\in Y} p(x,y)\log p(x \mid y) \\ &= H(X) - H(X \mid Y). \end{aligned}$$

Now we can combine these two identities to get the third identity:

$$\begin{aligned} I(X,Y;Z) &= H(X,Y) - H(X,Y \mid Z) = H(X) + H(Y \mid X) - [H(X|Z) + H(Y \mid X,Z)] \\ &= I(X;Z) + I(Y;Z \mid X). \end{aligned}$$

□

3.3 Kullback – Leibler divergence

Now that we have all the ingredients we can define the KL-divergence.

Definition 3.3.1 (Kullback – Leibler divergence)**.** Let (X,p,Σ) and (X,q,Σ) be two discrete probability spaces, with the the same event space and sigma algebra, but different probability measures. Assuming $p << q$, The KL-divergence is defined as,

$$D_{KL}(P \parallel Q) = -\sum_{i=1}^{n} p(x_i)\log\frac{q(x_i)}{p(x_i)} = \mathbb{E}_P\left[\log\frac{p(X)}{q(X)}\right]. \tag{3.6}$$

This is a well defined quantity whenever p is dominated by q. To see this we need to check some boundary cases. Namely, whenever $q(x) = 0$ or whenever $q(x) \neq 0$, but $p(x) = 0$.

Proposition 3.3.1. Whenever $p << q$, then we have the following identities:

- $q(x) = 0 \implies p(x) \log\left(\frac{q(x)}{p(x)}\right) = 0$,
- $q(x) \neq 0$, $p(x) = 0 \implies p(x) \log\left(\frac{q(x)}{p(x)}\right) = 0$.

Proof. The first assertion is equivalent to checking that $x \log(x) \to 0$, whenever $x \to 0$. This is the case because the fraction within the logarithm will go to 0 as $q(x) \to 0$, as we have that p is dominated by q. We find the following limit:

$$\lim_{x \to 0} x \log(x) = \lim_{t \to \infty} \frac{\log\left(\frac{1}{t}\right)}{t} = \lim_{t \to \infty} \frac{\frac{-1}{t}}{1} = 0. \tag{3.7}$$

For the second identity we use a similar argument. We note that this question is the same as asking about what happens to $\frac{\log(x)}{x}$ as $x \to \infty$,

$$\lim_{x \to \infty} \frac{\log(x)}{x} = \lim_{x \to \infty} \frac{\frac{1}{x}}{1} = 0. \tag{3.8}$$

□

We can now safely fill in zeroes whenever we meet such boundary cases and the KL-divergence will behave nicely. The assumption that $p << q$ is not a trivial assumption, but in the real world not crazy to ask. We want that our distribution q approximates some real world phenomenon with distribution p. If we have that p gives a positive probability for an event B that cannot happen (i.e. $q(B) = 0$) in our approximation, it would be nonsensical to look at this distribution q and we will look at a different distribution

It was noted before that the KL-divergence is **not** a metric, most notably because it is not symmetric, which should be clear from its definition. Formally speaking, the KL-divergence will not give us a distance between our two worlds. However, it can be seen as the information that we lose about $q(x)$ by using $p(x)$. In Bayesian inference this non commutative order is logical as we start out with a prior $p(x)$ and infer a posterior $q(x)$, which is generally not a symmetric operation.

Even though the KL-divergence is not a formal metric, it does have some properties that are useful for treating it as a distance between probability measures.

Proposition 3.3.2. The KL - divergence is positive definite, meaning:

- $D_{KL}(P \parallel Q) \geq 0$,
- $D_{KL}(P \parallel Q) = 0 \iff p = q$.

Proof. Rewriting the KL-divergence as:

$$\begin{aligned} D_{KL}(P \parallel Q) = \sum_{i=1}^{N} p_i \log p_i - \sum_{i=1}^{N} p_i \log q_i \geq 0 \iff \\ - \sum_{i=1}^{N} p_i \log p_i \leq - \sum_{i=1}^{N} p_i \log q_i. \end{aligned} \tag{3.9}$$

The final inequality is also known as the Gibbs' inequality. To prove that inequality, we us the fact that $\log x \leq x - 1$ with equality only whenever $x = 1$. Using this we get,

$$-\sum_{i=1}^{N} p_i \log p_i + \sum_{i=1}^{N} p_i \log q_i = \sum_{i=1}^{N} p_i \log \frac{q_i}{p_i} \leq \sum_{i=1}^{N} p_i \left(\frac{q_i}{p_i} - 1 \right) \tag{3.10}$$

$$= \sum_{i=1}^{N} q_i - \sum_{i=1}^{N} p_i = 1 - 1 = 0 \iff \tag{3.11}$$

$$-\sum_{i=1}^{N} p_i \log p_i \leq -\sum_{i=1}^{N} p_i \log q_i. \tag{3.12}$$

Whenever we want equality we need to have that $\frac{q_i}{p_i} = 1$ for every $p_i \in P, q_i \in Q$. Which is only the case when $q_i = p_i \iff P = Q$.

The fact that the KL-divergence is always positive and only equal for equivalent measures is incredibly useful and will be used extensively later on in this chapter. □

Now, if we look at our two models in figure 2.10 we note the following: in world P the distribution p is chosen by our modelling assumptions and some learning algorithm. However, because world Q is unknown it allows many different distributions. The only requirement being that they need to satisfy the condition $q(X, Y|Z) = q(X|Z)q(Y|Z)$. For the purpose of measuring the performance of our world P we look at,

$$\mathcal{J} = \min_{q \models Q} D_{KL}(p \parallel q). \tag{3.13}$$

Now that we have a way of measuring the difference between our two worlds it is still not clear how to compute this quantity. Unfortunately this expression is intractable in most cases and we will have to settle with some upper bounds for it, but first we would like to see what this KL-divergence would look like in our case.

3.3.1 Information vs other statistics

Before we move on towards applying these definitions to graphical models, it is useful to think about why we would want to adopt this information theoretic framework. Mutual information and the KL-divergence are not too say the easiest quantities to calculate. So there must be some advantage to using them over other commonly used statistics, such as the sample covariance matrix, correlation factors or the fisher information.

Informally speaking, the reason we use information is because it catches *any* dependency between two variables X, Y. Not just a linear or higher order dependencies. This is demonstrated by the fact that $I(X;Y) = 0$ if and only if the variables are independent. So any dependency will result into the mutual information to be non-zero. In machine learning context this is useful, as we want to make as little as possible assumptions about the process we are studying. The idealistic goal of any machine learning algorithm should be to find the correct underlying dependency by itself. We will look at two specific case to showcase that information encapsulates other statistics. First, we will look at the bivariate normal distribution, whose two random variables have a linear dependency between them. This will also be evident from the mutual information.

