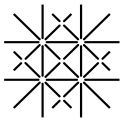


University of Basel

Estimating Cause-Effects with the Deep Information Bottleneck

Master Thesis

Natural Science Faculty of the University of Basel
Department of Mathematics and Computer Science
Biomedical Data Analysis

Examiner: Prof. Dr. Volker Roth
Supervision: Mario Wieser and Sonali Parbhoo

Yue Xu
yue.xu@unibas.ch
2015-062-904

November 15, 2019

Acknowledgements

I would like to thank Professor Volker Roth for giving me the opportunity to work on this interesting topic.

I would like to thank both of my supervisors for offering me advice.

Furthermore, special thanks go to my supervisor Mario Wieser for not only providing so many feedbacks and support, but also for doing the proofreading for me.

Additionally, I would like to thank my family, friends and peers for encouraging me.

Finally, I would like to give my sincere thanks to my father and Yingni.

Abstract

Estimating cause-effects is essential in various application fields, such as offering the most effective medication treatment for a patient. In this thesis, we focus on evaluating the causal inference of an intervention in the presence of confounding. With a general discriminative model, especially with supervised learning, it only allows us to identify whether a treatment needs to be conducted or not. Rather than solving the standard classification or regression problem with the deep neural networks, we adopt the more powerful generative models, such as Variational Autoencoder (VAE), for analyzing the causal relationship between events in terms of treatment decision and treatment effect.

However, similar to a fundamental element in machine learning and statistics, extracting relevant factors from complex data becomes the most crucial task for causal inference due to a high-dimensional input of the biological dataset. We propose an approach based on the generative model of deep Information Bottleneck (IB) under the decision-theoretic framework in order to identify the latent confounding, which is called the Cause-Effects deep Information Bottleneck model (CEIB). The confounders we would like to obtain, they are the latent variables regarding the genetic information collected from patients, which could potentially affect both treatment and effect. More specifically, our approach aims to reconstruct a low-dimensional latent space, which is containing the essential factors extracted from the input dataset for predicting the Average Causal Effect (ACE).

Our model is evaluated on a real-world dataset of the Infant Health and Development Program (IHDP). The information curve is experimentally analyzed with the purpose to select the best compression parameter of β-value for CEIB, since this curve shows us a "shortcut" for accessing all the model performances along with the β changes. Therefore, the optimal performance appears to be a sufficient statistic, which provides a guarantee for model reliability. Finally, we will show that our model is significantly more robust and flexible than existing methods and therefore will achieve the state-of-the-art performance regarding measured confounding.

Contents

Abstract ii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Goal . . . 1
- 1.3 Thesis structure . . . 2

2 Related Work 3

3 Background 5
- 3.1 Information-Theoretic Concept for Machine Learning . . . 5
  - 3.1.1 Entropy . . . 5
  - 3.1.2 Mutual Information . . . 7
  - 3.1.3 Rate Distortion . . . 9
  - 3.1.4 Information Bottleneck Principle . . . 11
- 3.2 Foundation of Deep Learning . . . 13
  - 3.2.1 Neuron Model . . . 13
  - 3.2.2 Activation Function . . . 14
  - 3.2.3 Multilayer Perceptron . . . 15
  - 3.2.4 Cost Function . . . 16
  - 3.2.5 Gradient Descent . . . 17
  - 3.2.6 Feedforward Training Process . . . 18
  - 3.2.7 Back Propagation . . . 19
- 3.3 Deep Generative Model . . . 23
  - 3.3.1 Autoencoder . . . 23
  - 3.3.2 Variational Autoencoder . . . 26
    - 3.3.2.1 KL Divergence . . . 26
    - 3.3.2.2 Approximate Inference . . . 27
    - 3.3.2.3 The Reparameterization Trick for Backpropagation . . . 32
- 3.4 Deep Information Bottleneck . . . 35
- 3.5 Deep Variational Information Bottleneck . . . 36
  - 3.5.1 The Variational Bound Examination . . . 37
  - 3.5.2 The Reparameterization Trick for deep VIB . . . 39
- 3.6 Causality . . . 41
  - 3.6.1 Cause-Effects . . . 41

3.6.2 Graphical Representation . . . 42
3.6.3 Models . . . 42
3.6.3.1 The Decision-Theoretic Model . . . 43
3.6.3.2 The Functional Model . . . 43
3.6.4 Confounding . . . 43
3.6.5 Conditional Independence . . . 44
3.6.6 Markov Property . . . 45
3.6.7 Confounder . . . 45
3.6.8 Causal Inference . . . 46
3.6.9 Average Causal Effect . . . 47

4 Model and Implementation **49**
4.1 Cause-Effect Deep Information Bottleneck Model . . . 49
4.1.1 Measured Confounder . . . 49
4.1.2 CEIB Model Structure . . . 50
4.1.3 CEIB Objective . . . 51
4.1.4 Estimate ACE with CEIB Model . . . 51
4.2 Implementation . . . 52
4.2.1 CEIB Model Implementation . . . 52
4.2.2 Reparameterization Trick for CEIB . . . 54

5 Experiments **56**
5.1 Information Curve . . . 57
5.1.1 Results . . . 57
5.2 IHDP Performance . . . 59
5.2.1 Results . . . 60

6 Conclusion **62**

Bibliography **64**
Appendices . . . 67
A Calculation of KL divergence of $-D_{KL}(q_\phi(z|x) \parallel p_\theta(z))$ (Gaussian case). . . . 67
B Infant Health and Development Program: Baselines . . . 68

Chapter 1

Introduction

1.1 Motivation

The causal effect takes a prominent role in many applications, such as precision medicine, or advertising placement. Since understand how a patient's health can be influenced by a certain type of medication, estimating causal effects of an intervention is becoming significant in the process of developing the proper treatments for patients. In statistics, the definition of causal inference (Dawid 2015) is a method to determine the causal relationship between variables. To find the cause-effects between variables is commonly more challenging than finding the correlation in between. For instance, with the increase in ice cream consumption, the number of drowning tends to increase as well, there is a positive correlation between these two variables. Obviously, ice cream selling forbidden cannot lead to decrease drownings. Thereby, there are existing other variables, namely the confounder or latent variables, which might cause both two variables. In this case, the confounder tends to be the outdoor weather temperature. Since more people go swimming when the outdoor temperature is getting higher, which result in an increase in drowning accidents. The Directed Acyclic Graph (DAG) (Pearl 2000) includes the causal links which are representing the dependence (causal relationships) between events.

The confounder might be hidden in a real-world environment, which leads to the uncertainty factors can be fluctuated randomly in a wide range in medical domains, for example, personal financial status or environmental factors in attempting to affect the medication effect potentially. It is difficult to collect all the irrelevant individual data, in particular, does not contain the clinical factors. Therefore, in this thesis, we only focus on the measured confounding without hidden features for our model.

1.2 Goal

The most crucial aspect of solving causal inference is to identify the confounder, which is influencing both treatment and outcome components simultaneously, and subsequently correct for it afterwards. In order to capture the cause-effects relationship between the observed data, the causal inference model is very useful to determine the important features among

the collected clinical factors, or evaluate the best medical treatment effect from the provided prescription medicines for a specific patient. However, due to a high-dimensional input data of the biological data sets for the patients (e.g., blood test, gender, gene sequence), we are not able to extract the important features from it directly. Benefit from the progress of unsupervised learning and the continuous improvement of the data processing capabilities in the modern age, many state-of-the-art techniques have been adopted in order to overcome the computational challenge. It allows the Artificial Neural Network (ANN) (Hornik et al. 1989) to perform the unsupervised learning, for the purpose of transforming a high-dimensional input into a low-dimensional representation of confounder. Whilst many latest studies which providing promising results, two of them are Information bottleneck (IB) criterion (Tishby et al. 2000); (Alemi et al. 2016) and Variational Autoencoder (VAE) (Kingma & Welling 2013). To our knowledge, there are only a few studies have been conducted to determine if these models could benefit causal inference.

Based on the confounding builds on the two data-generating interventional regimes (Dawid 2015), under which the available data is imposed on whether the treatment is conducted or not. Consequently, we address this causal inference problem by introducing IB theory, for the sake of overcoming the challenge of high-dimensional input data. Motived by the existing researching of the casual model build on VAE framework (Louizos et al. 2017), it allows to predict the Individual Treatment Effect (ITE). In this thesis, we will focus on the IB principle and aim to learn a low-dimensional representation from input data; it serves as the extract features from the genetic information that influence both treatment and outcome variables. Compare to the VAE model from a different perspective, a trained model of causal effect inference builds on the IB framework with deep neural architectures. It is capable of recovering ACE and predicting the best clinical treatment for the patients. In the experiments, it will be conducted on a dataset from the Infant Health and Development Program (IHDP). (McCormick et al. 2006)

1.3 Thesis structure

The thesis is organized as follows: Following a short overview of existing related work, Chapter 2 describes a general background for the foundation of deep learning and the IB framework. The Generative model of VAE and deep IB are described in details. Furthermore, we introduce the concepts of causality based on DAG in Chapter 3.6. In Chapter 4, we build the causal inference model with the IB framework (CEIB) and the implementation is presented as well. The experiment results obtained in Chapter 5 in terms of the information curve and average treatment effects. Finally, the connection between VAE and IB model regarding the causal inference is drawn, and a conclusion and the outlook of future work is given.

Chapter 2

Related Work

Recently, (Shalit et al. 2017) introduces a method of Treatment-Agnostic Representation Network (TARNet) to solve the causal inference, which is a feedforward network and focusing on "no-hidden confounders" assumption. Since the distributions of the two treatments are different, each example is either with treatment $T = 0$ or treatment $T = 1$. Therefore, it is not allowed to infer the active treatment effect ($T = 1$) for a patient who is receiving control treatment ($T = 0$) in the training data. This model seeks for a transformed representation Φ; it plays the role of regularization. In order to enforce the same distributions for two treatments, Φ aims to penalize the distributional treatment distance in the representation space. In other words, by adjusting the weight of the penalty term, the TARNet learns non-linear representations and hypotheses, which is balancing the tradeoff between the prediction accuracy and treatment group imbalance. However, this algorithm only adopted for predicting the Individual Cause Effect (ICE) on the basis of observational data, instead of ACE estimation.

In the relevant recent work by Louizos et al. (2017), which is building on the framework of VAE for solving the causal effect inference with the proxy variables. This approach is called Causal Effect Variational Autoencoder (CEVAE). It is a generative model; the general idea of this approach is attempting to estimate ICE with a latent space model. This work particularly proposes a solution to the causal inference with the cofounders are hidden or unmeasured. The architecture of CEVAE with the deep latent-variable model consists of a pair of DNNs. The inference network is adapted as the encoder for estimating the posterior of $q(z, t, y|x)$, and the decoder of the model has a similar architecture to the TARNet and aims to recover the joint distribution of $p(x, z, t, y)$ approximately. This optimization process used to obtain the latent space z for identifying ACE and ICE, it also belongs to the maximum-likelihood based method.

Parbhoo et al. (2018) propose a method to perform casual inference with the CEIB model. This method focusses on training the encoder with the partial covariate in order to learn a discrete representation of the latent space. The decoder of this model also follows the similar structure of TARNet to recover ACE. They discuss the confounding information from both measured and hidden point of view respectively, and applying the CEIB model to the causal inference with the dataset has only partial covariate available for testing.

Our work is related to the proposed Gaussian IB approach (Chechik et al. 2005). In this work, rather than discrete clusters, they apply Gaussian variables to the IB framework for obtaining a continuous representation of latent space. With the general IB objective in a principled manner (Tishby et al. 2000), they demonstrate that the optimal representation of IB turns out to be a noisy linear projection on the eigenvectors of the normalized regression matrix $\Sigma_{x|y}\Sigma_x^{-1}$, which is on the basis of canonical correlation analysis. Thus, an analytic Gaussian projection of the optimal solution under the IB framework can be obtained. Additionally, they investigate the ranks of the matrix in solutions in terms of the compression level of the IB method. Finally, they analyze the information curve in the perspective of scaling each eigenvector, which is depicted by the relationship between eigenvectors and the tradeoff controlling parameter β.

Chapter 3

Background

3.1 Information-Theoretic Concept for Machine Learning

3.1.1 Entropy

Before diving into the concept of IB principle, we will start with some essential background knowledge about the information theory. As mentioned above, with a particular focus on the use of information. Rather than capturing the value in the system, the information theory is transmitting the information to the recipient. In Shannon's theory, the information of the message is coded by binary digits, which is called bits. For example, a signal of the weather forecast will be encoded into bits for broadcasting. Suppose the weather has four possible states, we can naively encode the weather information shows in Figure 3.1. Each of the weather symbols equally contains two bits, the sequence of bits concatenated together to form the encoded string (signal), or namely codewords to send to the receiver.

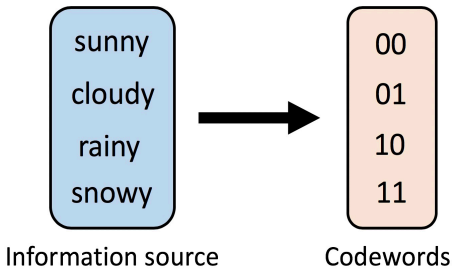


Figure 3.1: An example of encoding information into binary digits (bits) in a general communication system.

However, this is not an optimal solution for transmitting messages due to the frequency of information occurrence. In general, the message from the weather station is not stable under these four conditions (for example, the weather with a bigger chance of being sunny or cloudy compare with snowy). Suppose a certain cost of transmitting information for every bit is existing, in order to achieve the same performance of information transmitting purpose, and meanwhile spend the minimal cost for transmitting, a possible solution is to decrease the length of the bits according to the frequency of the encoded message occurrence (probability

of occurrence). We make use of the following diagram to visualize the bits (see Figure 3.2). The horizontal axis denotes the length $L(x)$ of the coding for each state, which is measured by bits, and the vertical axis denotes the probability distributions $p(x)$ of according weather states. A naive approach for demonstrating the weather forecast shows on the left-hand side in Figure 3.2, which is coding each information source by 2 bits regardless of the weather states frequency. The other strategy on the right-hand side, which is constructing a fairly good coding of the message corresponding to state probability. Since we reduce the length of the information which is appearing more frequently (for example, "sunny"), the second coding method turns out using the smaller area in total. This implies that the average length of the coding message can be compressed from 2 bits to 1.75 bits. The second improved coding method enables the communication system to transmit less information to express the same information source on average.

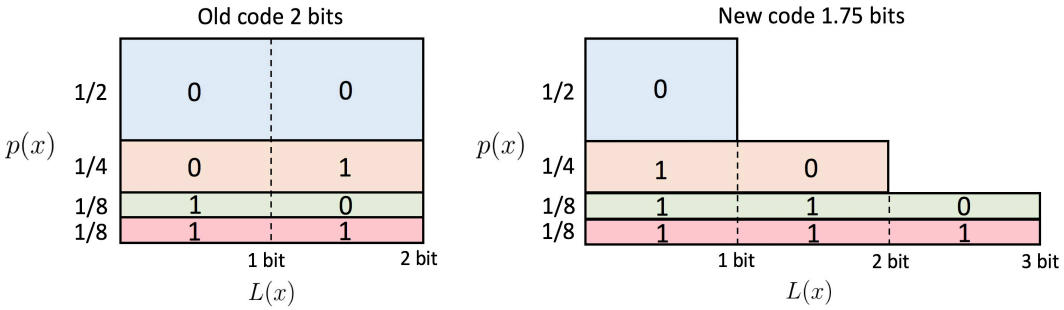


Figure 3.2: Visualize the coding messages into bits by different approaches. Different colours correspond to four weather states. The total area of all rectangles represents the length of bits required for coding a message. Reproduced images adapted from: http://colah.github.io/posts/2015-09-Visual-Information/.