Theorem 3.3.1. Let $(X, Y) \sim N((\mu_X, \mu_Y), \Sigma)$ with correlation coefficient ρ. The mutual information is then given by,

$$I(X;Y) = -\frac{1}{2} \log(1 - \rho^2). \tag{3.14}$$

Proof. We will make use of the fact that we can write the covariance matrix as,

$$\Sigma = \begin{bmatrix} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2 \end{bmatrix}. \tag{3.15}$$

We will also use the following from proposition 3.2.1,

$$I(X;Y) = H(X) + H(Y) - H(X,Y). \tag{3.16}$$

Now we just need to calculate the three entropies given above. Recall that the marginal of X_i of a bivariate normal distribution is again normally distributed with mean μ_i and variance σ_i^2. The three entropies are then given by:

1. $H(X) = -\int -\frac{1}{2}\log(2\pi\sigma_X^2) - \frac{1}{2\sigma_X^2}(x-\mu_X)^2 \mathrm{d}P(x) = \frac{1}{2}\log(2\pi\sigma_X^2) + \frac{1}{2}$.
2. Analogously we find $H(Y) = \frac{1}{2}\log(2\pi\sigma_Y^2) + \frac{1}{2}$.
3. The final entropy requires a bit more work. To start this calculation we note that we can write $p(x,y)$ as,

$$p(x,y) = \frac{1}{2\pi|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2(1-\rho^2)}\left[\frac{(x-\mu_X)^2}{\sigma_X^2} + \frac{(y-\mu_Y)^2}{\sigma_Y^2} - \frac{2\rho(x-\mu_X)(y-\mu_Y)}{\sigma_X\sigma_Y}\right]}. \tag{3.17}$$

Now we can calculate the entropy,

$$\begin{aligned} H(X,Y) &= \int \log 2\pi|\Sigma|^{\frac{1}{2}} + \\ &\frac{1}{2(1-\rho^2)}\left[\frac{(x-\mu_X)^2}{\sigma_X^2} + \frac{(y-\mu_Y)^2}{\sigma_Y^2} - \frac{2\rho(x-\mu_X)(y-\mu_Y)}{\sigma_X\sigma_Y}\right]\mathrm{d}P(x,y) \\ &= \log 2\pi|\Sigma|^{\frac{1}{2}} + \frac{1}{2(1-\rho^2)}(2 - \frac{2\rho(\rho\sigma_X\sigma_Y)}{\sigma_X\sigma_Y}) \\ &= \log 2\pi|\Sigma|^{\frac{1}{2}} + 1. \end{aligned} \tag{3.18}$$

Putting this all together we finally get,

$$\begin{aligned} I(X;Y) &= \frac{1}{2}\log(2\pi\sigma_X^2) + \frac{1}{2} + \frac{1}{2}\log(2\pi\sigma_Y^2) + \frac{1}{2} - \log 2\pi|\Sigma|^{\frac{1}{2}} - 1 \\ &= \frac{1}{2}\log(\sigma_X^2\sigma_Y^2) - \frac{1}{2}\log(\sigma_X^2\sigma_Y^2 - \rho^2\sigma_X^2\sigma_Y^2) \\ &= -\frac{1}{2}\log(1-\rho^2). \end{aligned}$$

Which shows that the mutual information captures the linear dependence of the two variables. □

To further strengthen our believe that the information tells us something about all dependencies, it is also possible to show that the second order term of the Taylor expansion of the KL-divergence is given by the *Fisher information*. The probability distribution has to be twice differentiable of course. A proof can be found in [9].

Some caution must be used whenever we are dealing with information. The property of information to incorporate all dependencies is useful in its own right, but also causes the information to be very sensitive to changes in the data. So, from a modelling perspective it is always required to argue if some dependency is logical. Otherwise, the model could create outcomes that are not desirable.

3.3.2 Multi-Information

For probability densities with more than two random variables we can generalise the idea of mutual information by defining the *Multi - information*, I^P which measures how well the joint probability function is approximated by the product of its marginals. Later on we will see that this is useful in calculating the bounds of $\mathcal{J}$.

Definition 3.3.2 (Multi-information)**.** Given a joint probability measure $p(X_1, \ldots, X_N)$ with marginal distributions $p(X_i)$, then the *multi-information* is given by,

$$I^p = D_{KL}\left[p(X_1, \ldots, X_n) \parallel \prod_{i=1}^{n} p(X_i)\right]. \tag{3.19}$$

Using the mutual information between nodes and their parent nodes we can find another expression for the multi-information whenever we are dealing with a graphical model.

Theorem 3.3.2. Given a graphical model $\mathcal{G}$ with random variables $(X_1, \ldots, X_N)$ on N nodes with a joint probability measure $p \models \mathcal{G}$. The multi-information can be written as,

$$I^p_{\mathcal{G}} = \sum_{i=1}^{N} I(X_i; \mathrm{Pa}(X_i)). \tag{3.20}$$

Proof. This identity follows directly from the definition of the multi-information and a graphical model,

$$\begin{aligned}
I^p_{\mathcal{G}} &= D_{KL}\left(p(X_1, \ldots, X_N) \parallel \prod_{i=1}^{N} p(X_i)\right) \\
&= \mathbb{E}_P\left[\log \frac{p(X_i, \ldots, X_N)}{\prod_{i=1}^{N} p(X_i)}\right] \\
&= \mathbb{E}_P\left[\log \prod_{i=1}^{N} \frac{p(X_i \mid \mathrm{Pa}(X_i))}{p(X_i)}\right] \\
&= \sum_{i=1}^{N} \mathbb{E}_P\left[\log \frac{p(X_i \mid \mathrm{Pa}(X_i))}{p(X_i)}\right] \\
&= \sum_{i=1}^{N} \mathbb{E}_P\left[\log \frac{p(X_i \mid \mathrm{Pa}(X_i))p(\mathrm{Pa}(X_i))}{p(X_i)p(\mathrm{Pa}(X_i))}\right] \\
&= \sum_{i=1}^{N} \mathbb{E}_P\left[\log \frac{p(X_i, \mathrm{Pa}(X_i))}{p(X_i)p(\mathrm{Pa}(X_i))}\right] = \sum_{i=1}^{N} I(X_i; \mathrm{Pa}(X_i)).
\end{aligned} \tag{3.21}$$

□

Friedman et al. [10] claim that it is possible to use the multi-information as a way to calculate $\mathcal{J}$, which is shown in theorem 3.3.3.