However, there is a tradeoff between the information lengths which are assigned to the different coding states. As it is shown in Figure 3.2, we need to assign more bits to the states of rainy and thunderstorm rather than sunny and rainy, the reason due to the decodable property of the message. The receiver can uniquely decode the signal in order to avoid ambiguity, which is a primary requirement for the technique of the information theory. Since only binary digits 0 and 1 can be used for coding message, a reasonable solution is to ensure that there is no coding message can become the prefix of another one. For instance, it will cause ambiguous decoding, if the receiver encountered with the encoding strings 10 and 1011. The unclear prefix 10 in the string 1011, obviously, it will confuse receiver to distinguish between them. We can compute the cost according to the length of the coding message (see Figure 3.3). If we can only use 2 bits (length 2) to code a message, there is a fifty percent chance the coding message will begin with 0. If the length allows to 3 bits (length 3) for coding, take 01 for example, there is a quarter chance to establish the code can begin with 01. Therefore, in general, which implies that the cost for coding will decrease exponentially with the length of the message. The cost of coding one message can be written as:

$$\text{Cost} = \frac{1}{2^L} \tag{3.1}$$

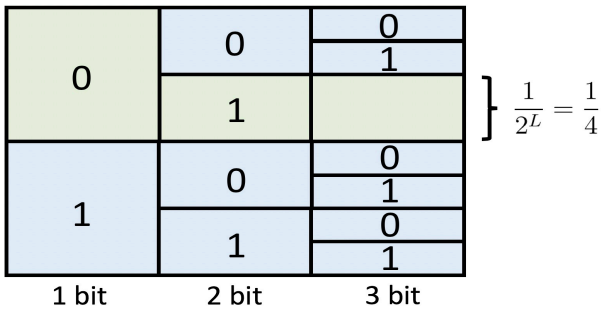


Figure 3.3: The relationship between the message coding length L and the cost we pay for coding it. Reproduced image adapted from http://colah.github.io/posts/2015-09-Visual-Information/.

Since the cost for the coding message is $\frac{1}{2^L}$ with message length of L, we can invert it to compute the message length with a fixed amount of cost:

$$L = \log_2 \left(\frac{1}{\text{cost}} \right) \tag{3.2}$$

The methodology of the optimal coding is assigning the portion of bits to an event which is equal to the probability distribution of this event $p(x)$. Therefore, the optimal coding length L for each event x can be written as:

$$L(x) = \log_2 \left(\frac{1}{p(x)} \right) \tag{3.3}$$

Finally, we can calculate the average encoding size in bits by using the minimum encoding size (represented by the area in Figure 3.2) which is known as the entropy:

$$H(p) = \sum_x p(x) \log_2 \left(\frac{1}{p(x)} \right) \tag{3.4}$$

where it is commonly written as:

$$H(p) = -\sum p(x) \log_2 (p(x)) \tag{3.5}$$

In summary, the entropy is commonly interpreted as quantifying the level of uncertainty or unpredictable of the information. When a message appears with a high value, it is indicating that this message has a much smaller probability of occurring. Since the rare information is infrequently happening than the receiver expects, it contains more specific information and tells us a more useful message. If the entropy is close to 0, it means that we receive a message with the state of a high probability that we can expect in particular, which is “meaningless”.

3.1.2 Mutual Information

Mutual information is usually connected to the KL divergence; it defines the relationship between two discrete random variables X and Y, which can be formally written as:

$$
\begin{aligned}
I(X;Y) &= \mathrm{D}_{\mathrm{KL}}(p(x,y)\|p(x)p(y)) \\
&= \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}
\end{aligned} \tag{3.6}
$$

Where $p(x, y)$ is a joint distribution of two variables, the marginal distribution of X and Y are denoting by $p(x)$ and $p(y)$. It measures one variable contains how much information about the other variable on average. Simply, it allows us to infer the information about Y if we had a variable X as they are sharing the mutual information. For example, X denotes a result of rolling a (6-sides) dice, Y represents whether a rolling result to be odd or even. They are sharing the same information, as the rolling result with a side number X also contains information Y regarding odd or even. On the contrary, suppose X and Y both represent the rolling results related to side numbers of two dices respectively, then these two variables are called statistically independent, and there is no mutual information shared between them, which is equal to 0. $I(X;Y)$ also represents a reduction of the uncertainty of X, since X contains the knowledge about Y. We can rewrite it as:

$$
\begin{aligned}
I(X;Y) &= \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \\
&= \sum_{x,y} p(x,y)\log\frac{p(x|y)}{p(x)} \\
&= -\sum_{x,y} p(x,y)\log p(x) + \sum_{x,y} p(x,y)\log p(x|y) \\
&= -\sum_{x} p(x)\log p(x) - (-\sum_{x,y} p(x,y)\log p(x|y)) \\
&= H(X) - H(X|Y) \\
&= H(X) + H(Y) - H(X,Y)
\end{aligned} \tag{3.7}
$$

As it is shown in Figure 3.4, mutual information indicates the overlap region in the diagram. It turns out to be the region by subtracting the conditional entropy $H(X|Y)$ from its corresponding entropy in X.

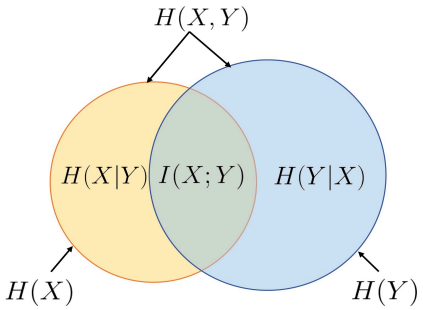


Figure 3.4: An illustration of the relationship between mutual information $I(X;Y)$ and joint entropy $H(X,Y)$. Visualize the conditional entropy of $H(X|Y)$ and $H(Y|X)$. $H(X)$ denotes the entropy in X, and $H(Y)$ denotes the entropy in Y.

For the continuous random variables, a differential entropy $h(\cdot)$ can be applied to generalize Equation 3.7:

$$\begin{aligned} I(X;Y) &= \int p(x,y) \log \frac{f(x,y)}{f(x)f(y)} dx dy \\ &= h(X) - h(X|Y) \end{aligned} \tag{3.8}$$

As mentioned above, the mutual information is intimately related to the KL divergence. It also measures the divergence or "distance" between two distributions:

$$I(X;Y) = D_{KL}\left(P_{XY}(x,y) \| P_X(x)P_Y(y)\right) \tag{3.9}$$

In another perspective, mutual information can be interpreted as the KL distance between the joint distribution of $P_{XY}(x,y)$ and the product of the independent marginal distributions of $P_X(x)P_Y(y)$. Since we discussed the non-negative property of KL divergence in subsubsection 3.3.2.1, this property for mutual information $I(X;Y)$ also holds. The difference is that the mutual information is symmetric: $I(X;Y) = I(Y;X)$. Minimizing the KL divergence aims to decrease the "distance" between two distributions. However, this also implies to maximize the mutual information and create more "overlap region" between two distributions.

Another essential property of mutual information is the Data Processing Inequality (DPI) (Cover & Thomas 2012). Suppose there is a Markov chain $X \to Z \to Y$ which is formed by three variables:

$$I(X;Z) \leq I(X;Y) \tag{3.10}$$

where the mutual information throughout the Markov chain can only decrease during the post-processing, the information processing always performs in a loose way. Simply, the variable Z cannot contain more information about X than Y contains about X. The inequality is proven by entropy calculation as follows:

$$\begin{aligned} I(X;Z) &= H(X) - H(X|Z) \\ &\leq H(X) - H(X|Y,Z) = H(X) - H(X|Y) = I(X;Y) \end{aligned} \tag{3.11}$$

where the inequality for entropy follows, it can only perform a deduction way when conditioning on an extra variable (Y or Z).

3.1.3 Rate Distortion

IB method is considered as an extension of the rate distortion theory. The relationship between these methods was proposed by Tishby et al., in 1999. The rate distortion theory aims to analysis a lossy source compression problem, which is capturing the tradeoff between the distortion, namely the average reconstructed representation of a signal, and the compression rate of the signal.

Assume we have two discrete random variables X and T; X denotes the original signal with a finite set of possible values $\mathcal{X}$. T denotes the compressed representation (or quantized codewords) of the X with a finite set of representative values $\mathcal{T}$. Formally, the compression defines as a (possibly stochastic) function which maps each value $x \in \mathcal{X}$ to its corresponding

compressed codebook value $t \in \mathcal{T}$. This mapping can be implicitly characterized through a conditional distribution $p(t|x)$ (Slonim 2002). The essential role of mapping for the rate distortion theory due to the fact that a perfect representation of X is redundant and unnecessary, this approach of mapping seems to make more sense to get a compressed representation with reasonable amount of information in general.

The quality of compression can be determined by this mutual information $I(T;X)$; the lower values of it implies the better quality. The extreme case appears when $I(T;X) = 0$, which is the compressed representation T only contains a single value. It measures the compactness of the new compressed representation T, which can also be interpreted as the complexity of the information.

However, the quality merely measured with this information rate (Tishby et al. 2000) is not a completely characterizing good quantization, since it can always drop down to 0 without containing any relevant information from the original signal. Therefore, we need to define a distortion function $d : \mathcal{X} \times \mathcal{T} \rightarrow \mathcal{R}^+$ with some additional constraints build upon the information rate. Generally, the smaller distortion value indicates better-compressed representations.

The distortion function $d(x,t)$ defines by taking inputs x and t, and the output value implies the different (distortion) between x and t. Finally, we would like to measure the amount of losing information during the mapping process through a deterministic $p(t|x)$, the expected distortion defined by encoding:

$$D(p) = \sum_{x,t} p(x,t)d(x,t) \tag{3.12}$$

We aim to search for a balance between distortion and compression (information rate). Therefore, the threshold of information loss defined by a specific value D^*, in order to prevent the expected distortion leads to lose too much information $D(p) > D^*$. Based on the setup above, we are able to determine how much information can achieve maximum compression (information rate of T as low as possible) by not allowing to distort the input exceed the threshold D^* (distortion of D as small as possible). More specifically, in theory, in order to obtain the optimal distortion value D^*, we define a distortion function and it returns the output with best possible (lowest) rate can be achieved, in other words, the mutual information under the lowest compression (encoding) $p(t|x)$.

$$R\left(D^*\right) = \min_{p(t|x):\, D(p) \leq D^*} I(X;T) \tag{3.13}$$

where the mutual information $I(X;T)$ represents the amount of compression information in total. As shown in Figure 3.5, the rate distortion curve describes the relationship between the distortion values D and the information rate T.

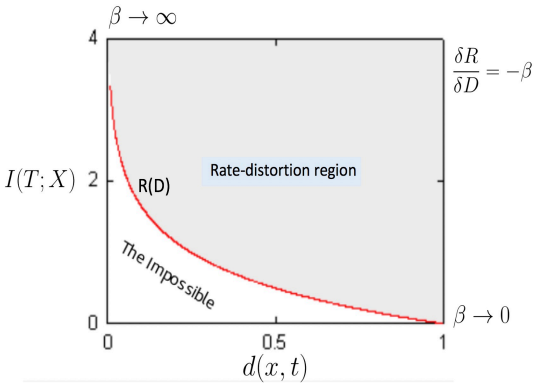


Figure 3.5: An illustration of the rate distortion function $R(D)$. It defines the monotonic curve regarding distortion D and the information compression (rate) T, this curve with a slope of $-\beta$. The region below the curve is unachievable, $R(D)$ denotes the minimal achievable compression-information. Reproduced image adapted from (Cover & Thomas 2012).

Therefore, targeting the rate-distortion function is converted to be an optimization problem which can be solved by the Blahut-Arimoto algorithm by introducing a Lagrange multiplier β, with the objective to minimize $\mathcal{F}[p(t|x)]$, which is subject to the normalization constraints of $\sum_t p(t|x) = 1$.

$$\mathcal{F}[p(t|x)] = I(T;X) + \beta D(p) \tag{3.14}$$

3.1.4 Information Bottleneck Principle

The original IB principle can be seen as an extension of the rate distortion theory in general, by additionally introducing another output variable Y. IB method aims to find an optimal compressed representation of input X; this "codewords" is denoted by T. In order to obtain such T, IB compresses the input information X, and meanwhile preserving as much information about Y as possible. In other words, the optimal compressed representation of T is extracting the most relevant information about Y from X. Since Y determines the relevant information in X needs to be preserved, we can specify a Markov chain: $Y \to X \to T$. The lossy compression of T is not capable of transforming more information than which is contained in the original data, due to the fact that T cannot produce new information about Y, it only depends on X. According to the DPI we discussed in subsection 3.1.2, for these three variables which are used to form the Markov chain, applying the definition of mutual information then we have:

$$I(X;Y) \geq I(T;Y) \tag{3.15}$$

Similar to the rate distortion theory, the purpose of IB principle is still minimizing the mutual information of $I(X;T)$, which is similar to the rate $p(t|x)$. However, unlike the constraint of rate distortion theory by using expected distortion $D(p)$, IB principle is restricted to the maximum information loss we can accept in Y when compressing X into T:

$$\min_{p(t|x):\, I(T;Y)\geq I^*} I(X;T) \tag{3.16}$$

where the information loss constraint can be measured by mutual information $I(T;Y)$, which is also leading to an optimization problem. IB principle can also defined to capture the tradeoff β between the pair of $I(T;X)$ and $I(T;Y)$ by minimizing the objective function:

$$\mathcal{L}[p(t|x)] = I(T;X) - \beta I(T;Y) \tag{3.17}$$

The objective is to find an optimal assignment with respect to the stochastic mapping $p(t|x)$. When we set $\beta \to 0$ as an extreme case, then $I(T;Y) = 0$, which implies that only the information compression term of $I(T;X)$ will be applied. The output information Y is no longer considered as the constraints, since the optimal assignment is $I(T;X) = 0$, which means that input variable X will be assigned to a single value, all of the relevant information related to Y is thrown away and there is no generalization bound existing. In contrast, $\beta \to \infty$ is indicating that T is forced to preserve all the information about Y, even the trivial and meaningless details of output Y need to be captured. In this case, X completely loses its compression ability.

More generally speaking, instead of compressing the input X itself into T in rate distortion theory, the key concept behind this original IB principle is introducing an additional output variable Y to represent the "information distortion", the compact representative of X is the product that we aim to obtain, which is denoting by T (see Figure 3.6). Therefore, For the interpretation of the IB principle intuitively, T performs as a bottleneck to squeeze out all the information irrelevant to Y from X, only the meaningful information which is related to Y in X remains. The optimal representation of T can be obtained by decreasing $I(T;X)$, which leads to the simplest statistic of $p(t|x)$. On the other hand, maximizing the constraints upon the predicting accuracy of $I(T;Y)$, which is equivalent to maximize the likelihood of prediction $p(y|t)$.

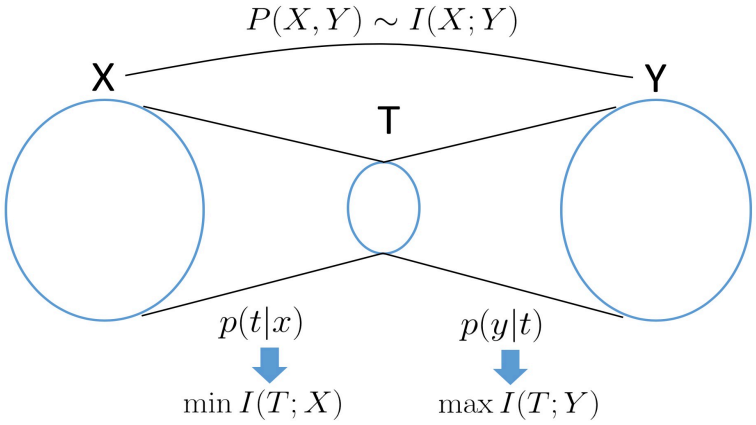


Figure 3.6: Given $p(x, y)$, for the IB principle, we replace the distortion $D(p)$ in the rate distortion theory to be the level of output Y accuracy $I(T;Y)$. Reproduced image adapted from (Slonim 2002)

3.2 Foundation of Deep Learning

Artificial Neural Networks (ANNs) has taken most of its inspiration from the biological neural networks that constitute animal brains (van Gerven & Bohte 2018). The architecture of ANNs attempts to simulate biological neurons with an artificial neuron, a so-called perceptron. The ultimate goal of the ANNs is to obtain an unknown mathematical function by training. In recent years, ANNs has been widely adopted to solve a variety of tasks, such as pattern recognition, medical diagnosis.

3.2.1 Neuron Model

Similar to the basis of biology, the units responsible for calculation in ANNs are called artificial neurons. The commonly used model for a neuron is the perceptron model (Minsky & Papert 2017). A perceptron is a fundamental component of a neural network. It consists of only one neuron. Figure 3.7 shows the graphical representation of a perceptron. The inputs of a vector $(x_1 ... x_n)$ transformed into a single output y, and each neuron receives multiple signals, they are proceeding to the next neuron after processing. The neurons are connected via weighted inputs. The single neuron takes inputs x and multiplies each of it by some specific weight w for that input. The weights for the inputs initialized as random values, by adjusting these weights during training, this process enables ANNs to learn the mathematical function gradually. The perceptron model also consists of a bias term of b and an activation function.

The perceptron model can be represented mathematically:

$$y = f(\sum_{i=1}^{n} x_i w_i + b) \tag{3.18}$$

where the combination function defines as $c = \sum_{i=1}^{n} x_i w_i + b$ with $i = 1, ..., n$, it is computed by the weighted sum $\sum_{i=1}^{n} x_i w_i$ plus the bias term b. The weighted sum is the linear combination of the weights w multiply by the inputs x. The output y is obtained by applying the weighted sum to the chosen activation function f. Each neuron owns its bias term, which is also learnable as weights. The purpose of the extra bias term is introduced before applying the activation function, is to increase the model flexibility in fitting the data. Since the bias expands the range of the weighted sum, which is encouraging more of the neurons consider as being activated. Intuitively, the bias term allows a linear separator (perceptron) $XW + b$ to gain the ability to shift the decision boundary left or right.

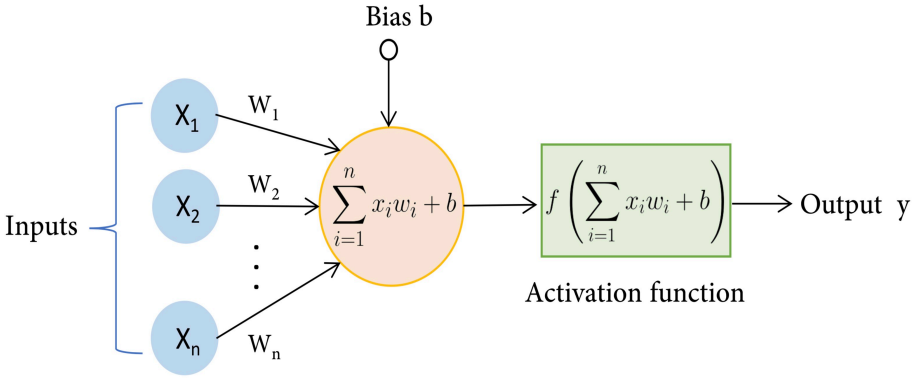


Figure 3.7: Illustration of a perceptron model with a single neuron.

3.2.2 Activation Function

There are mainly two types of activation function: linear and non-linear activation functions. In order to confine the result of the perceptron without going infinity, we would like to map the outcome into a certain range via non-linear activation functions. The activation function can be altered according to the output of a particular neuron. On the other hand, instead of applying monotonic activation function to determine the output y as 0 or 1 simply, the output of activation function (sigmoid function in this case) in Equation 3.19 is mapped between 0 to 1, it is especially useful for models which we adopt in order to predict the probability as an output. It is commonly used also due to this function is continuous and differentiable, which means the slope of the sigmoid function curve exists between any two points, and it is an essential property for training with gradient descent algorithm. There are several activation functions in the neural network are used in general:

$$\text{sigmoid(x)} = \frac{1}{1+e^{-x}} \tag{3.19}$$

$$\tanh(\text{x}) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{3.20}$$

$$\text{ReLU(x)} = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases} \tag{3.21}$$

$$\text{Leaky ReLU(x)} = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \tag{3.22}$$

$$\text{ELU(x)} = \begin{cases} \alpha\,(e^{x}-1) & x \leq 0 \\ x & x > 0 \end{cases} \tag{3.23}$$

$$\tag{3.24}$$

As shown in Figure 3.8, another default activation function widely adopted by ANNs is the Exponential Linear Unit function (ELU) shown in Equation 3.23, since it tends to achieve the better performance for the used model. It is a non-linear function and similar to the Rectified Linear Unit (RELU). ELU only converts the negative inputs to be a positive output with an extra α and directly outputs itself if it is a positive input.

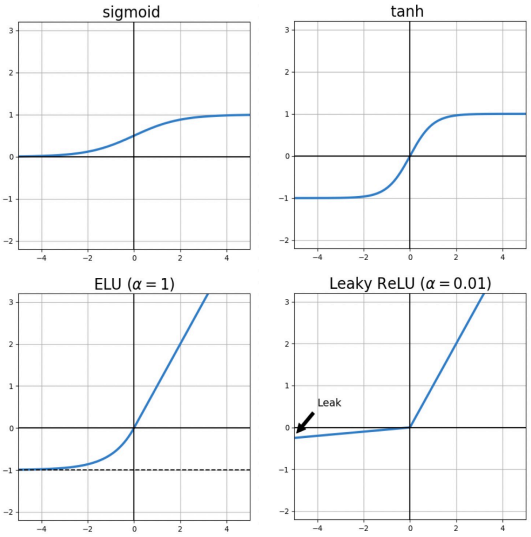


Figure 3.8: The plot of activation functions. The tanh activation function is similar to the logistic sigmoid, and its range is expanded from (-1 to 1), which is mainly used for the classification between two classes. The Leaky ReLU increases the lower bound of the ReLU function commonly with $a = 0.01$.

3.2.3 Multilayer Perceptron

However, a Single-Layer Perceptron (SLP) can only be used to implement on the linearly separable functions, which is also known as a linear binary classifier, and it is the simplest type of ANNs. To combat the limitation of SLP for solving the non-linear problem, the architecture of a Multilayer Perceptron (MLP) refers to numerous neurons and the connections between them. The MLP describes as a finite acyclic graph, and the input signal only can be forwarded to the next layer, which is a feed-forward network and each unit in a layer requires to (fully) connect to the units in the previous layer.

The Deep Neural Networks (DNNs) is an ANNs with multiple hidden layers between the input and output layers (Schmidhuber 2015). Differ from the multilayer perceptron network, DNNs has many more hidden layers for making the model more complex, and the layers of DNNs can be partially or sparsely connected rather than fully connected, and each unit in it is not necessary to connect every other node. Figure 3.9 depicts an example of DNNs; each unit in the ANNs is considered as a perceptron model.

To summarize, after passing the combination function $XW + b$ into activation function $f(\sum_{i=1}^{n} x_i w_i + b)$ for processing, each neuron's output value is forwarded to the neurons in the next layer. The connection is assigned with a large weight infers this neuron is more relevant to the previous corresponding neuron. The extra bias term allows the activation functions to shift in order to fit the data.

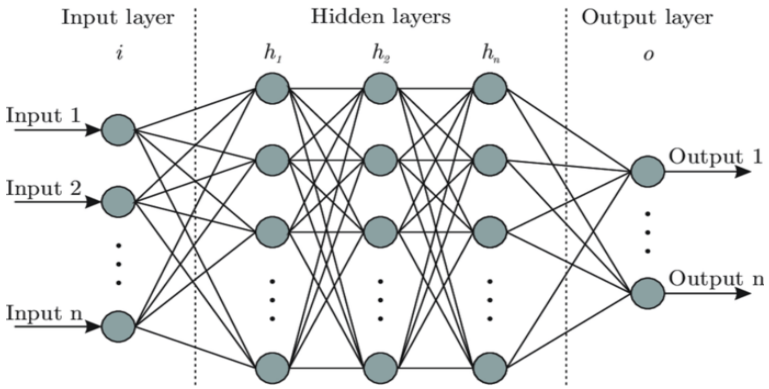


Figure 3.9: Visualize a classical ANNs with three hidden layers. The first hidden layer h_1 makes a decision based on the values from the input layer. Hidden layer h_2 makes a decision based on the outputs from the previous hidden layer h_1; it is processing at the higher and more abstract level. Image adapted from (Bre et al. 2018).

3.2.4 Cost Function

The evaluation performance of the machine learning model is directly related to the cost function. Given the input x and outcome y, the ultimate goal of a supervised learning problem is to construct a hypothesis or so-called functions of inputs $f(x)$ for predicting the observed target of y. The cost functions can be described as how well the predicting function $f(x)$ fits the available given dataset. For instance, the target y of a classification task is the desired labels for each data point x. The essence of training is the process of minimizing the cost functions, and the smaller values of cost function indicate the better fit.

In short, the essential role of the cost function is measuring how well the model that we predict by calculating a difference or distance between the predicted outcome value $\hat{Y}$ and the actual outcome Y. It is also represented as an objective function for algorithms. Depending on the algorithms, we have different types of cost functions according to its ability to estimate the relationship between input and output.

According to the different problem-types, the quadratic cost function is commonly used as the loss function for regression purpose, also known as Mean Squared Error (MSE):

$$L = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{3.25}$$

where for the number of n data points, it calculates the average squared deviation between the predict values $\hat{Y}_i$ and the ground truth of Y_i.

Another loss function widely used for either binary classification or multiclass classification problems is Cross-Entropy(CE):

$$L = \frac{1}{n}\sum_{i=1}^{n}(Y_i \ \log \hat{Y}_i + (1 - Y_i) \ \log(1 - \hat{Y}_i)) \tag{3.26}$$

If the number of classification class is bigger than 2, then we define the multi-class (categorical) cross-entropy as the sum of the differences between each separated true label distribution and the predicted label distribution.

$$L = -\frac{1}{n}\sum_{k}\sum_{i=1}^{n} Y_i^{(k)} \log \hat{Y}_i^{(k)} \tag{3.27}$$

where k denotes the class label. The larger value of CE indicates the neuron tends to learn faster. Therefore, the advantage of using CE as cost function allows the algorithm to gain better performance.

3.2.5 Gradient Descent

The mechanism for a training process is minimizing the loss function by updating the ANNs parameters of weights and bias step by step. Gradient Descent (GD) is the most popular algorithm for this purpose. GD is an iterative optimization algorithm; it keeps tracing of the error and evaluating how well the model fits the data.

Once we define our loss function, in order to make sure the local minimum is existing, it is necessary to apply the loss function to the activation function, which is required to be differentiable. This means its partial derivatives can set to be 0 for calculating the minimum of loss function. When there exists no closed-form solution for the derivative equation, such a problem can be solved by GD. The gradient (slope) in Figure 3.10 denotes the ratio of error changing to the weight changing of the agent. Searching for the gradient is equivalent to the search for partial derivatives of the loss function. The strategy for GD shows that the weight keeps moving to a neighbouring position along the negative gradient of the loss function, since which is the direction with the steepest descent — repeating this process until it converges at the minimum loss point, and it leads the gradient approaches to zero.

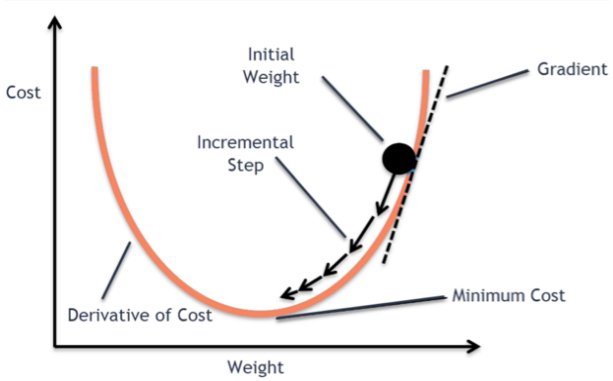


Figure 3.10: Illustration of GD algorithm. Image adapted from (Russell & Norvig 2016)

In short, with the help of the cost function, GD enables the ANNs to correct its behaviour in order to minimize mistakes. For a small number of examples, we calculate the partial derivatives of each training dataset for updating the model parameter, which is the batch GD. When the number of training samples increases, the computing complexity would be increased significantly as well. In deep learning, instead of counting it each time when the new data adding into the set, we only consider a single training point at a time, which is called Stochastic Gradient Descent (SGD). The idea of SGD is by using the new adding data for updating the parameter of the model that is trained with old training samples.

Since the loss function is already defined in subsection 3.2.4, here we take MSE (see Equation 3.25) as our cost function for the backpropagation demonstration purpose. Combining with the GD in subsection 3.2.5, the main task of backpropagation is computing the partial derivative of the weight Δw_i based on the forward result after i-th iteration. As shown in the algorithm 1, the newly updated weight $*w_i$ is equal to the old weight minus Δw_i.

Algorithm 1: Gradient Descent Algorithm

$W \leftarrow$ weight is randomly picked initially
loop until convergence **do**
  for each w_i **in W do**
    $* \; w_i \leftarrow w_i - \Delta w_i$

3.2.6 Feedforward Training Process

The perceptrons have been around since the 1940s, but the single-layer neural network is only capable of solving the linear classification tasks. The weakness of perceptrons can be easily proven by a simple classification task such as XOR. Until the late 1980s, the ANNs becomes a major topic of machine intelligence due to the arrival of backpropagation algorithm for a multilayer network, and the neuron network also includes the extra hidden layers between the input and output layer. Each iteration of the training process consists of feedforward and backpropagation phases — a perceptron (neuron) as the building block of the

neuron network for learning.

During feedforward training, each neuron takes the input and generates a prediction, and continuously feeds the predictions to the neurons in the next layer until the network generates a predicted output eventually. Based on the supervised learning system of ANNs, we can compute the deviation of the given ground truth from our prediction value at the last layer. The error measured by the feedforward training keeps iteratively passing to the input layer in the backpropagation training, it aims to search for the optimal values of perceptron weights, in order to generate the most accurate prediction.

An example of feedforward ANNs is available in Figure 3.12, and the input dataset is fed into the first input layer, which is processed by the perceptron system. It is shown in subsection 3.2.1. The neurons generate outputs of the hidden layer:

$$y_3 = f_1(W_1 X_1 + W_4 X_2 + b_1) = f_1(\Sigma_1) \tag{3.28}$$
$$y_4 = f_2(W_3 X_1 + W_2 X_2 + b_1) = f_2(\Sigma_2) \tag{3.29}$$

The outputs of the hidden layer y_3 and y_4 come after the step of applying activation functions f to the sum-product Σ. The result computed by the previous layer's neurons fed into the next layer as input. In this case, we are able to compute the predicted result of $\hat{Y}$:

$$\hat{Y} = f_3(y_3 W_5 + y_4 W_6 + b_2) = f_3(\Sigma_3) \tag{3.30}$$

Therefore, the error comes from prediction during feedforward training is equal to $Y - \hat{Y}$, which is ready to be adopted by the loss function and passed backward to the first input layer again by the backpropagation algorithm.

3.2.7 Back Propagation

In order to reduce the error between the predicted and the true output, for each iteration within the backpropagation procedure, we aim to find the gradient along the steepest descent direction lies on the loss function. This mechanism will update (learn) the weights and bias and such that the cost function will keep decreasing comparing with the previous iteration until converge. By correcting the parameters of the ANNs iteratively, finally, the network can achieve the desired behaviour.

The advantage of involving chain rule in backpropagation is to avoid the computing redundancy. By visiting each path only once, it allows us to obtain the partial derivative of the last node (output) respect to all its previous nodes. As an example shown in Figure 3.11, in order to calculate the gradient of output node **e** respect to the input node **a**, we start from the backwards direction. After computing the gradient of every node's output concerning its input along the path until it reaches **a**, we multiply them together. Since two paths are both existing for node **a** contributes to **e**, the gradient descent will be added for the total derivative:

$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial d} \times (\frac{\partial d}{\partial b} \times \frac{\partial b}{\partial a} + \frac{\partial d}{\partial c} \times \frac{\partial c}{\partial a}) \tag{3.31}$$

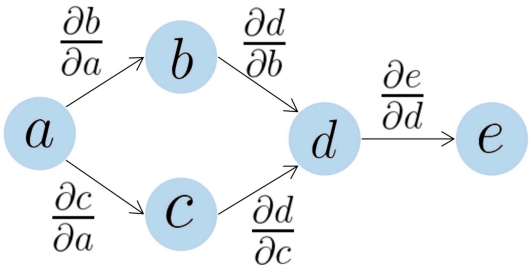


Figure 3.11: Visualize the computational graph of the chain rule.

According to the chain rule, we are now able to compute the gradient descent for the loss function L with respect to weight w and bias b, here w and b are both initialized with random values. Figure 3.12 shows an example of the ANNs by using backpropagation to compute GD.