Theorem 3.3.3. Let $\mathcal{G}_1$, and $\mathcal{G}_2$ be 2 graphical models. Let $p \models \mathcal{G}_1$, then we can calculate the KL-divergence in terms of the multi-informations,

$$\mathcal{J} = \min_{q \models \mathcal{G}_2} D_{KL}[p \parallel q] = I^p_{\mathcal{G}_1} - I^p_{\mathcal{G}_2}. \tag{3.22}$$

Proof. We will not prove this claim completely, as that requires knowledge outside of the scope of this these. We can give an intuition why this is true however by first writing out the KL-divergence between p and q:

$$\begin{aligned} D_{KL}(p \parallel q) &= \mathbb{E}_P\left[\log \frac{p(X_1,\dots,X_N)}{q(X_1,\dots,X_N)}\right] \\ &= \mathbb{E}_P\left[\log \frac{p(X_1,\dots,X_N)}{q(X_1,\dots,X_N)} \frac{\prod_{i=1}^N p(X_i)}{\prod_{i=1}^N p(X_i)} \frac{\prod_{i=1}^N q(X_i)}{\prod_{i=1}^N q(X_i)}\right] \\ &= \mathbb{E}_P\left[\log \frac{p(X_1,\dots,X_N)}{\prod_{i=1}^N p(X_i)}\right] - \mathbb{E}_P\left[\log \frac{q(X_1,\dots,X_N)}{\prod_{i=1}^N q(X_i)}\right] \\ &+ \mathbb{E}_P\left[\log \frac{\prod_{i=1}^N p(X_i)}{\prod_{i=1}^N q(X_i)}\right] \\ &= I^p_{\mathcal{G}_1} - I^p_{\mathcal{G}_2} + \mathbb{E}_P\left[\log \frac{\prod_{i=1}^N p(X_i)}{\prod_{i=1}^N q(X_i)}\right]. \end{aligned} \tag{3.23}$$

The q that minimises the KL-divergence is known as the *information reversal projection* [8] and under some assumption you can show it exists and is unique. We will assume that this projection must have the same marginals as p [6][7]. That means the last term in our above derivation vanishes and we are left with $I^p_{\mathcal{G}_1} - I^p_{\mathcal{G}_2}$. □

Now that we have the tools we can define the minimal KL-divergence between the worlds P and Q.

Theorem 3.3.4. For the models presented in figure 2.10 we now see that we get the following $\mathcal{J}$:

$$\mathcal{J} = I(\Theta; X^N, Y^N) + \sum_{i=1}^{N} [I(X_i;\Phi) + I(Y_i; X_i, \Phi) + I(Z_i; X_i, \Theta) - I(X_i; Z_i) - I(Y_i; Z_i)]. \tag{3.24}$$

Proof. As given by the figure we can factor the distribution p with the following dependencies:

- The random variable Θ is dependent on (X^N, Y^N).
- Every X_i is dependent only on the process Φ.
- Every Y_i is dependent on both X_i and Φ.
- Finally in P the latent variable Z_i is dependent on X_i and Θ.

For the world Q we only have that both X_i and Y_i are dependent on Z_i. This combined with theorem 3.3.3 gives the above expression. □

3.4 Functional bounds

In theory we now have a way of computing the difference between our two worlds. However, in general these forms are not tractable. The terms $I(X_i;\Phi)$ and $I(Y_i; X_i : \Phi)$ are even completely out of our control. So, even if we know the values, then they would be of no use as we could not change them. For the other terms we will have to resort to upper bounds. Luckily, these bounds can be established by well known quantities in the machine learning context.

3.4.1 Relative entropy

The first bound we look at is called the *relative entropy*. This is the mutual information between the posterior of our parameter in world P and an independent prior $q(\Theta)$.

Proposition 3.4.1. Given parameter Θ and data $\{(X_i, Y_i)\}_{i=1}^N$. The relative entropy with it's corresponding bound is given by,

$$S = \mathbb{E}_P \left[\log \frac{p(\Theta \,|X^N, Y^N)}{q(\Theta)} \right] \geq I(\Theta; X^N, Y^N) \tag{3.25}$$

Proof. We start out by writing out the mutual information,

$$\begin{aligned} I(\Theta; X^N, Y^N) &= \mathbb{E}_P \left[\log \frac{p(\Theta, X^N, Y^N)}{p(\Theta)p(X^N, Y^N)} \right] \\ &= \mathbb{E}_P \left[\log \frac{p(\Theta \,|\, X^N, Y^N)}{p(\Theta)} \right] \\ &= \mathbb{E}_P \left[\log \frac{p(\Theta \,|\, X^N, Y^N) q(\Theta)}{p(\Theta) q(\Theta)} \right] \\ &= \mathbb{E}_P \left[\log \frac{p(\Theta \,|\, X^N, Y^N)}{q(\Theta)} \right] - \mathbb{E}_P \left[\log \frac{p(\Theta)}{q(\Theta)} \right] \\ &= \mathbb{E}_P \left[\log \frac{p(\Theta \,|\, X^N, Y^N)}{q(\Theta)} \right] - D_{KL} \left[p(\Theta) \;||\; q(\Theta) \right] \\ &\leq \mathbb{E}_P \left[\frac{p(\Theta \,|\, X^N, Y^N)}{q(\Theta)} \right] = S. \end{aligned} \tag{3.26}$$

The last inequality follows from the fact that the KL-divergence is positive definite. □

3.4.2 Rate

The second upper bound we have is something named the *Rate*. This is a bound for how much information our representation Z_i retains about its input data X_i. First, we give the bound for a single data point

Proposition 3.4.2. Given a random variable X_i, it's representation Z_i and a parameter Θ, the rate and the corresponding bound is given by,

$$R_i = \mathbb{E}_P \left[\log \frac{p(Z_i \,|X_i, \Theta)}{q(Z_i)} \right] \geq I(Z_i; X_i, \Theta). \tag{3.27}$$

Proof. This proof goes completely analogous to the previous proof by using the following substitutions:

- $\Theta \to Z_i$,
- $X^N \to X_i$,
- $Y^N \to \Theta$.

□

The full bound over all data is then given by the sum $\sum_{i=1}^N R_i = R$.