We consider the scenario of computing GD for w_1: The path of the chain rule from ① to ⑤ in Figure 3.12 is corresponding to the following functions with forwarding computation. Note that for mathematical convivence, in order to remove the number of 2 in Equation 3.35, we can modify the cost function as follows:

$$L = \frac{1}{2}(Y - \hat{Y})^2 \tag{3.32}$$

$$\begin{cases} L = \frac{1}{2}(Y - \hat{Y})^2 & ① \\ \hat{Y} = f(\Sigma_3) & ② \\ \Sigma_3 = y_3 W_5 + y_4 W_6 & ③ \\ y_3 = f(\Sigma_1) & ④ \\ \Sigma_1 = X_1 W_1 + X_2 W_4 & ⑤ \end{cases} \tag{3.33}$$

where y represents the outcome after the previous node applying the activation function $f(\cdot)$ (sigmoid in this case) to the combination function, and Σ represents the combination function which computed by the weights connected to the current neuron.

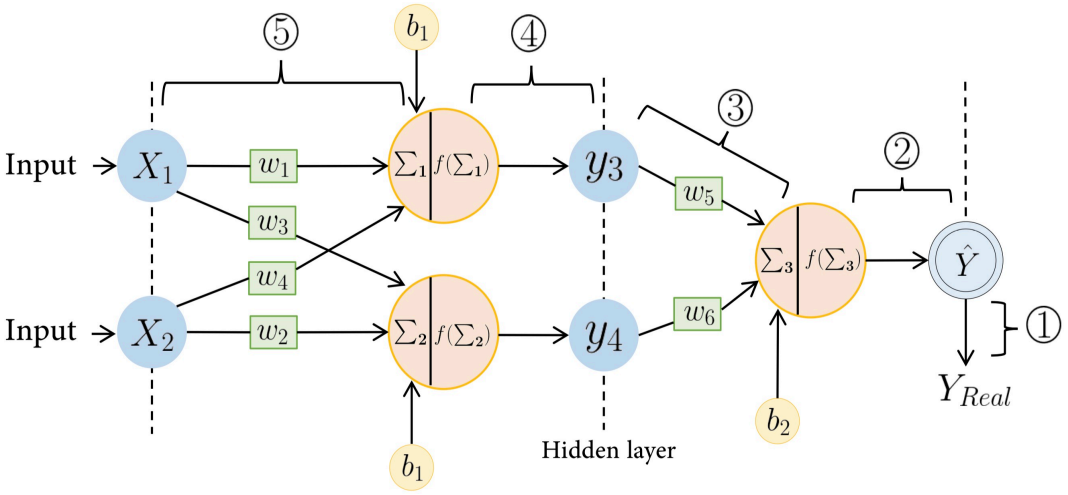


Figure 3.12: Visualize backpropagation with a three-layer ANNs. X denotes the input samples, Y is the real output, and $\hat{Y}$ is the forwarding output. The orange unit represents the neuron in the network. Blue circles denote the input and output variables. The small green square represents the weight for input, and the yellow circle is the bias.

Therefore, according to the above formulas, the partial derivative Δw_1 of L respect to w_1 can be inferred as follows. In this case, the activation function $f(x)$ is the sigmoid function. Hence, the derivative of it can be written as $f'(x) = f(x)(1 - f(x))$.

$$\Delta W_1 = \frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial \hat{Y}} \cdot \frac{\partial \hat{Y}}{\partial \Sigma_3} \cdot \frac{\partial \Sigma_3}{\partial y_3} \cdot \frac{\partial y_3}{\partial \Sigma_1} \cdot \frac{\partial \Sigma_1}{\partial W_1} \tag{3.34}$$

$$= -2(Y - \hat{Y}) \cdot f'(\Sigma_3) \cdot W_5 \cdot f'(\Sigma_1) \cdot X_1 \tag{3.35}$$

$$= (\hat{Y} - Y) \cdot f'(\Sigma_3) \cdot W_5 \cdot f'(\Sigma_1) \cdot X_1 \tag{3.36}$$

$$= (\hat{Y} - Y) \cdot f(\Sigma_3) \cdot (1 - f(\Sigma_3)) \cdot W_5 \cdot f(\Sigma_1) \cdot (1 - f(\Sigma_1)) \cdot X_1 \tag{3.37}$$

Then we can take a small step along the direction of the negative gradient to compute the update weight for w_1 according to algorithm 1:

$$* \; W_1 = W_1 - \alpha \Delta W_1 \tag{3.38}$$

The same strategy can be applied to update the bias as well, take b_2 for example:

$$\Delta b_2 = \frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial \hat{Y}} \cdot \frac{\partial \hat{Y}}{\partial \Sigma_3} \cdot \frac{\partial \Sigma_3}{\partial b_2} \tag{3.39}$$

$$= (\hat{Y} - Y) \cdot f'(\Sigma_3) \tag{3.40}$$

$$= (\hat{Y} - Y) \cdot f(\Sigma_3) \cdot (1 - f(\Sigma_3)) \tag{3.41}$$

$$\begin{cases} L = \frac{1}{2}(Y - \hat{Y})^2 & ① \\ \hat{Y} = f(\Sigma_3) & ② \\ \Sigma_3 = y_3 W_5 + y_4 W_6 + b_2 & ③ \end{cases} \tag{3.42}$$

With all the newly update parameters $*w_i$ and $*b_i$, which allows us to proceed to the next iteration of forwarding computation. By correcting the weight and bias slightly during each iteration of forward and backpropagation, the error of the predicting output will become smaller until finally converge. It means the predicted value $\hat{Y}$ is very close to the ground truth of Y.

There are several activation functions f (in subsection 3.2.2) can be chosen for the ANNs. Each of them has its advantages and disadvantages. The main challenge for the sigmoid activation function to confront is the vanishing gradient problem. The sigmoid derivative is shown in Figure 3.14, when $|x|$ increases, $f'(x)$ tends to become 0. This indicates that the gradients will constantly decrease and gradually vanish, consequently making learning become slowly. We would only gain a meaningful gradient as long as the neuron output is confined inside the area around 0. For instance, in our example above, once the result of $f'(\Sigma_3)$ or $f'(\Sigma_1)$ occasionally goes beyond "the sweet spot" and falls into the saturable regions, the value of GD Δw_1 tends to vanish. As a result, this neuron can barely update its parameters and pass the signals to previous layers. Tanh function derivative also suffers from this restriction.

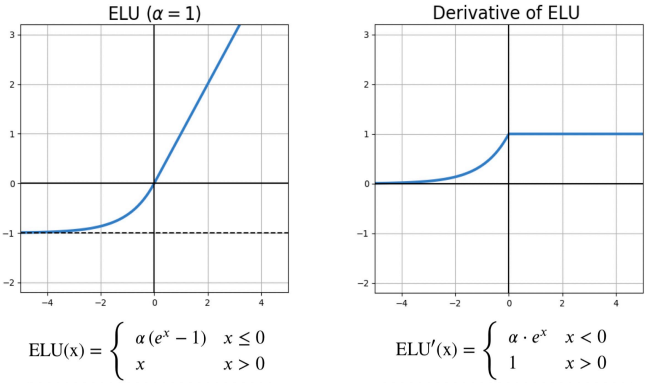


Figure 3.13: The plot of ELU activation function and the derivative of ELU

That is the reason why ELU and ReLU become the popular activation functions, which has already introduced in Equation 3.23, the derivative of it allows the neurons to learn faster, as the gradient of ReLU is equal to 1 for all the positive inputs, which can be efficiently backpropagated such that encouraging ANNs to converge faster. The drawback of ReLU is when the neuron input x equal to or smaller than 0, the GD will become 0 as well (see Figure 3.14). The left-hand side of ReLU GD which result in the dying ReLU problem and the network is not able to perform backpropagation anymore. To solve this limitation, ELU introduces an extra slope in order to compensate for the non-negative inputs, such that the DG will become positive values, and this allows the backpropagation to continue.

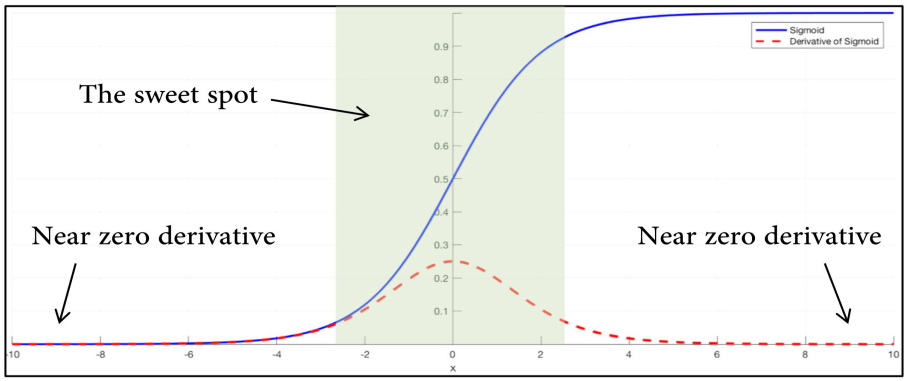


Figure 3.14: The derivative of sigmoid activation function. Reproduced image adapted from https://harrisonjansma.com/batchnorm.html.

3.3 Deep Generative Model

The main concept of this section regarding VAE is based on the work of (Kingma & Welling 2013). The generative model has achieved tremendous success for deep learning. Unlike supervised learning is to label the training dataset in order to predict the output, the generative model can be adopted for unsupervised learning for revealing the underlying hidden structure of the data. The generative models can be classified as two types of probabilistic models: prescribed and implicit probabilistic models (Mohamed & Lakshminarayanan 2016). The first model has explicit density, which is able to specify a log-likelihood function based on the observed training data. With the second model, we can determine a stochastic procedure for generating data, which can be adopted for the Markov Chain, such as problems in ecology and climate.

Since it is not possible to model the exact distribution of the training samples, no matter with prescribed or implicit probabilistic models, therefore, the goal of the generative model is to learn a distribution from the training dataset, which tends to approximate the real data distribution, and subsequently by sampling from this distribution to generate the new data points.

VAE is one of the generative models and which belongs to the prescribed probabilistic model. Since this model can generate data samples which are similar to the original inputs, thereby it can be applied to construct artificial human faces, generate music, perform denoising task for images, etc.

3.3.1 Autoencoder

The architecture of Autoencoder (AE) is shown in Figure 3.15, which is consist of a pair of (fully) connected neural networks, an encoder and a decoder. The encoder network takes in inputs of x_i and aims to map them into a dense, low-dimensional, vector representation of $h(x_i)$. The decoder can be used to convert this latent structure back to the same dimensional

outputs $\hat{x}_i$, which are similar to the original inputs x_i. Hence, AE is a data compression algorithm; it aims to learn a compressed representation of input for predicting itself, which is called dimensionality reduction. Meanwhile, the restrictions required applying to the hidden layer in order to prevent AE from learning an identity mapping in the encoder for predicting the outputs.

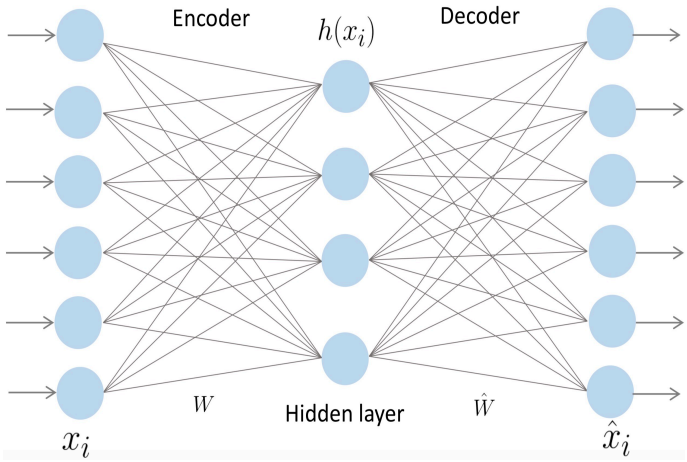


Figure 3.15: Illustration of an AE architecture. W denotes the weights of encoder network, $\hat{W}$ denotes the weights of decoder network.

Similar to the Principle Component Analysis (PCA) for conducting dimension reduction purpose, which aims at finding the eigenvectors in a low-dimensional projection in order to minimize the reconstruction errors. Intuitively, PCA targets the most critical components from the training samples by searching for the K-largest eigenvectors. AE is closely related to PCA in the form of dimensionality reduction; the objective function of both algorithms are trying to minimize reconstruction errors.

However, PCA only allows us to find the best linear reconstruction of the data (projection), which is close to the original input. AE aims to extend this ability of PCA and enable it to solve a more complex problem by the neural network. There are many types of AE, the architectures of encoder and decoder can be MLP, CNN or RNN. The simplest form of an AE is a vanilla AE with three layers, and both input and output with the same units (see Figure 3.15). According to the neuron model shown in section subsection 3.2.1, an encoder can be described by:

$$h(x_i) = f(Wx_i + b) \tag{3.43}$$

The decoder can be calculated by:

$$\hat{x}_i = g(\hat{W}h(x_i) + c) \tag{3.44}$$

where $f(\cdot)$ and $g(\cdot)$ correspond to the activation functions of encoder and decoder network, e.g. sigmoid, ReLU, b and c are the encoder and decoder bias vectors. W and $\hat{W}$ denote the encoder and decoder weight matrices respectively.

The loss function L defines the reconstruction error that measures the difference between x_i and $\hat{x}_i$, the objective of AE is to obtain a small value of L, and enable the AE model to learn an approximation such that the input is similar to the output. Note that the loss function can be either defined as the following MSE or cross-entropy.

$$L = \frac{1}{2}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{3.45}$$

where N denotes the number of input data, and the AE objective function can be inferred as:

$$\frac{1}{2}\underset{W\hat{W}}{min}\|x_i - \overbrace{g(\underbrace{f(x_i; W)}_{①}; \hat{W})}^{②}\|_2^2 \tag{3.46}$$

where the linear function of ① section of $f(\cdot)$ represents the encoder, and the ② section of $g(\cdot)$ represents the decoder network. It shows that the essential basis of AE remains the linear functions. Thereby, unlike PCA can only deal with the linear tasks, AE takes advantage of the neural network with multilayer structure such that both encoder and decoder can solve nonlinear problems.

Rather than merely duplicating of the input to the output, more importantly, AE tries to obtain the useful features from an original input by this "copying procedure". The reason why the latent representation $h(x_i)$ has a smaller number of dimensionality than the input in general, it belongs to the useful features can only be extracted through the hidden layer with this constraints. In other words, the insignificant factors of the training examples can be removed by forcing the AE to maintain the most prominent features during the data compression process in the encoder, simultaneously, the decoder is responsible for reversing the compression process and reproducing the outputs which are required as close as the inputs.

However, the fundamental potential drawback of AE, that is, its latent space contains the compressed vectors; they are usually not continuous (Roberts et al. 2018). The compressed latent space trained by AE contains gaps between the clusters of samples. It seems to make sense, as this learnt latent space supposed to be a much more solid and more uncomplicated representation than the original input itself. Thereby, AE model might achieve impressive performance on data compression or clustering, but which is lack of ability to reveal the real underlying structure of the data, since we do not want to replicate the same image that we put in the model.

Instead, a meaningful generative model with an essential requirement: we could randomly sample from a continuous latent space; therefore, it will enable the reasonable variations on the inputs. For example, if we attempt to decode a random vector inside the gap region of the latent space, the result will turn out to be unrealistic. It is because the decoder fails to find the existing encoded vectors within such "holes" in the latent space (see Figure 3.20).

The model is unable to decode a random vector properly if it never learns how to encode it.

Due to the limitation of AE, it is not a method that can be widely used in real-world applications. It usually can only be adopted to perform the data denoising task for images. The denoising AE is by adding the random noise to the input and forcing the hidden layer to learn the underlying meaningful features; eventually, the model becomes smart enough to recover an original noise-free input.

3.3.2 Variational Autoencoder

In order to combat the limitation of AE, the KL divergence needs to be introduced regarding Variational Autoencoder (VAE). Instead of simply compressing the input dataset into a dense low-dimensional space, VAE allows us to generate meaningful new samples which are similar to the original inputs. Thus, the VAE model can gain more flexibility to modify the original data according to desire. The details will be discussed in this section.

3.3.2.1 KL Divergence

The information theory was created by Claude Shannon. The goal of his work is to efficiently and reliably transmit a message from sender to recipient without losing any information. The most important metric in the information theory, which is known as Entropy (Shannon 1948). The KL divergence is a metric used to measure how a probability distribution differs from a second probability distribution by calculating the cross-entropy minus the entropy. It can be interpreted as how many bits of information we expect to lose when we approximate one probability distribution to another. For two random variables P and Q, $D_{KL}(P\|Q)$ is given by:

$$D_{KL}(P\|Q) = H(P, Q) - H(P) \tag{3.47}$$

The KL divergence can also be written in the following expectation form:

$$\begin{aligned} D_{KL}(P\|Q) &= E_P[-\log\ Q(X)] - E_P[-\log\ P(X)] \\ &= E_P[\log\ P(X) - \log\ Q(X)] \end{aligned} \tag{3.48}$$

where it can be expressed either by the discrete summation form or the continuous integral form:

$$D_{KL}(P\|Q) = \sum_i P(i)\ \log\ \frac{P(i)}{Q(i)} \tag{3.49}$$

$$D_{KL}(P\|Q) = \int P(x)\ \log\ \frac{P(x)}{Q(x)} dx \tag{3.50}$$

The KL divergence is non-negative: $D_{KL}(P\|Q) \geq 0$. It is indicating that these two distributions are identical $P = Q$ when the KL divergence is equal to 0. Furthermore, the KL divergence is not symmetric $D_{KL}(P\|Q) \neq D_{KL}(Q\|P)$. The reason due to the fact that the cross-entropy of $H(P, Q)$ and $H(Q, P)$ themselves are asymmetric, since they use different

probability distributions P and Q for computing the expectations, respectively. Intuitively, as an example shown in Figure 3.16. Suppose we are given two probability distributions: Q is a unimodal and P is multimodal. The $D_{KL}(P\|Q)$ in (a) denotes that we aim to use distribution P to compute the expectation. The distribution of unimodal Q will try to cover the multimodal distribution P. However, as the $D_{KL}(Q\|P)$ shown in (b) and (c), we use the distribution of unimodal Q to compute the expectation. Thus, it turns out that unimodal Q tries to fit into the multimodal P.

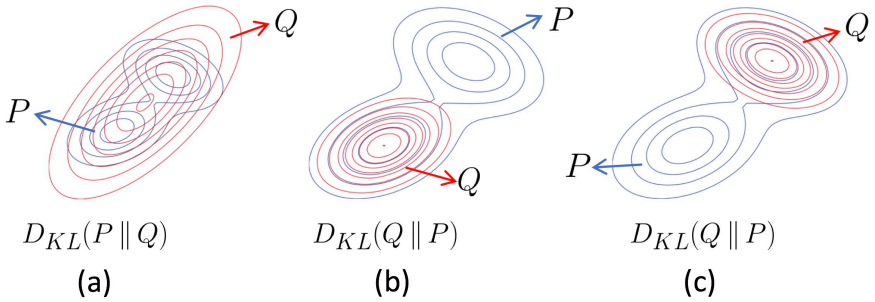


Figure 3.16: The comparison of $D_{KL}(P \| Q)$ and $D_{KL}(Q \| P)$. Reproduced images adapted from: https://ermongroup.github.io/cs228-notes/inference/variational/.

3.3.2.2 Approximate Inference

VAE is a generative model shown in Figure 3.17, which consists of an encoder $q_\phi(z|x)$ (posterior) and a decoder $p_\theta(x|z)$ (likelihood), which are defined by a pair of (fully connected) ANNs. Similar to AE, one of the primary purposes of VAE is to extract the most critical features from a high-dimensional input by minimizing the reconstruction errors. The reason why the latent space Z has lower dimensions than input is to prevent the encoder $q_\phi(z|x)$ from encoding a completely identical mapping, which means the latent space tends to avoid keeping all the trivial information from resource data.

Suppose we are given a high-dimensional observed dataset X in Figure 3.17. By training, the encoder network will learn several latent variables (features), the number of which depends on the dimensionality of the latent space Z, these features refer to the descriptive attributes of the data samples. We provide an example of training the MNIST datasets of handwritten digits by VAE. In our case, if we only would like to encode this digit image to be a three-dimension representation, then the VAE model will result in three essential features such as edge, stroke and skeleton. The latent variable vectors z_i formed by the mean μ_i and the variance σ_i of each feature accordingly. The new samples in the output $\hat{X}$ will be generated based on Z.

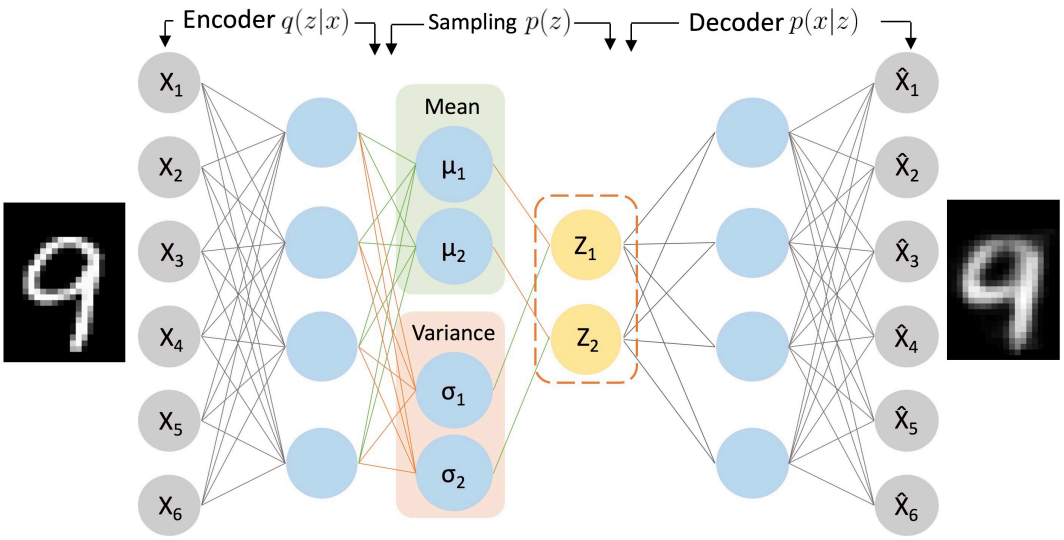


Figure 3.17: Illustration of VAE model architecture. The grey circles denote the observed variables of input and output. The hidden layers of DNNs are represented by the blue circles, and the yellow ones denote the latent variables. The output layer of an encoder, which is called latent space representation of Z. The same way of expression is also adopted in the following.

There are two basic rules for generating new samples in the latent space. First, sampling the latent variable vector z_i which obtained from the posterior $q_\phi(z|x)$. This sampling is forced to approximate to the prior distribution $p(z) = \mathcal{N}(z|0, I)$; secondly, after sampling, the decoder $p_\theta(x|z)$ plays the role of evaluating the reconstruction accuracy. The output $\hat{x}_i$ computed by the conditional likelihood function $p_\theta(x|z = z^{(i)})$with the sampled latent vector z_i. This will be discussed in the following.

In comparison with the generative model of AE, the process of constructing the latent space Z in VAE is a stochastic sampling process, which differs from simply learning a mapping to a deterministic variable (fixed vector) by AE (Amine 2018). In addition, unlike the non-continuous latent space of AE, another important property of VAE is to force the latent space Z to be continuous. More specifically, by using KL divergence to restrict the distribution of latent variables to become a normal distribution (Gaussian), such that we can randomly pick a vector in it and decode a meaningful result. It implies that we are able to sample the latent variable from a normal distribution, in order to reconstruct the new realistic samples.

Since we are interested in the inference problem, and that means: compute the conditional distribution (posterior) of the hidden variables given some observed data, that is, $p(z|x)$:

$$p(z|x) = \frac{p(z, x)}{p(x)} \tag{3.51}$$

Due to the intractability of the marginal distribution of $p(x)$ (see Equation 3.51), the true posterior density is intractable as well.

$$p(x) = \int p(z, x)\, dz \tag{3.52}$$

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} \tag{3.53}$$

Since this inference problem is not solvable directly, the idea of variational inference is to search for an additional distribution $q(z|x)$, a so-called variational approximation and make it approximate to the target distribution $p(z|x)$. Thereby, we could use this auxiliary distribution of $q(z|x)$ instead of the true distribution $p(z|x)$. This approach can be achieved by minimizing the KL divergence in order to make these two distributions as close as possible. Eventually, this inference problem is cast to be an optimization problem with KL divergence:

$$q^*(z|x) = argmax_{q(z|x)\in\mathscr{L}}\, KL(q(z|x)\|p(z|x)) \tag{3.54}$$

where $q(z|x) \in \mathscr{L}$ denotes each of the $q(z|x)$ is one of the candidate distributions, and it tends to approximate to the true posterior. In order to accomplish this optimization problem, we need to specify an approximating family $\mathscr{L}$, which consists of such candidate distributions. In short, when we are given an intractable distribution $p(z|x)$, the variational inference aims at solving the casted optimization problem over a tractable probability distribution of $\mathscr{L}$.

In other words, the goal of targeting posterior is becoming to find the best approximation $q^*(z|x)$ among $\mathscr{L}$; we choose the most similar one to the true distribution $p(z|x)$ (see Equation 3.54). In doing so, we can find the approximating solution by inquiring whether the specify $q(z|x)$ could best fit $p(z|x)$ with measuring KL divergence of $D_{KL}(q(z|x) \,\|\, p(z|x))$ (see subsubsection 3.3.2.1).

In order to demonstrate the derivation of variational inference, we can write the objective as KL divergence initially:

$$\begin{aligned} D_{KL}(q(z|x) \,\|\, p(z|x)) &= \sum_z q(z|x)\, \log\, \frac{q(z|x)}{p(z|x)} \\ &= E\left[\log\, \frac{q(z|x)}{p(z|x)}\right] \\ &= E\left[\log\, q(z|x) - \log\, p(z|x)\right] \\ &= E\left[\log\, q(z|x) - \log\, \frac{p(x|z)p(z)}{p(x)}\right] \\ &= E\left[\log q(z|x) - \log\, p(x|z) - log\, p(z) + \log\, p(x)\right] \end{aligned} \tag{3.55}$$

where $\log p(x)$ is still existing, this objective can hardly be solved directly. In order to make the KL divergence problem become solvable, the Evidence Lower Bound (ELBO) needs to be introduced. It is defined as the following:

$$\begin{aligned} ELBO(q) &= E\left[\log p(x|z)\right] + E\left[\log p(z)\right] - E\left[\log q(z|x)\right] \\ &= E\left[\log p(z, x)\right] - E\left[\log q(z|x)\right] \end{aligned} \tag{3.56}$$

where the ELBO can be obtained by removing the intractable term $\log p(x)$ in Equation 3.55. In the following, we omit the expectation symbol E for convenience. Therefore, we can rewrite the objective as:

$$J(q) = D_{KL}(q(z|x)\|p(z|x)) = \log\ p(x) - ELBO(q) \tag{3.57}$$

Due to the non-negative property of the KL divergence, we have the following equation:

$$\log\ p(x) = D_{KL}(q(z|x)\ \|\ p(z|x)) + ELBO(q) \geq ELBO(q) \tag{3.58}$$

The definition of ELBO indicates that it is the lower bound of $\log p(x)$ (log evidence).

We can rearrange the equation as:

$$J(q) = ELBO(q) = \log\ p(x) - D_{KL}(q(z|x)\ \|\ p(z|x)) \tag{3.59}$$

This VAE objective can be interpreted as meaning that we try to model the data X, which is defined by the distribution of $p(x)$ with the error term of $D_{KL}(q(z|x)\ \|\ p(z|x))$. Hence, VAE aims to search for the lower bound of $p(x)$. It is capable of solving the untrackable evidence, due to the fact that VAE avoids targeting the exactly real distribution directly. This approach cleverly converts the original problem to be an optimization one and search for a bound instead, which tends to be trackable.

Recall that the VAE objective is to minimize the KL divergence since it switches to be the reduction term on the right-hand side in Equation 3.59. Then the objective is equivalent to maximize the ELBO, which can be interpreted as minimizing the negative objective of $-J(q)$.

We derive the ELBO based on Equation 3.56. The objective is not including the intractable evidence $p(x)$ so far.

$$\begin{aligned} ELBO(q) &= \log\ p(x|z) - (\log\ q(z|x) - \log\ p(z)) \\ &= \log\ p(x|z) - D_{KL}(q(z|x)\|p(z)) \end{aligned} \tag{3.60}$$

Rewrite objective (cost function) with parameters as following:

$$\begin{aligned} \mathcal{L}_{VAE}(\phi, \theta, x, z) &= -ELBO \\ &= E_{z\sim q_\phi(z|x)}\left[\log\ p_\theta(x|z)\right] - D_{KL}(q_\phi(z|x)\ \|\ p_\theta(z)) \end{aligned} \tag{3.61}$$

It turns out, the goal of VAE is to minimize the cost function $\mathcal{L}(\phi, \theta, x, z)$. Therefore, we want to keep the value of the KL divergence term towards 0 as much as possible (as small as possible). Meanwhile, increasing the likelihood term $E_{z\sim q_\phi(z|x)}\left[log\ p_\theta(x|z)\right]$ as big as possible (minimize the error).

Explain the objective function from a mathematical perspective: the KL divergence term including a prior, we assume that it follows a standard Gaussian distribution $p_\theta(z) = \mathcal{N}(0, I)$. In addition, the variational approximate posterior $q_\phi(z|x) = \mathcal{N}(z; \mu_z, \sigma_z^2 I)$. The variational approximate posterior is a multivariate Gaussian with a diagonal covariance structure. The

reason belongs to the simplifying assumption about dimensionality independent that we make, and hence the covariance matrix only has nonzero values on the diagonal. There is existing no correlation between each dimension, such that the output information appears to be two vectors μ_z and σ_z^2. They are used to describe the mean and standard deviation of the latent space distribution, which is a Gaussian output of the encoding MLP. It is obtained from adding an extra fully connected linear layer between the output layer of the MLP and z. Both μ and σ have the same dimensions as z. That is, in latent space z, each dimension d of the vectors corresponds to its normal distribution $\mathcal{N}(\mu_d, \sigma_d^2)$, such that from which z_d can be sampled. The computing result of KL divergence in Gaussian case (see Appendix A for a detailed computation):

$$-D_{KL}(q_\phi(z|x) \,\|\, p_\theta(z)) = \frac{1}{2}\sum_{j=1}^{J}(1 + log\,((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2) \tag{3.62}$$

Where j denotes the dimensionality of the z. μ_j and σ_j represent the j-th element of these vectors. We can also rewrite this term as following. Note that the $\Sigma(X)$ is just a vector since it is a diagonal matrix as we mentioned above:

$$D_{KL}\underbrace{[\mathcal{N}(\mu(X), \Sigma(X))}_{Q(z|X)} \,\|\, \underbrace{\mathcal{N}(0,1)]}_{P(z)} = \frac{1}{2}\sum_{j}(\Sigma(X) + \mu^2(X) - 1 - log\,\Sigma(X)) \tag{3.63}$$

For the decoding network $E_{z\sim q_\phi(z|x)}\left[log\, p_\theta(x|z)\right]$, it aims to reconstruct the original input X. Depending on the data type, and the network can be either modelled by MLP or Bernoulli. It describes the log-likelihood of the observed x given the z that we have sampled. This term can be maximized when $p_\theta(x|z)$ assigns a high probability to the input X. We could simply adopt methods like Markov Chain Monte Carlo (MCMC) for sampling the new data. Unfortunately, these sampling-based methods have several shortcomings:

$$E_{z\sim q_\phi(z|x)}\left[log\, p_\theta(x|z)\right] \approx \frac{1}{L}\sum_{l=1}^{L} log\, p_\theta(x|z^{(l)}), z^{(l)} \sim q_\phi(z|x) \tag{3.64}$$

where L denotes the dimensionality of z, it requires to compute the iterative inference schemes for each data point, which is computationally expensive. On the other hand, MCMC converges extremely slowly, since it can provide guarantees of asymptotically producing exact samples from the target density (Roberts et al. 2004). However, for VAE, we would prefer to find an alternative optimization approach tends to converge faster, even though it only allows to search for a density which is approximate to the target. We will discuss the details in the next section.