3.4.3 Classification error

The next bound we are going to look at is a more well known concept, albeit that we will introduce it in an unfamiliar way. The *classification error* refers to a measure of how far the output of a model is from the true value. Various models use different quantities, commonly used quantities are any of the averaged L^p differences, most notably $p = 2$, accuracy in percentages or hinge loss [22]. The Classification error is used in algorithms like *linear regression*, *support vector machines* and *neural networks* to optimise their outcomes. These metrics are also referred to as loss functions. In our setting the classification error measures how much information we have of Y_i in our representation Z_i.

Proposition 3.4.3. Given a random variable Y_i, a representation Z_i, the classification error for a singel data point and it's corresponding bound is given by,

$$C_i = -\mathbb{E}_P[\log q(Y_i \,|Z_i\,)] \geq H(Y_i) - I(Y_i;\, Z_i) = H(Y_i \,|\, Z_i). \tag{3.28}$$

Proof. First we note that the final equality is given by the properties shown in theorem 3.2.1. Next, we prove the inequality by a similar approach as the other bounds by writing out the definition of $H(Y_i \,|\, Z_i)$,

$$\begin{aligned} H(Y_i \,|\, Z_i) &= -\mathbb{E}_P\left[\log p(Y_i|Z_i)\right] = -\mathbb{E}_P\left[\log \frac{p(Y_i|Z_i)q(Y_i \,|\, Z_i)}{q(Y_i \,|\, Z_i)}\right] \\ &= -\mathbb{E}_P\left[\log q(Y_i \,|\, Z_i)\right] - \mathbb{E}_P\left[\log \frac{p(Y_i|Z_i)}{q(Y_i \,|\, Z_i)}\right] \\ &= -\mathbb{E}_P\left[\log q(Y_i \,|\, Z_i)\right] - D_{KL}\left[p(Y_i \,|\, Z_i) \,||\, q(Y_i \,|\, Z_i)\right] \\ &\leq -\mathbb{E}_P\left[\log q(Y_i \,|\, Z_i)\right] = C_i. \end{aligned} \tag{3.29}$$

□

To get the bound over the full data set we again sum over all data points $\sum_{i=1}^{N} C_i = C$.

3.4.4 Distortion

Similarly, we can look at a measure that indicates how close the representation Z_i is to the true input value X_i. This is commonly referred to as the *distortion*. The methods for defining the distortion are all fairly similar to the classification error options. As an example, consider the case when we have clustered data $\{x_i\}_{i=1}^{N}$. Let the centres of each cluster be represented by $\{d_i\}_{i=1}^{K}$. We could represent every point by the cluster centre. If we pick the mean squared error our distortion would be $D = \frac{1}{N}\sum_{i=1}^{N}(x_i - d_{x_i})^2$. However, in this case we will define the distortion in a different way.

Proposition 3.4.4. Given a random variable X_i and it's representation Z_i, the rate for a single point and it's corresponding bound is given by,

$$D_i = -\mathbb{E}_P[\log q(X_i \,|\, Z_i)] \geq H(X_i) - I(X_i; Z_i) = H(X_i \,|\, Z_i). \tag{3.30}$$

Proof. Note that the only difference between this definition and the classification error is that that Y_i is substituted with X_i. Thus, the proof will go analogously and we will not repeat it. □

The bound that we will be using is again the sum of all the D_i's, namely $\sum_{i=1}^{N} D_i = D$.

3.5 Complete bound

Now that we all these bounds we can use them to find an upper bound for the $\mathcal{J}$ quantity defined in theorem 3.3.4.

Theorem 3.5.1. For the models presented in figure 2.10 the following bound holds,

$$\mathcal{J} \leq S + R + D + C - \sum_{i=1}^{N} H(Y_i, X_i \,|\, \Phi). \tag{3.31}$$

Proof. We already did most of the legwork when we were defining the functional bounds. Proofing this bound is now only a case of filling in the previous bounds. By theorem 3.3.4 we have,

$$\begin{aligned}
\mathcal{J} &= I(\Theta; X^N, Y^N) + \sum_{i=1}^{N} [I(X_i; \Phi) + I(Y_i; X_i, \Phi) + I(Z_i; X_i, \Theta) - I(X_i; Z_i) - I(Y_i; Z_i)] \\
&\leq S + R + D + C - \sum_{i=1}^{N} [H(X_i) + H(Y_i) - I(X_i; \Phi) - I(Y_i; X_i, \Phi)] \\
&= S + R + D + C - \sum_{i=1}^{N} [H(X_i \,|\, \Phi) + H(Y_i | X_i, \Phi)] \\
&= S + R + D + C - \sum_{i=1}^{N} H(Y_i, X_i \,|\, \Phi).
\end{aligned} \tag{3.32}$$

Where the last equality follows from the properties of conditional entropy defined in proposition 3.2.1 □

This expression gives us the quintessential bound for this chapter. The final term in the expression, $\sum_{i=1}^{N} H(Y_i, X_i \,|\, \Phi)$, is completely out of our control, as it is determined by the data and the data generating process. The other four terms take on well defined forms whenever we choose our distributions p and q.

4 Objectives of models

Now that we have our functional bound on the minimal D_{KL} we will show that this bound is indeed a generalisation of the objective for a few machine learning algorithms. In general we would like to find a procedure that finds p and q that minimises this bound. This will not guarantee that we find the best possible p and q, but in general our solution will lie close to the correct distributions. Unfortunately, no general solution for finding such a procedure has been found. We refer to respective articles for the specific procedures for each method. As a side note, from the derivations of R, C, D, S we can see that the gap between the bound and the actual $\mathcal{J}$ is determined by $D_{KL}\left[p(\Theta) \,\|\, q(\Theta)\right]$, $\sum_i D_{KL}\left[p(Z_i) \,\|\, q(Z_i)\right]$, $\sum_i D_{KL}\left[p(Y_i \,|\, Z_i) \,\|\, q(Y_i \,|\, Z_i)\right]$ and $\sum_i D_{KL}\left[p(Y_i \,|\, Z_i) \,\|\, q(X_i \,|\, Z_i)\right]$. Meaning that if these divergences are small, the bound becomes tight.

For the next part we follow the reasoning of Alemi and Fisher [1]. However, the validity of these claims rely on some underlying assumptions. Taking these assumptions as true we can argue that the region of optimal, attainable points in $\mathbb{R}^4$ traced out by S, R, D, C result into a 3-dimensional surface. If we assume that this surface is smooth enough it is possible to describe this surface as the kernel of some real valued function, namely $f(R, C, D, S) = 0$. Of course, depending on what exact model we are using to learn we have different restrictions on the probability distributions that we are allowed to incorporate. As is commonly done, we can impose these constrains by utilising Lagrange multipliers. Note that he use of Lagrange multipliers does rely on the fact that our optimal surface is smooth enough. This results in the following objective.