Consequently, now we can write our objective as:

$$\mathcal{L}_{VAE}(\phi, \theta, x, z) \approx \frac{1}{2}\sum_{j=1}^{J}(1 + log\,((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2) + \frac{1}{L}\sum_{l=1}^{L} log\, p_\theta(x|z^{(l)}), z^{(l)} \sim q_\phi(z|x) \tag{3.65}$$

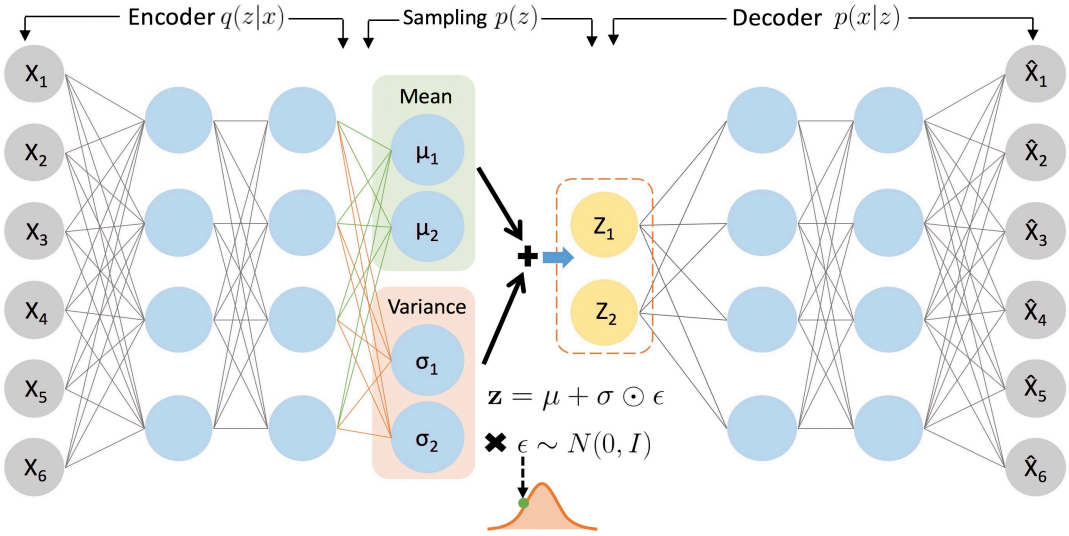


Figure 3.18: The architecture of the VAE model with the reparameterization trick. For the encoding, we first sample from a standard normal distribution $\mathcal{N}(0, I)$, furthermore, by using this sampling noise term ϵ together with μ and σ which are trained by the encoder network in order to encode latent space z afterwards.

3.3.2.3 The Reparameterization Trick for Backpropagation

In order to optimize our objective, this can be achieved by a good estimation of the gradient. Luckily, the KL divergence result can be differentiated. However, the problem scenario comes from the second term $E_{z\sim q_\phi(z|x)}\left[log\ p_\theta(x|z)\right]$. This term cannot be differentiated directly for the backpropagation, due to the fact that the process of sampling from $q_\phi(z|x)$ which is non-deterministic and thus the gradient is impossible to backpropagate through the VAE model. Additionally, we already discussed the difficulty by using MCMC sampling-based algorithms.

To combat this limitation, the key behind VAE is the reparameterization trick. As it is shown in Figure 3.18. Instead of sampling z from the $\mathcal{N}(\mu, \sigma^2)$ directly, where μ, σ^2 are the vectors trained by encoder network, we sample a noise (auxiliary) variable ϵ from a standard normal distribution $\epsilon \sim \mathcal{N}(0, 1)$. Since the standard normal distribution is an instance of the normal distribution with $\mu = 0$ and $\sigma^2 = 1$, every normal distribution $\mathcal{N}(\mu, \sigma^2)$ can be written as:

$$\mathcal{N}(\mu, \sigma^2) \sim \mu + \sigma^2 \cdot \mathcal{N}(0, I) \tag{3.66}$$

During the forward processing, initially, we apply this property to draw a sample ϵ from the standard normal distribution and transform it into the latent sample z according to the Equation 3.67 for each dimension. In which, symbol $\odot$ denotes an element-wise product.

$$z = \mu + \sigma \odot \epsilon \quad \epsilon \sim \mathcal{N}(0, I) \tag{3.67}$$

The essential technique of the reparameterization trick is during the backpropagation of training, it allows us to obtain a differentiable likelihood expectation $E_{z\sim q_\phi(z|x)}\left[\log\ p_\theta(x|z)\right]$ of

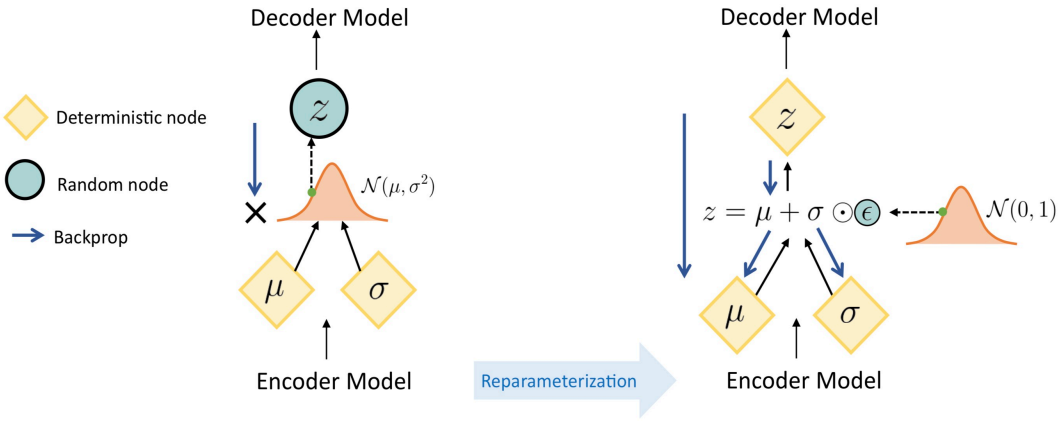


Figure 3.19: The VAE without using the reparameterization trick (left), it does not allow backpropagation. The model enables backpropagation is using the reparameterization trick (right).

the variational lower bound. As it is shown in Figure 3.19, the encoder network without reparameterization trick can only sample from the deterministic node (variables) with μ and σ for encoding z. The only difference is that the encoder with reparameterization trick samples from a standard normal distribution first, and by using this sampling noise term ϵ together with μ and σ for encoding z afterwards.

In doing so, the differentiable transformations of the estimator can be accomplished, since this trick tends to avoid the parameter ϕ of $q_\phi(\cdot)$ (see Equation 3.70). It wittily introduces a differentiable transformation $g_\phi(\cdot)$ with the noise variable ϵ. This property of reparameterization trick can be proven as follows:

Given a deterministic mapping $z = g_\phi(\epsilon, x)$, and $q_\phi(z|x) \prod_i dz_i = p(\epsilon) \prod_i d\epsilon_i$ follows. So, we can derive the expectation:

$$\begin{aligned} E_{q_\phi(z|x)}[f(z)] &= \int q_\phi(z|x) f(z) dz = \int p(\epsilon) f(z) d\epsilon \\ &= \int p(\epsilon) f(g_\phi(\epsilon, x)) d\epsilon = E_{p(\epsilon)}[f(g_\phi(\epsilon, x))] \end{aligned} \tag{3.68}$$

where it is possible to represent z as a deterministic variable:

$$z = g_\phi(\epsilon, x), \quad \epsilon \sim \mathcal{N}(0, 1) \tag{3.69}$$

More specifically, for any random variable z, since the $E_{p(\epsilon)}[f(g_\phi(\epsilon, x))]$ is differentiable, the expectation value of $f(z)$:

$$\begin{aligned} E_{q_\phi(z|x)}[f(z)] &= E_{\epsilon \sim \mathcal{N}(0,1)}[f(g_\phi(\epsilon, x))] \\ &\approx \frac{1}{L} \sum_{l=1}^{L} f(g_\phi(\epsilon, x)) \quad \epsilon^{(l)} \sim p(\epsilon) \end{aligned} \tag{3.70}$$

Therefore, the sampling phase only depends on $\epsilon^{(l)} \sim \mathcal{N}(0,1)$ rather than the posterior parameterized with ϕ, $z^{(l)} \sim q_\phi(z|x)$, which results in this expectation becomes differentiable as mentioned above. We can rewrite the decoder in the objective as:

$$E_{q_\phi(z|x)}\left[\log\ p_\theta(x|z)\right] \approx \frac{1}{L}\sum_{l=1}^{L} \log\ p_\theta(x|z^{(l)}), \quad z^{(l)} = g_\phi(x, \epsilon^{(l)}), \quad \epsilon^{(l)} \sim \mathcal{N}(0,1) \tag{3.71}$$

The complete process of updating the parameters of variational lower bound by Stochastic Gradient Variational Bayesian (SGVB) approach is given in algorithm 2.

Algorithm 2: SGVB algorithm

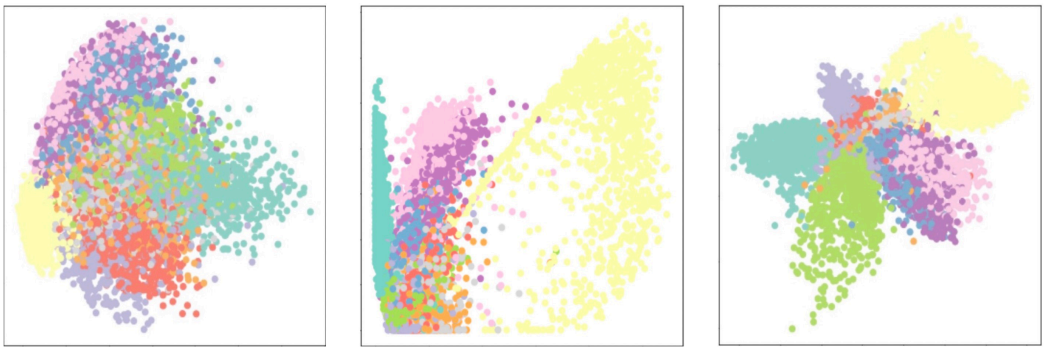
$\theta, \phi \leftarrow$ Initialize parameters
while until convergence **do**
  $X^M \leftarrow$ draw M data points
  $\epsilon \leftarrow$ sample from $p(\epsilon)$
  $g \leftarrow$ compute gradient $\nabla_{\theta,\phi}\hat{\mathcal{L}}^M(\theta, \phi; X^M, \epsilon)$
  $\theta, \phi \leftarrow$ update using SGD
end while

Given a full dataset X with N data points, where the minibatch X^M is the sample randomly drawn from X with M data points. We can now apply algorithm 2 to the objective function, to construct a model estimator with the marginal likelihood lower bound of the full dataset.

$$\begin{aligned} &\mathcal{L}(\theta, \phi; X) \approx \hat{\mathcal{L}}^M(\theta, \phi; X^M) \\ &= \frac{N}{M}\sum_{i=1}^{M}\{\frac{1}{2}\sum_{j=1}^{J}(1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2) + \frac{1}{L}\sum_{l=1}^{L}\log\ p_\theta(x|z^{(l)})\} \\ &\quad \text{where} \quad z^{(l)} = g_\phi(x, \epsilon^{(l)}) = \mu + \sigma \odot \epsilon^{(l)} \qquad \epsilon^{(l)} \sim \mathcal{N}(0,1) \end{aligned} \tag{3.72}$$

In summary, the VAE has a similar structure to the nonlinear AE model, except for VAE makes a change to force the latent values are sampled from a normal distribution. The prior $p(z)$ encourages the latent space to shape under this assumption. This can be achieved by using the KL divergence in order to make sure the latent distribution $q_\phi(z|x)$ is approaching to the prior $p_\theta(z)$.

We provide an example of training the MNIST datasets by these three models. Comparing the results of these three models are shown in Figure 3.20. The linear dimension reduction model of PCA has the worst performance. The nonlinear AE model generates mutually independent clusters. The latent space consists of large holes and gaps between clusters. An AE might be suitable for filtering the noise. However, it is challenging to generate new samples from the sparse vector space. If the decoder intends to sample a variation from there, it will occur the unrealistic or invalid outputs, because they are the unidentifiable regions which are left non-encoded by the encoder. On the contrary, VAE enables us to take advantage of the continuous latent space based on prior knowledge that we offer. It could generate meaningful variations based on the input data when we pick a random vector in latent space for decoding. Meanwhile, the most valuable information about input remains unaffected.



(a) Latent distribution by label for PCA (b) Latent distribution by label for AE (c) Latent distribution by label for VAE

Figure 3.20: Visualize the (two-dimensional) latent space of PCA(a), AE(b) and VAE(b). We train the MNIST datasets by these three models. Each colour represents the distribution with one digital number by labelling in latent space. The code is available via https://github.com/YueXu8231/VAE

3.4 Deep Information Bottleneck

The IB principle was first introduced in 1999 (Tishby et al. 2000). It is extended from the rate distortion theory and holding a similar structure to the deep learning. Rather than by using three variables to describe the relevant information throughout the IB model, deep IB treats X, Y, T as the layers in DNNs. The information compression phase of the IB principle is according to an encoder in DNNs, and the constraints for the compression can be seen as a decoder. According to the original IB principle we discussed in subsection 3.1.4, deep IB also can be cast into the following optimization problem:

$$p(\mathbf{t}|\mathbf{x}) = \underset{p(t|x),p(y|t),p(t)}{\arg\min} \{I(X;T) - \beta I(T;Y)\} \tag{3.73}$$

where $I(X;T)$ measures how much the information in the input X has been compressed into a more compact representation T. $I(T;Y)$ denotes the quality of reconstructing result, which means how close our predicting output from the desired output (ground truth label). The low-dimensional latent space T that we aim to obtain turns out to be a multivariate Gaussian distribution (see Figure 3.21).

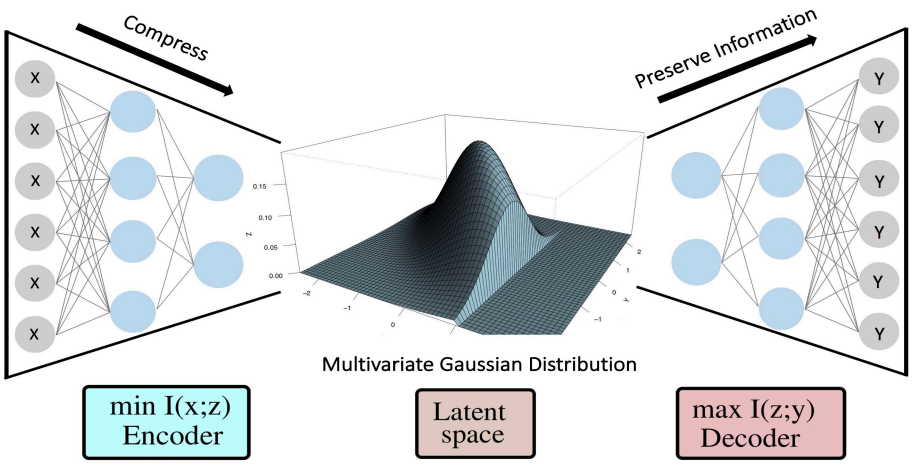


Figure 3.21: Visualize the deep IB model. In supervised learning scenarios, deep IB determines the optimal representation by compressing the information in the input X, meanwhile, preserving the information related to the desired output Y. Reproduced image adapted from `http://ml4ms2019.aalto.fi/wp-content/uploads/2019/05/VolkerRoth_2019.pdf`

The purpose of IB interpretation of DNNs (see Equation 3.73) is still to search for a balance between information-compression and the reconstruction accuracy, which is under the same structure as the original IB principle: by squeezing the information through the bottleneck, hence only the most useful factors regarding the real output Y remain in the data X. The tradeoff parameter β regarding the level of compression, the larger β tends to preserve more information on the outcome Y, which leads to the lower compression on X. The essential technique behind the IB interpretation based on DNNs is substantially a procedure of removing the irrelevant information, which is called "learning by forgetting".

3.5 Deep Variational Information Bottleneck

The original IB tries to explain the DNNs in an information-theoretic objective point of view. However, this method encounters a challenge in computation. The solution is infeasible to apply to the DNNs (Chechik et al. 2005). It is not providing the experiment result because of its impracticable property when applied to the DNNs. Besides merely analyzing the DNNs with IB method. Alemi et al. (2016) proposes a variational approximation to IB method. Instead of targeting the IB theoretical bound, this approach is to construct a lower bound on the IB objective by using variational inference, which is known as Variational Information Bottleneck (VIB). Since the VIB provides promising results for solving the supervised learning problem in general, it describes a variation principle for extracting a compressed representation that only captures the relevant information between the input and output signals (messages). This method is also can be adopted in unsupervised learning. In this paper, we focus on applying VIB to solve the cause-effects problem; we will discuss it in the next chapter.