Proposition 4.0.1 (Lagrange multipliers)**.** Given a data generating process Φ we would like to find a procedure that finds p and q such that,

$$\min_{p \models P, q \models Q} R + \delta D + \gamma C + \sigma S. \tag{4.1}$$

The final sections of this thesis will be dedicated to showing that we can retrieve the specific bounds of some machine learning algorithms when using the objective as described in proposition 4.1.

4.1 Supervised learning

Supervised Learning can be separated into two classes. Namely, *regression* and *classification*. Regression occurs when our target space contains an infinite amount of values, countable or uncountable. Classification happens when our target space contains only finitely many values. For our discussion we will look at the case of regression. Instead of just looking at linear regression we will look at the general case, where we assume our data comes in the form of $\{(X_i, Y_i)\}_{i=1}^N$, where X_i can be multi-dimensional, with the following properties:

- The random variable Y_i is given by, $Y_i = f(X_i) + \epsilon_i$ for some unknown function $f : \mathbb{R}^N \to \mathbb{R}$.
- The error term e_i is given by $\epsilon_i \sim N(0, \sigma^2)$ for all ϵ_i.

A well known objective used to find an optimal approximating $\hat{f}$ for f is the *Sum of Squared residual errors* (SSE),

$$SSE = \sum_{i=1}^{N} (\hat{f}(x_i) - y_i)^2. \tag{4.2}$$

What we will show is that loss function is determined by the assumption that the ϵ_i's are normally distributed, modulo some constant. To do this we will look at the objective defined in proposition 4.1. We start by choosing our constraints in such a way that we only retain C. Then, we notice that Z_i is given by $\hat{f}(X_i)$ in this case. Furthermore, by assumption of $\epsilon_i \sim N(0, \sigma^2)$ we have that $q(Y_i|Z_i)$ is a normal distribution with mean Z_i and variance σ^2. This gives us,

$$\begin{aligned}
C &= \sum_{i=1}^{N} -\mathbb{E}_P \left[\log(q(Y_i \mid Z_i))\right] \\
&= -\sum_{i=1}^{N} \int \log(q(y_i \mid z_i) \, \mathrm{d}P(y_i \mid z_i) \\
&= \sum_{i=1}^{N} \int (y_i - \hat{f}(x_i))^2 + \log \sqrt{2\pi\sigma^2} \, \mathrm{d}P(y_i \mid \hat{f}(X_i)) \\
&= \frac{N}{2} \log 2\pi\sigma^2 + \sum_{i=1}^{N} \int (y_i - \hat{f}(x_i))^2 \, \mathrm{d}P(y_i \mid \hat{f}(X_i)) \\
&= \frac{N}{2} \log 2\pi\sigma^2 + \sum_{i=1}^{N} \mathbb{E}_P[(Y_i - \hat{f}(X_i))^2].
\end{aligned}$$

We can ignore the first term in the last expression, our main focus of interest is the second term. From maximum likelihood estimation we know that this quantity is minimised whenever we pick $\hat{f}$ in such a way that,

$$\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{f}(x_i))^2.$$

is minimal. Summing this N times gives us exactly the SSE.

4.2 Bayesian neural network

As in chapter 2 we will now be looking at a more specific example by analysing the Bayesian Neural network model. We now have the tools to define what objective we want to minimise and we will see that this is of the form as in proposition 4.1, when choosing the right constraints. Just to reiterate what the Bayesian neural network does again, we have a neural network with weights $\mathbf{w}$, who are each sampled from a posterior that is dependent on the data $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^{N}$. To find an approximation to the posterior predictive, we introduced a variational distribution $q(\mathbf{w} \mid \theta)$. The objective would be to minimise the KL-divergence between the true distribution p and the approximating one q, which is $D_{KL}[q(\mathbf{w} \mid \theta) \,\|\, q(\mathbf{w} \mid \mathcal{D})]$. For the following analysis to align with our framework, we will relabel the distribution p as q and the distribution q as p.

Proposition 4.2.1. For the Bayesian Neural network the objective of minimising the quantity $D_{KL}[p(\mathbf{w} \mid \theta) \,\|\, q(\mathbf{w} \mid \mathcal{D})]$ is equivalent to 4.1 when keeping only terms $C + S$, which can be achieved by letting $\gamma \to \infty$ while $\frac{\gamma}{\sigma} \to 1$ and $\delta = 0$.

Proof. We start of by writing out the KL-divergence,

$$\begin{aligned} D_{KL}[p(\mathbf{w} \mid \theta) \,\|\, q(\mathbf{w} \mid \mathcal{D})] &= \int \log \frac{p(\mathbf{w} \mid \theta)}{q(\mathbf{w} \mid \mathcal{D})} \,\mathrm{d}P(\mathbf{w} \mid \theta) \\ &= \int \log \frac{p(\mathbf{w} \mid \theta) q(\mathcal{D})}{q(\mathcal{D} \mid \mathbf{w}) q(\mathbf{w})} \,\mathrm{d}P(\mathbf{w} \mid \theta) \\ &= \int \log \frac{p(\mathbf{w} \mid \theta)}{q(\mathbf{w})} \,\mathrm{d}P(\mathbf{w} \mid \theta) - \int \log q(\mathcal{D} \mid \mathbf{w}) \,\mathrm{d}P(\mathbf{w} \mid \theta) \\ &\quad + \log q(\mathcal{D}) \\ &= \mathbb{E}_{p(\mathbf{w} \mid \theta)} \left[\log \frac{p(\mathbf{w} \mid \theta)}{q(\mathbf{w})} \right] - \mathbb{E}_{p(\mathbf{w} \mid \theta)}[\log q(\mathcal{D} \mid \mathbf{w})] + \log q(\mathcal{D}) \\ &= S + C + \log q(\mathcal{D}). \end{aligned} \tag{4.3}$$

Minimising over θ now gives the desired result. We have to stretch our definition of S and C quite a bit however. Namely, here S does not depend on the data and C has no mention of the latent variable and target variable. This quantity is also referred to as the variational free energy [11]. □

4.3 Variational auto-encoder

For the variational auto-encoder we have to do less work, as the original paper embraces the same framework in a less general way. The goal of the variational auto-encoder is to find the marginal predictive distribution $q_\psi(\mathbf{x})$. However, the posterior $q_\psi(\mathbf{z} \mid \mathbf{x})$ is in most cases not tractable. So we approximate it with a $p_\theta(\mathbf{z} \mid \mathbf{x})$. In this case we want to find a ψ that maximises the marginal likelihood. Note that we have swapped the roles of p and q compared to chapter 2. This is done to keep the notation in line with chapter 3 and our objective.

Proposition 4.3.1. For the variational auto-encoder the objective of maximising the marginal likelihood is equivalent to minimising in 4.1 when keeping only $R + D$, by letting $\gamma = 0$, $\sigma = 0$ and $\delta = 1$.

Proof. We start by writing out the marginal likelihood for one data point, as the full likelihood is given by the sum log likelihoods per point,

$$\begin{aligned} \log q_\psi(\mathbf{x}_i) &= \int \log q_\psi(\mathbf{x}_i) \,\mathrm{d}P_\theta(\mathbf{z_i} \mid \mathbf{x}_i) \\ &= \int \log \frac{p_\theta(\mathbf{z}_i \mid \mathbf{x}_i) q_\psi(\mathbf{x}_i, \mathbf{z}_i)}{q_\psi(\mathbf{z}_i \mid \mathbf{x}_i) p_\theta(\mathbf{z}_i \mid \mathbf{x}_i)} \,\mathrm{d}P_\theta(\mathbf{z_i} \mid \mathbf{x}_i) \\ &= \int \log \frac{p_\theta(\mathbf{z}_i \mid \mathbf{x}_i)}{q_\psi(\mathbf{z}_i \mid \mathbf{x}_i)} \,\mathrm{d}P_\theta(\mathbf{z_i} \mid \mathbf{x}_i) + \int \log q_\psi(\mathbf{x}_i, \mathbf{z}_i) \,\mathrm{d}P_\theta(\mathbf{z_i} \mid \mathbf{x}_i) \\ &\quad - \int \log p_\theta(\mathbf{z}_i \mid \mathbf{x}_i) \,\mathrm{d}P_\theta(\mathbf{z_i} \mid \mathbf{x}_i) \\ &= D_{KL}\left[p_\theta(\mathbf{z}_i \mid \mathbf{x}_i) \,\|\, q_\psi(\mathbf{z}_i \mid \mathbf{x}_i)\right] + \mathcal{L}(\theta, \psi, \mathbf{x}_i, \mathbf{z}_i). \end{aligned} \tag{4.4}$$

The KL-divergence term is something we cannot calculate, but if we ignore it we see that $\log p(\mathbf{x}_i) \geq \mathcal{L}(\theta, \psi, \mathbf{x}_i, \mathbf{z}_i)$. We can rewrite this expression as,

$$\begin{aligned}\mathcal{L}(\theta, \psi, \mathbf{x}_i, \mathbf{z}_i) &= \int \log q_\psi(\mathbf{x}_i, \mathbf{z}_i)\, \mathrm{d}P_\theta(\mathbf{z_i} \,|\, \mathbf{x}_i) - \int \log p_\theta(\mathbf{z}_i \,|\, \mathbf{x}_i)\, \mathrm{d}P_\theta(\mathbf{z_i} \,|\, \mathbf{x}_i) \\ &= \int \log \frac{q_\psi(\mathbf{x}_i|\mathbf{z}_i) q_\psi(\mathbf{z}_i)}{p_\theta(\mathbf{z}_i \,|\, \mathbf{x}_i)}\, \mathrm{d}P_\theta(\mathbf{z_i} \,|\, \mathbf{x}_i) \\ &= \int \log \frac{q_\psi(\mathbf{z}_i)}{p_\theta(\mathbf{z}_i \,|\, \mathbf{x}_i)}\, \mathrm{d}P_\theta(\mathbf{z_i} \,|\, \mathbf{x}_i) + \mathbb{E}_{p_\theta(\mathbf{z}_i \,|\, \mathbf{x}_i)}[\log q_\psi(\mathbf{x}_i \,|\, \mathbf{z}_i)] \\ &= -D_{KL}\left[p_\theta(\mathbf{z}_i \,|\, \mathbf{x}_i) \,\|\, q_\psi(\mathbf{z}_i)\right] + \mathbb{E}_{p(\mathbf{z}_i \,|\, \mathbf{x}_i)}[\log q_\psi(\mathbf{x}_i \,|\, \mathbf{z}_i)] = -(R_i + D_i).\end{aligned} \tag{4.5}$$

Summing over all data points then gives that the full likelihood is greater or equal to $-(R+D)$. So minimising $R+D$ will maximise the lower bound on the marginal likelihood, which is the objective of VAE. □

4.4 Information bottleneck

As promised, we will also be looking at the two information bottleneck methods. Starting with the variational information bottleneck.

4.4.1 Variational information bottleneck

The main idea behind the variational information bottleneck is to minimise the reconstruction error when calculating Y from Z. However, the amount of information that Z is allowed to have about X is under some constraint. This suggests that the relevant functionals that we want to keep from proposition 4.1 are R and C. The quantity R measures how much complexity we allow in our Z given X and the functional C is needed as it measures what the error will be between our representation Z and the true Y. We will see that this intuition coincides with the objective as presented in the article by Alemi et al. [2]. It could also be argued that D is needed, but we will see that this is not the case with the variational information bottleneck.

Showing what the objective is for the variational information bottleneck requires a bit of work. First, we will introduce the original objective of the information bottleneck method. Then, we will show that this has a lower bound given in terms of the variational approximation, which in turn will give us the required R_i and C_i.

Definition 4.4.1. Given data $\{(X, Y)\}_{i=1}^N$ with assumed dependencies as in 2.9. The information bottleneck objective is given by,

$$\max_\theta \sum_{i=1}^N I(Z_i; Y_i) - \beta I(Z_i; X_i, \theta). \tag{4.6}$$

Here β regulates how much information of X we allow to be preserved in Z.

What we will show is the following proposition.