Here we use Z to denote the compact representation of the bottleneck layer, or called latent space in DNNs. Recall that the IB objective we discussed above (see Equation 3.73). Equivalently, it can also be formulated as follows:

$$p(\mathbf{z}|\mathbf{x}) = \underset{p(\mathbf{z}|\mathbf{x}),p(\mathbf{y}|\mathbf{z}),p(\mathbf{z})}{\arg\min} \; \mathbf{I}(X;Z) - \beta\, \mathbf{I}(Z;Y) \tag{3.74}$$

3.5.1 The Variational Bound Examination

In order to free the computational burden, Alemi et al. (2016) apply the variational approach to the deep IB objective in Equation 3.74. We will examine the variational bound on the terms of $I(Z;Y)$ and $I(X;Z)$, separately.

First, we aim to obtain the variational bound for $I(Z;Y)$, according to the mutual information definition with continuous random variables (see Equation 3.8):

$$\begin{aligned} I(Z;Y) &= \int p(y,z) \log \frac{p(y,z)}{p(y)p(z)} dy\, dz \\ &= \int p(y,z) \log \frac{p(z)p(y|z)}{p(y)p(z)} dy\, dz \\ &= \int p(y,z) \log \frac{p(y|z)}{p(y)} dy\, dz \end{aligned} \tag{3.75}$$

where the decoder of deep VIB is defined by $p(y|z)$ which depend on the encoder, according to the Markov chain:

$$\begin{aligned} p(y|z) &= \int p(x,y|z) dx = p(y|x)p(x|z) dx \\ &= \int p(y|x) \frac{p(z|x)p(x)}{p(z)} dx \end{aligned} \tag{3.76}$$

where the probability distribution of the decoder $p(y|z)$ is unsolvable directly due to the intractability of $p(z)$. Thereby, we introduce an additional distribution $q(y|z)$, which is a variational approximation to the true decoder $p(y|z)$. The KL divergence of these two distributions can be defined as follows, where the value of it is required to be bigger or equal to 0 due to the non-negative property of the KL divergence:

$$\begin{aligned} D_{KL}(p(y|z)\|q(y|z)) &= \int p(y|z) \log \frac{p(y|z)}{q(y|z)} dy\, dz \geq 0 \\ \Rightarrow \int p(y|z) \log p(y|z)\, dy\, dz &\geq \int p(y|z) \log q(y|z)\, dy\, dz \end{aligned} \tag{3.77}$$

This inequality allows us to create a lower bound for the $I(Z;Y)$:

$$I(Z;Y) = \int p(y,z) \log \frac{p(y|z)}{p(y)} dy dz \geq \int p(y,z) \log \frac{q(y|z)}{p(y)} dy\, dz \tag{3.78}$$

Then it leads to:

$$\begin{aligned} I(Z;Y) &\geq \int p(y,z)\log q(y|z)dy\,dz - \int p(y)\log p(y)dy \\ &= \int p(y,z)\log q(y|z)dy\,dz + H(Y) \end{aligned} \tag{3.79}$$

where $H(Y)$ denotes the entropy of the ground-truth label, hence it can be ignored during the optimization process.

According to the Markov chain, we rewrite $p(y,z)$ as:

$$p(y,z) = \int p(x)p(y|x)p(z|x)dx \tag{3.80}$$

Therefore,

$$I(Z;Y) \geq \int p(x)p(y|x)p(z|x)\log q(y|z)dx\,dy\,dz \tag{3.81}$$

Since $p(x)p(y|x) = p(x,y)$:

$$I(Z;Y) \geq \int p(x,y)p(z|x)\log q(y|z)dx\,dy\,dz \tag{3.82}$$

We consider using empirical data distribution (see Equation 3.83) to approximate the joint distribution $p(x,y)$ as:

$$p(x,y) = \frac{1}{N}\sum_{n=1}^{N} g_{x_n}(x)\, g_{y_n}(y) \tag{3.83}$$

Thereby, the lower bound for the term of $I(Z;Y)$ becomes:

$$I(Z;Y) \gtrapprox \frac{1}{N}\sum_{n=1}^{N}\int p\left(z|x_n\right)\log q\left(y_n|z\right)dz \tag{3.84}$$

Secondly, we examine the term of $\beta I(Z;X)$ in the deep VIB objective, following the definition of mutual information we have:

$$\begin{aligned} I(Z;X) &= \int p(x,z)\log\frac{p(z|x)}{p(z)}dx\,dz \\ &= \int p(x,z)\log p(z|x)dxdz - \int p(z)\log p(z)dz \end{aligned} \tag{3.85}$$

Due to the intractability of marginal distribution $p(z)$, we give a prior knowledge to this latent space; the encoder adopts the same strategy from VAE (see subsubsection 3.3.2.2). Similarly, we use a variational marginal $q(z) = \mathcal{N}(0,I)$ to approximate this intractable marginal distribution $p(z)$, combining with the KL divergence non-negative property, we can infer the following inequality:

$$D_{KL}(p(z), q(z)) \geq 0 \Rightarrow \int p(z) \log \frac{p(z)}{q(z)} dz \geq 0 \tag{3.86}$$

$$\int p(z) \log p(z) dz \geq \int p(z) \log q(z) dz \tag{3.87}$$

By applying Equation 3.87 to the result of $\beta I(Z;X)$ (see Equation 3.85), which enables a variational bound for this term:

$$\begin{aligned} I(Z;X) &\leq \int p(x,z) \log p(z|x) dx \, dz - \int p(z) \log q(z) dz \\ &\leq \int p(x)p(z|x) \log p(z|x) dx \, dz - \int p(z) \log q(z) dz \\ &= \int p(x)p(z|x) \log \frac{p(z|x)}{q(z)} dx dz \end{aligned} \tag{3.88}$$

With the empirical data distribution, it can be rewritten as:

$$I(Z;X) \lessapprox \frac{1}{N} \sum_{n=1}^{N} p(z|x_n) \log \frac{p(z|x_n)}{q(z)} \tag{3.89}$$

Eventually, we combine these two variational bounds together for constructing a lower bound L of the deep VIB objective:

$$L = I(X;Z) - \beta I(Z;Y) \approx \frac{1}{N} \sum_{n=1}^{N} \left[\int p(z|x_n) \log \frac{p(z|x_n)}{q(z)} \, dz - \beta \int p\left(z|x_n\right) \log q\left(y_n|z\right) dz \right] \tag{3.90}$$

According to the definition of KL divergence subsubsection 3.3.2.1, we can rewrite the first term of the objective as:

$$\int p\left(z|x_n\right) \log \frac{p\left(z|x_n\right)}{q(z)} \, dz = \; D_{KL}\left[p\left(Z|x_n\right), q(Z)\right] \quad q(Z) \sim \mathcal{N}(0, I) \tag{3.91}$$

Rearrange these two terms, and we obtain the deep VIB objective:

$$L_{IB} = D_{KL}\left[p\left(Z|x_n\right), q(Z)\right] - \beta \; E_{z \sim p(z|x)}\left[\log q\left(y_n|z\right)\right] \quad q(Z) \sim \mathcal{N}(0, I) \tag{3.92}$$

where we successfully cast the VIB objective to be an optimization problem until now. Recall that throughout this lower bound, which allows us to minimize this objective.

3.5.2 The Reparameterization Trick for deep VIB

Assume that the deep network of an encoder which is defined as $f(\cdot)$, then $p(Z|x) = \mathcal{N}\left(z|f_\mu(x), f_\Sigma(x)\right)$. The compact representation Z is therefore sampled from the normal distribution $z \sim p(z|x) = \mathcal{N}\left(\mu, \sigma^2\right)$. However, in practice, we encounter difficulties through backpropagation due to the non-deterministic property of Z, as the reparameterization trick we discussed about VAE (see subsubsection 3.3.2.3). The same technique is

applied to VIB in order to obtain an unbiased gradient estimate of the real expected gradient by Monte Carlo sampling (Blundell et al. 2015). This allows the stochastic gradient descent to apply for optimizing the lower bound of deep VIB. Since the reparameterization trick by introducing a stochastic input $\epsilon \sim N(0, I)$, furthermore, we can sample from a deterministic mapping $z = f(x, \epsilon)$. This enables the gradient to backpropagate in the network.

Based on the reparameterization trick, the deep VIB objective can be either expressed as:

$$L_{IB} = D_{KL}\left[p\left(Z|x_n\right), q(Z)\right] - \beta E_{\epsilon\sim p(\epsilon)}\left[\log q\left(y_n|f\left(x_n, \epsilon\right)\right)\right] \tag{3.93}$$

where $z = f(x, \epsilon)$, with $\epsilon \sim \mathcal{N}(0, I)$, $f(\cdot)$ denotes encoder network, which is deterministic with x and the noisy term of ϵ.

In summary, we could explain the VIB objective in machine learning point of view as well. Comparing with the commonly used optimization algorithm framework (e.g. SGD) to minimize the objective by GD. The objective function of VIB also applies the same methodology:

$$\text{Cost Function} = \underbrace{\text{Regularizer}}_{Encoder} + \underbrace{\text{Loss Function}}_{Decoder} \tag{3.94}$$

The second term of $q(y|z)$ in the VIB objective (see Equation 3.93) describes the reconstruction error, which is corresponding to the loss function that we would like to minimize. The KL term describes that we aim to approximate the encoder's distribution $p(Z|x)$ towards $p(Z)$, with $p(Z) = \mathcal{N}(0, 1)$ in order to encourage the latent space toward a "healthy and simple" assumption, which is corresponding to the regularizer. It plays the role of a penalty term for punishing the model becomes too complex since we force Z follows a standard normal distribution. The model will receive a penalty once Z is violating this complexity constraint. (see Figure 3.22).

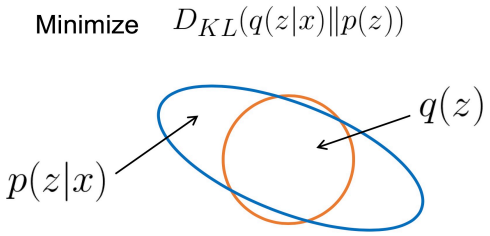


Figure 3.22: Visualize a two-dimensional latent space of Z. KL term enforces posterior $q(z|x)$ towards the prior $q(z)$.

As the objective function in general, there is a tradeoff between the loss function and the regularization term. It appears as a training balance between the reconstruction accuracy and the KL penalty (see Figure 3.23). We expect Z only keeps the most valuable information in the latent space, therefore, VIB (or deep IB) tends to sacrifice part of the accuracy during

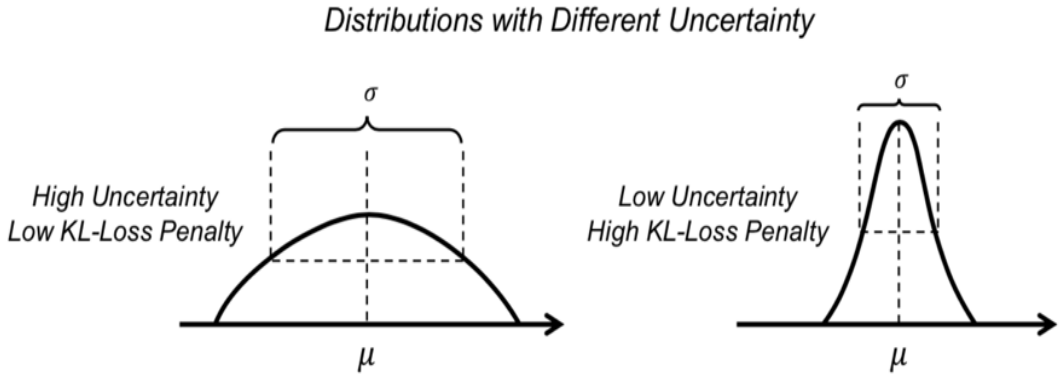


Figure 3.23: Visualize distributions with different uncertainty, image adopted from (Amine 2018). σ is a measurement of the decoder uncertainty. Higher uncertainty denotes the lower KL penalty term performed on the model. Without uncertainty, the decoder would gain complete accuracy by the decoder, and therefore, VIB might lose its compression ability in the encoder at all.

predicting $q(y|z)$, by reducing the model complexity to improve the generalization ability, it can be understood as the role of regularization for preventing model overfitting.

3.6 Causality

This section is mainly based on the study of (Dawid 2015), which describes the foundation and background of causality. The main reason for developing approaches on the basis of causality is to separate the system into cause and effect variables, for the purpose of identifying the effect caused by a certain type of intervention (Buzzoni 2014). Since the goal of the thesis is predicting the ACE for the patients, this is well suited for solving problems of determining the effects of applied causes; we mainly discuss under the decision-theoretic framework of statistical causality.

3.6.1 Cause-Effects

In statistical causality, most of the studies are focusing on the Cause of Effect (CoE) problems. The relationship between cause-effects is an event of cause leads to other events of effect happen; the effect has to occur after the cause in a temporal perspective. Simply, the main task of CoE is seeking future outcomes based on the observed dataset.

There is a famous sentence in statistics “correlation does not imply causation”; the reason behind this misconception due to the hidden variables, namely confounders, which cause the correlation. A classic example of coffee consumption and Nobel prizes is shown in Figure 3.24. Messerli et al. (2012) shows that there is a significant linear correlation between coffee consumption per capita and the number of Nobel prizes of a country. Nevertheless, there is no direct cause effect link between the coffee consumption and Nobel prizes vari-

ables, since we might avoid making a conclusion that drinking more coffee will more likely to win Nobel prize. The reason for this correlation arises might due to the hidden factor of wealth; it increases the financial funding of the education and research, which could cause the effect of raising the number of Nobel Prize winners.

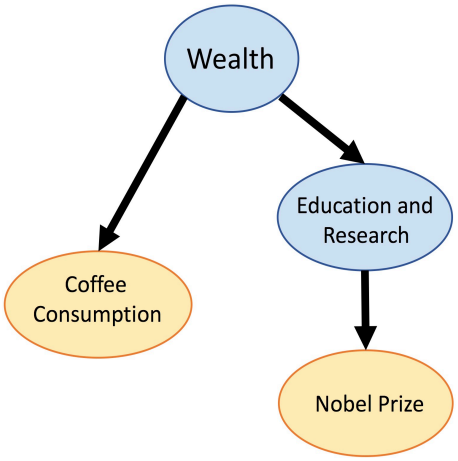


Figure 3.24: Visualize a strong correlation between coffee consumption and the number of Nobel prizes of a country. The yellow circles denote the observed variables of cause-effects; the blue ones indicate the variables regarding hidden factors.

3.6.2 Graphical Representation

A graph G is defined as $G = (V, E)$, it consists of nodes (vertices) V, and edges $\varepsilon \subseteq V^2$ with $(v, v) \notin \varepsilon$ for any $v \in V$. Vertices V denote the variables (events), and the edges correspond to cause-effects relationships in between (see Figure 3.24). Following the terminology in (Spirtes et al. 2000), G is called Directed Acyclic Graph (DAG) if there is a finite directed graph without the directed circle. The parent node of V represents the cause, and the child node is the effect caused by the parent node links to it. It is not allowed to appear more than one parent for a child node due to the immorality of DAG. Simply, one variable can cause many effects, but an effect can only belong to one cause. The important property of DAG is the acyclic structure can always lead to a topological ordering; thus, each event is directed from cause to effect in a sequence, and the topological ordering of DAG is not necessary to be unique.

3.6.3 Models

Based on the work of (Dawid 2015), there are several alternative models to formulate a decision problem of statistical causality. We mainly adopt two of them to analysis the problem of assessing the effects of given interventions on the outcomes.

3.6.3.1 The Decision-Theoretic Model

A general model adopted for determining treatment effects for the patients is a DT model. The essential attribute of this model is to allow simply two hypothesis distributions for the effect of Y. The cause is represented by a binary variable X; it can either set to be 0 or 1.

For simplicity, the distributions assumed to satisfy the normal distribution:

$$Y|(X = 0|X = 1) \sim \mathcal{N}(\mu_x, \sigma^2) \tag{3.95}$$

The probability density function of conditional distribution for the binary variable X:

$$P(Y|(X = 0|X = 1)) = (2\pi\sigma^2)^{-\frac{1}{2}} exp - \frac{(y - u_x)^2}{2\sigma^2} \tag{3.96}$$

3.6.3.2 The Functional Model

Based on the graphical representation of causality. The functional model can be written in a standard form from a mathematical perspective:

$$Y = f(X, U) \tag{3.97}$$

X denotes the cause of interest (parent), and Y represents the outcome of interest (child), where U denotes an additional noise term and which can be interpreted as a bivariate distribution, and it is independent of X.