Proposition 4.4.1. The objective as given in definition 4.4.1 has a lower bound given by,

$$\sum_{i=1}^N I(Z_i; Y_i) - \beta I(Z_i; X_i, \theta) \geq -(\beta R + C). \tag{4.7}$$

Proof. We have already seen some useful inequalities for this in chapter 3. Namely,

$$I(Z_i, Y_i) \geq H(Y_i) - C_i \geq -C_i. \tag{4.8}$$

The last inequality follows from the fact that $H(Y_i)$ is always positive. Next, we focus on the term $-\beta I(Z_i; X_i, \theta)$. Again, we already have seen a useful inequality given in chapter 3,

$$I(Z_i; X_i, \theta) \leq R_i \iff -\beta I(Z_i; X_i, \theta) \geq -\beta R_i. \tag{4.9}$$

Now summing over all the terms gives the desired result,

$$\begin{aligned} &I(Z_i, Y_i) - \beta I(Z_i; X_i, \theta) \geq -(\beta R_i + C_i) \quad \text{for all } i \in \{1, \dots, N\} \implies \\ &\sum_{i=1}^{N} I(Z_i, Y_i) - \beta I(Z_i; X_i, \theta) \geq -(\beta R + C). \end{aligned} \tag{4.10}$$

□

Instead of maximising the original objective, we can maximise the lower bound $-(\beta R + C)$ to come to an approximate solution. This is equivalent to,

$$\min_{\theta} R + \frac{1}{\beta} C. \tag{4.11}$$

Here we recognise the objective from the start of this chapter, but with the constraints $\delta = \sigma = 0$ and $\gamma = \frac{1}{\beta}$.

4.4.2 Deterministic information bottleneck

We are trying to maximise the term governing the reconstruction of our output $I(Z_i; Y_i)$ while setting constraints on how much information we are allowed to keep in $I(Z_i; X_i, \theta)$ when using the information bottleneck and variational information bottleneck method. In the case of the deterministic information bottleneck, we are not looking to constrain the amount of information that Z_i is allowed to have of X_i. Instead, we want to put constraints on how much information we are allowed to use in general to describe Z_i. To achieve this constraint we change $I(Z_i; X_i, \theta)$ into $H(Z_i)$.

Definition 4.4.2. Given data $\{(X_i, Y_i)\}_{i=1}^N$ with assumed dependencies as in 2.9. The deterministic information bottleneck objective is given by,

$$\max_{\theta} \sum_{i=1}^{N} I(Z_i; Y_i) - \beta H(Z_i). \tag{4.12}$$

Here β regulates how much information of X we allow to be preserved in Z.

An overview of why this objective results into a deterministic outcome is given below. For a more detailed explanation we refer to the original paper by Strouse and Schwab [26]. We want to find an iterative process that attains the goal given in definition 4.4.2. To achieve this it is necessary to define the objective as a limit of another objective. This objective is given by,

$$\max_{\theta} \sum_{i=1}^{N} \alpha H(Z_i \,|\, X_i) + I(Z_i; Y_i) - \beta H(Z_i). \tag{4.13}$$

We recover our objective by taking the limit $\alpha \to 0$. A formal solution to this problem can be given. Namely,

$$q_\alpha(\mathbf{z}_i \mid \mathbf{x}_i) = \frac{1}{Z(\mathbf{x}_i, \alpha, \beta)} e^{\frac{1}{\alpha}(\log q_\alpha(t) - \beta D_{KL}[p(\mathbf{y}_i \mid \mathbf{x}_i) \,\|\, q_\alpha(\mathbf{y}_i \mid \mathbf{z}_i)])}, \tag{4.14}$$

$$q_\alpha(\mathbf{y}_i \mid \mathbf{z}_i) = \frac{1}{q_\alpha(\mathbf{z}_i)} \int q_\alpha(\mathbf{z}_i \mid \mathbf{x}_i) p(\mathbf{y}_i \mid \mathbf{x}_i) \, \mathrm{d}P(\mathbf{x}_i). \tag{4.15}$$

where $Z(\mathbf{x}_i, \alpha, \beta)$ is the normalisation constant. If we take the limit $\alpha \to 0$, we see that in the first expression the exponential will blow up. This might look like not something useful, but that means that $q_\alpha(\mathbf{z}_i \mid \mathbf{x}_i)$ collapses into a delta distribution at the point $\mathbf{z}_i$ where $\log q_\alpha(t) - \beta D_{KL}\left[p(\mathbf{y}_i \mid \mathbf{x}_i) \,\|\, q_\alpha(\mathbf{y}_i \mid \mathbf{z}_i)\right]$ is the biggest. This can be interepreted as function that maps $\mathbf{x}_i$ to a given $\mathbf{z}_i$ en thus gives a *deterministic* encoder.

This method does not fit into the framework shown in proposition 4.1. The fact that we can find a function instead of a probability distribution by slightly tweaking the objective is interesting however. Maybe there are ways to tweak the objective in proposition 4.1 in a similar way to also get a function instead of a distribution? That question is unfortunately out of the scope for this thesis however.

5 Conclusion

The goal of this thesis was to describe a new framework to study machine learning algorithms. We argued that the goal of a representation learning algorithm can be represented in terms of two graphical models, world P and world Q. By minimising the KL-divergence between the probability densities describing the two models, we find a general objective representing multiple representation learning algorithms.

To reach this goal, we first introduced the concept of a graphical model and explained it usefulness. Namely, the visual representation of certain dependencies allow for an intuitive interpretation. Later one we also saw that these graphical models also allow for some nicer computations of information theoretic quantities. Before we displayed how we can describe representation learning algorithms in general with a graphical model, it was necessary to show specific examples. In particular we looked at supervised learning, Bayesian neural networks, variational auto-encoders and the information bottleneck. For each, a corresponding graphical model was constructed. From there a more general formulation was shown in terms of world P and world Q.

With this formulation we can construct a general objective for these algorithms in terms of the KL-divergence between the probability densities describing world P and world Q. It was argued that the KL-divergence was not possible to calculate directly. An upper bound was formally shown to exists, which consisted of 4 quantities: S, R, C and D.

To finalise the analysis of this approach we show that this bound is in fact more general. Again, we considered the cases of supervised learning, Bayesian neural networks, variational auto-encoders and the information bottleneck. For each method we showed that it was possible to retrieve their respective objective from the general objective, by picking the right constraints on this objective in terms of Lagrangian multipliers.

Throughout this thesis, we not much is said much about the convergence of these algorithms. There was no general description found for this aspect. The framework we adopted to describe the models has not shown a clear way to incorporate this aspect of modelling yet. The papers describing the studied algorithms do provide optimisation schemes for each individual models, but no general solution was found so far.

Unfortunately, the connection between thermodynamics and the approach outlined in this thesis were not fully explored. The approach in this thesis lead to connections that were too arbitrary and did not result into new insights. However, we are confident that there are many connections that can be made between machine learning and physics. One example is the connection between the renormalisation group and deep learning [20]. These final remarks are left to pique the readers interest into further research subjects.

Bibliography

[1] Alexander A. Alemi and Ian Fischer. Therml: Thermodynamics of machine learning. *CoRR*, abs/1807.04162, 2018.

[2] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[4] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[5] Lei Cai, Hongyang Gao, and Shuiwang Ji. Multi-stage variational auto-encoders for coarse-to-fine image generation. 05 2017.

[6] I. Csiszar. i-divergence geometry of probability distributions and minimization problems. *Ann. Probab.*, 3(1):146–158, 02 1975.

[7] Imre Csiszar. Sanov property, generalized i-projection and a conditional limit theorem. *Ann. Probab.*, 12(3):768–793, 08 1984.

[8] Imre Csiszár and Frantisek Matus. Information projections revisited. *IEEE Transactions on Information Theory*, 49(6):1474–1490, 2003.

[9] Anand Dabak and Don Johnson. Relations between kullback-leibler distance and fisher information. 02 2003.

[10] Nir Friedman, Ori Mosenzon, Noam Slonim, and Naftali Tishby. Multivariate information bottleneck. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 152–161. Morgan Kaufmann Publishers Inc., 2001.

[11] Karl Friston, Jérémie Mattout, Nelson Trujillo-Barreto, John Ashburner, and Will Penny. Variational free energy and the laplace approximation. *Neuroimage*, 34(1):220–234, 2007.

[12] David Galas, Gregory Dewey, James Kunert-Graf, and Nikita Sakhanenko. Expansion of the kullback-leibler divergence, and a new class of information metrics. *Axioms*, 6(2):8, 2017.

[13] Étienne Ghys. *The Lorenz Attractor, a Paradigm for Chaos*, pages 1–54. Springer Basel, Basel, 2013.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[15] Alexander Hentschel and Barry C. Sanders. Machine learning for precise quantum measurement. *Phys. Rev. Lett.*, 104:063603, Feb 2010.

[16] Fischer HP. Mathematical modeling of complex biological systems: from parts lists to understanding systems behavior. *Alcohol Res Health. 2008;31(1):4959*, 2008.

[17] Diederik P Kingma. Variational inference & deep learning: A new synthesis. 2017.

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[19] Valerio Lucarini. Modelling complexity: the case of climate science. *arXiv preprint arXiv:1106.1265*, 2011.

[20] Pankaj Mehta and David J Schwab. An exact mapping between the variational renormalization group and deep learning. *arXiv preprint arXiv:1410.3831*, 2014.

[21] Smiruthi Ramasubramanian and Yoram Rudy. The structural basis of iks ion-channel activation: Mechanistic insights from molecular simulations. *Biophysical journal*, 114(11):2584–2594, 2018.

[22] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–1076, 2004.

[23] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[24] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, IL, 1949.

[25] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 7 1948.

[26] DJ Strouse and David J Schwab. The deterministic information bottleneck. *Neural computation*, 29(6):1611–1630, 2017.

[27] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[28] Evert PL Van Nieuwenburg, Ye-Hua Liu, and Sebastian D Huber. Learning phase transitions by confusion. *Nature Physics*, 13(5):435, 2017.

Popular summary

From when the first machines were built a question arose: "Is it possible to create a machine that learns and thinks like a human?" This question gives rise to the field of artificial intelligence. Many different sub-fields have since emerged. Areas of research are for example if a computer can recognise human faces on a picture, if a computer could generate a coherent written story or if a computer can determine if a patient is sick. One of these sub-fields is called *representation learning* and the core idea of this type of learning is to automatically break up the problem in smaller pieces before a decision is made. In an ideal world this would mean for example that a computer learns that a handwritten 9 is made up out of 2 parts. Namely, a circle and a stick. This process is shown in figure 5.1. Of course, this is not what really happens, but this concept is what drives the intuition for developing these kind of algorithms.

Figure 5.1: The ideal representation of a handwritten 9.

In the field of representation learning there are many different algorithms that try to mimic this type of learning. A question that one might ask is: "How do these algorithm work?" or "Why does this algorithm work?". These algorithms learn in steps and in each step a certain quantity is computed, that is uniquely determined by which algorithm you choose. This quantity determines how well the algorithm has learned the task so far. For the algorithm to perform well this quantity sometimes needs to be large and sometimes it needs to be small. So after each step we look if we can improve the calculated quantity and change the parameters of the algorithm accordingly. In this thesis we showed that all these unique quantities can be derived from only 4 distinct numbers. This means that if we want to answer the two questions above, that we do not have to look at every algorithm individually necessarily, but can analyse them all together with only these 4 numbers.

Let us take the example of figure 5.1 to get an idea of what this quantity looks like. First we have to get used to some terms. We call the actual picture of the handwritten 9 the input X. The outcome would then be that the computer knows that this picture is the digit 9. That outcome is what we denote by Y. The circle and stick on the right is what we call the representation Z of X. This representation is useful, because it allows for the computer to learn the idea behind what a 9 is, but it does lose some information of the original input. Reconstructing a 9 form a circle and stick can be done in many different ways. To recover the original picture the exact same way as before is unlikely. Similarly, saying that we have seen a 9 from just a circle and a stick will sometimes result into the wrong answer. It could just as well be a 0 and a 1 written closed together.

These two pitfalls give rise to 2 out of the 4 terms needed and can be described as follows:

- The first problem has to do with the loss of information when trying to reconstruct X from Z. This is called the *distortion*.
- The second problem resulted from Z not having complete information of the outcome Y. This is called the *classification error*.

Of course, we feed the algorithm not only 1 example. We try to give it as much as possible examples as possible. For each example the algorithm calculates the distortion and classification error and uses it to improve itself. The third number is similar to the distortion, but it works just the other way around. Instead of looking how much information is lost when reconstructing X from Z, it is the amount of information that is retained in Z from X. This number is called the *Rate*.

Finally, each algorithm has a set of parameters that determine what decisions it makes. The third number that we are interested in describes how much information the parameters incorporate about the complete data set. This quantity is called the *relative entropy*. A small summary of the terms:

- The first number is the *distortion*, the amount of information lost when rebuilding X from Z.
- Secondly, we have the *classification error*, the amount of information Z has about the outcome Y.
- The third term is the *rate*, the amount of information that Z is allowed to have from X.
- Finally, we have a number called the *relative entropy*, which tells us how much information the parameters of the algorithm have about the full data set.

It turns out that these 4 numbers are enough to describe how a vast amount of algorithms work in representation learning. You recover the specific algorithms again by focusing on 1 or more of the numbers. This is the main result of this thesis and opens up the way of new analyses of how and why certain representation learning algorithms work the way they do.