3.6.4 Confounding

Whereas so far, we only make the assumption that the treatment assignments are under controlled rather than real-world circumstances, which indicates that we can decide to set $X = 1$ or $X = 0$. However, in the real-world, variable X is no longer fixed. For example, we only have the datasets are collected from either treated or untreated patients, since X is becoming a random variable, it is no longer our choice to offer which treatment anymore. Therefore, for the model without confounding, it is essential to specify the requirement that independence in the observed data between X and U (Dawid 1979). The functional model is shown in Equation 3.98. Here, the observational distribution of X is dependent on the additional extraneous random variable of U, $U \perp\!\!\!\perp X$ holds.

$$\begin{aligned} Y &= f(X, U) \\ X &\perp\!\!\!\perp U \end{aligned} \tag{3.98}$$

In order to introduce confounding, we need to talk about interventional regimes. These regimes describe the data generating rules, different data generating rule according to different confounding. Formally presenting regimes, a nonstochastic variable F_x is introduced, which includes values 0, 1, and $\emptyset$ (idle). $F_x = \emptyset$ indicates that all the available data have been observed, namely under the observational regime. $F_x = 1$ represents the data generating rule is under $X = 1$ (e.g. generating data according to all the patients who receive the active treatment), a so-called interventional regime. $F_x = 0$ indicates the interventional regime of $X = 0$.

Hence, confounding describes the relationship between these binary interventional regime variables. The absence of confounding turns out to be the observational regime, which is the observational data gathered in regime $F_x = \emptyset$.

3.6.5 Conditional Independence

According to the interventional regimes, we obtain the model function:

$$Y = f(X = x, U) \quad (x = 0, 1) \quad X \perp\!\!\!\perp U \; holds \tag{3.99}$$

For example, $Y = f(1, U)$ computes the treatment effect regarding the patients receiving treatments, in which the interventional regime $F_x = 1$. The ultimate goal is to measure the difference between two distributions under $F_x = 0$ and $F_x = 1$, which implies that the effect Y under its corresponding interventional regime.

Consider the conditional distribution with the observational regime $F_x = \emptyset$, since the multivariate normal distribution U has the same distribution in all the regimes, there exists a marginal distribution when $F_x = 1$. Thus, we can transfer this problem to $Y = f(1, U)$ given $X = 1$ in the observational regime $F_x = \emptyset$, the result distribution is the same as the distribution of $Y = f(1, U)$ under the regime of $F_x = 1$. symbol $\approx$ denotes that they have the same distributions.

$$(Y \mid F_x = x) \approx (Y \mid X = x; F_x = \emptyset) \;\; (x = 0, 1) \tag{3.100}$$

No matter we assign $X = 0$ or $X = 1$ in their corresponding regime of $F_x = 0$ or $F_x = 1$, the outcome probability always equals to 1 with $F_x = x$, then the conditional distribution of Y be written as follows:

$$(Y \mid X = x, F_x = x) \approx (Y \mid X = x; F_x = \emptyset) \;\; (x = 0, 1) \tag{3.101}$$

The independence between observational and interventional regimes enables us to conclude the conditional independence property (Dawid 1979). Again, the cause-effects problem converted to be a probabilistic one.

$$Y \perp\!\!\!\perp F_x \mid X \tag{3.102}$$

where the effect Y is independent of the intervention F_x given cause X (see Figure 3.25).

In the DAG, the fundamental core of the conditional independence property, that is no confounding for cause-effects, see Figure 3.25.

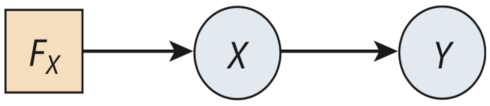


Figure 3.25: DAG with no confounding, image adopted from (Dawid 2015).

3.6.6 Markov Property

Markov property is widely used for dependency analysis on the basis of graphical modelling. The local Markov property defines as: given a DAG G, each variable in G only depends on its parent. The joint distribution of node X satisfies:

$$P(X) = \prod_{X_i \in X} P(X_i \mid Parents(X_i)) \tag{3.103}$$

The global Markov property with respect to the DAG G, which is equivalent to the local Markov property once the joint distribution has a density. For the disjoint nodes (or sets) A, B, C, A and B are d-separation (Geiger et al. 1990) by C, then A is independent of B given C.

$$A, B \; d-sep. \; by \; C \Rightarrow A \perp\!\!\!\perp B \mid C \tag{3.104}$$

Another way to demonstrate the d-separation of the global Markov property is by Markov equivalent. An immorality in G turns out to be a child has more than one parents, such as $A \rightarrow C \leftarrow B$. The following example clearly shows that (d) is violating the Markov global property as it has an immorality. (a), (b), and (c) are Markov equivalent, since they represent that $A \perp\!\!\!\perp C \mid B$, A is independent of C given B. (d) only shows that $A \perp\!\!\!\perp B$, A is independent of B. The following formulas show that DAGs with three nodes:

$$(a)\ A \rightarrow B \rightarrow C$$
$$(b)\ A \leftarrow B \leftarrow C$$
$$(c)\ A \leftarrow B \rightarrow C$$
$$(d)\ A \rightarrow B \leftarrow C$$

In the treatment effect problem, $Y \perp\!\!\!\perp F_x \mid X$ can be intuitively comprehended as the effects of a treatment only depend on which treatment is offered to patient by the doctor. The outcome with an active treatment will not be influenced by the outcome under the decision of controlled treatment. As already mentioned, this conditional independency property indicates that there is no confounding for cause-effects.

3.6.7 Confounder

In order to introduce the confounder, or namely confounding variables, the difference between confounding and confounder need to be clarified first. The confounder is not a component or requirement for confounding. As mentioned above, no confounding has to satisfy $Y \perp\!\!\!\perp F_T \mid T$. However, making the assumption of no confounding in the real-world seems unrealistic. Instead, the outcome Y is no longer independent of the option of a treatment: $Y \not\perp\!\!\!\perp F_T \mid T$, which is called confounding.

As already mentioned, the correlation between two events does not imply cause-effects; there

might be existing the (possibly hidden) variable that indirectly causes both simultaneously. The confounder is formally defined as: a variable satisfying sufficient covariate property, or so-called conditional independent of the covariate. Since the confounding in the real-case scenario is unavoidable, a variable U is named a covariate when it satisfies:

$$U \perp\!\!\!\perp F_T \tag{3.105}$$

$$Y \perp\!\!\!\perp F_T \mid (U, T) \tag{3.106}$$

Suppose we have a functional model $Y = f(T, U)$, and the purpose is to obtain a variable U that meets the requirement of influencing the pair of Y and T in the meantime. Hence, intuitively, the ultimate task of extracting the covariate variable for treatment effect, which is indicating that the variable U becomes the confounder in the model.

In Equation 3.109, rather than the conditioning holds for the interventional regime $(F_T|T)$, here we treat the functional model $f(T, U)$ as a component. The function is always applied regardless of the regime operating. Let us assume in the observational regime, $Y \mid (U, T = \emptyset)$, the treatment effect is evaluated only based on a paticular variable (e.g. gender) when the patient receiving active treatment $F_T = 1$. This $Y \mid (U, T = 1)$, which is equivalent to the treatment effect is obtained from this variable U under the interventional regime $F_T = 1$. Therefore, we can infer that $Y \mid (U, T = t)$ is equivalent to $Y \mid U$ in the interventional regime $F_T = t$, and $Y \perp\!\!\!\perp F_T \mid (U, T)$ also holds.

According to this sufficient covariate property, as shown in Figure 3.26, there is existing confounding for the T and Y pair. Either causal link a or b is missing, the variable U will become non-confounding.

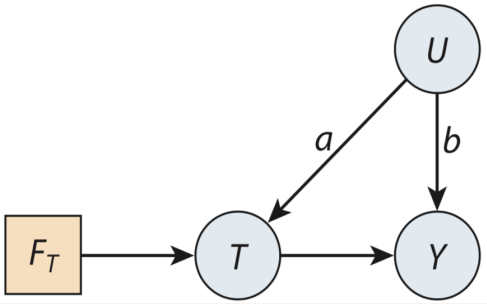


Figure 3.26: Illusion of the confounder U affects both T and Y with the observational regime $F_X = \emptyset$, image adopted from (Dawid 2015)

3.6.8 Causal Inference

Consider the DT model scenario: the patients are divided into two groups, each group can only be assigned one treatment $X = 1$ (e.g. offering a pain killer), or no treatment $X = 0$ (e.g. offering no pain killer). P_1, P_0 are the hypothesis distributions with these treatment

decisions accordingly. Y_i denotes a treatment outcome on the condition of whether the i-th individual receiving a treatment x or not, this can be denoted as $Y_i|X = x$. With the above information, we can randomly sample the distribution of P_1 based on the responses with the treated patients. Similarly, we could draw the distribution P_0 from untreated patients:

$$Y_i|X = x \sim \mathcal{N}(\mu_x, \sigma^2) \tag{3.107}$$

Here all the individual outcomes are independent, such that, solving causal inference is measuring the difference of two distributions:

$$\delta := E_{P1}(Y) - E_{P0}(Y) \tag{3.108}$$

3.6.9 Average Causal Effect

There are two hypothesis distributions of P_0, P_1 of Y under the conditions of $X = 0$ and $X = 1$. Since we are interested in the treatment effect on average, and the interventional regimes need to be involved in the DT model. Thus, ACE under the DT framework can also be defined as:

$$ACE := E(Y \mid F_T = 1) - E(Y \mid F_T = 0) \tag{3.109}$$

where P_1 for Y in regime $F_T = 1$, and P_0 for Y in regime $F_T = 0$. However, it is impossible to apply to the observational data in the real-world environment, the reason is, commonly we cannot have the data of treatment effect which collected from the patients receiving both active treatment and control treatment. In order to guide computation through such observational data, an assumption of the observational regime has to make. Consequently, the ACE can be obtained:

$$ACE := E(Y \mid T = 1, F_T = \emptyset) - E(Y \mid T = 0, F_T = \emptyset) \tag{3.110}$$

However, Equation 3.110 only holds under the assumption of $Y \perp\!\!\!\perp F_T \mid T$, there is no confounding on T, the reason belongs to the interventional regime of $Y \perp\!\!\!\perp F_T = t(t = 0, 1)$ is equivalent to the observational regime $Y \mid T = t, F_T = \emptyset$. Although Equation 3.110 seems to be reasonable in theory, as mentioned, even $Y \perp\!\!\!\perp F_T \mid T$ itself is not the case for confounding variable (confounder). Therefore, it might be much more realistic to consider confounder U for ACE, which is the concept of deconfounding.

According to the sufficient covariate attribute of confounder U, the confounding pair of Y, T in observational regime needs to be modified to Y given (U, T). Simply, given an observed data with partial patients receiving active treatment and the rest receiving control treatment, we can specify the different expectation of a specific treatment effect conditional on U. An intuitive way of understanding deconfounding for Specific Causal Effect (SCE) is to consider that an individual-level effect of Y given U is a random variable, this can be replaced by $Y = (U, T = t, F_T = t)$, then it turns out to be a function involving U. Additionally, the independent attribute of the individual-level causal effect also holds with the observational regime (see Equation 3.111). The probability of Y equals to 1 in each interventional regime, then we obtain:

$$SCE_U := E(Y \mid U, T = 1, F_T = \emptyset) - E(Y \mid U, T = 0, F_T = \emptyset) \tag{3.111}$$

Furthermore, the ACE with confounder is composed of the overall expectation of SCE. Additionally, confounder U is independent of F_T. We can define ACE based on SCE_U:

$$ACE = E(SCE_U \mid F_T = \emptyset) \tag{3.112}$$

Chapter 4

Model and Implementation

The goal of this thesis is to apply deep VIB model for solving cause effect inference. In this chapter, the details regarding the implementation of Cause-Effects with Deep Information Bottleneck (CEIB) model for evaluating ACE is presented. During the training process, CEIB learns a low-dimensional and continuous "bottleneck" representation of the confounding information, the so-called latent space. This representation enables us to predict the average treatment effect.

4.1 Cause-Effect Deep Information Bottleneck Model

In this paper, we are focusing on developing a CEIB model concerning the measured confounders in order to estimate the ACE performance of causality with the DT model (see subsubsection 3.6.3.1).

4.1.1 Measured Confounder

As discussed above, extracting confounders is an essential task for inferring a causal relationship from the observed dataset. With the formal definition of confounder Z, which is a variable that tends to affect both the treatment intervention T and the treatment outcome Y. The ultimate purpose of our work is to obtain these confounders by CEIB model.

The measured confounder implies that all the confounders are completely observable in a given data. None of them is hidden among the high-dimensional input of X. For example, with a high-dimensional input biological dataset which is collected from patients, the main goal is to directly "filter out" the important attributes from it, such as blood type, rather than searching for the ones beyond inputs.

Although, as pointed out in the real world, it might exist unmeasured (hidden) confounders can influence both Y and T as well. In our case, a treatment effect could be potentially affected by economic status and hygiene lifestyles or eating habit of a patient. However, these hidden confounders are commonly excluded from the biological data; they are more challenging to be collected due to the wide range of possibilities and uncertainties. In the

work of (Louizos et al. 2017); they employ the VAE model to estimate the causal effect with hidden confounders, which is known as CEVAE. Note that these two models have a similar model structure, except for an extra proxy variable (Montgomery et al. 2000) in the encoder, which is adopted for those hidden confounders. For instance, instead of measuring the economic status, we can simply use a "proxy" for accessing the information of the residence place and job title of the patient.

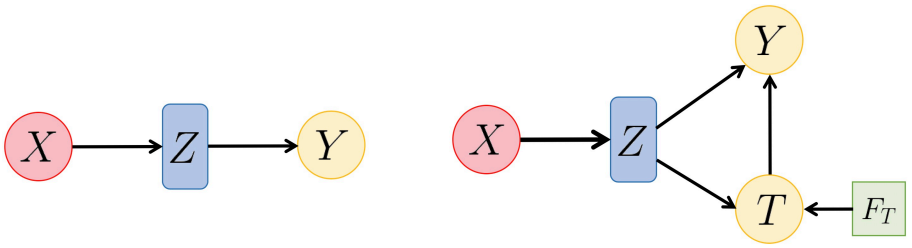
Contrary to the hidden confounders, we only consider solving the causal inference problem by targeting the measured confounders by CEIB model. The task is to extract the confounders directly within the scope of the existing variables in the given X. We are not expecting any unobserved feature in the data to determine the causal effects of an intervention in general.

4.1.2 CEIB Model Structure

Since the problem of causal inference has an additional treatment variable T in order to allow the interventional treatment F_T to impact on the outcome of Y. Based on the deep IB model we discussed in section 3.4, rather than using a single output variable Y in the decoder (see Figure 4.1 (a)), CEIB model replaces Y with a pair of T and Y. The interventional regime variable F_T (see subsection 3.6.4) enables us to control the effect comes from different interventional regimes.

With this design purpose, CEIB extends the decoder part of deep IB to be a structure of cause-effects cofounding (see Figure 4.1 (b)). Therefore, the latent variable Z in deep VIB model refers to the (measured) confounders in the CEIB model, since we would like to predict not merely the outcome of input X but also the causal effects based on the intervention regimes. Another explanation for CEIB model is that we aim to adopt a deep VIB for learning a medical therapy outcome Y (effect) under the condition the all the confounders are available in the input X (cause).

Figure 4.1 demonstrates the architecture differences between the two models of deep IB (VIB) and CEIB. With a set of data collected from patients, X denotes a high-dimensional input data which can be the features such as blood count or other genetic information. T is a binary treatment variable which is representing whether the patients are treated or not, such that the intervention variable F_T owns two states (on or off). Y is the outcome under a particular treatment, which is described by a number in a particular range for indicating the treatment effect, or "treatment score" for each patient. A low-dimensional latent space of Z represents the observed confounders we aim to learn by training.



(a) Deep IB model (b) CEIB model with measured confounding

Figure 4.1: Visualize the deep IB model and the CEIB model with measured confounding variables. The red circle denotes the variable in the encoder, and the yellow circle denotes the variable involving in the decoder. The blue rectangle represents the latent variable, and the green square indicates the intervention variable. The same way of expression is adopted in the following as well.

4.1.3 CEIB Objective

Based on the CEIB model structure in Figure 4.1, the objective function of CEIB can be rewritten as the deep IB combining with the confounding of cause-effects. It involves the treatment variable T into the model decoder. Formally, we cast it into the flowing optimization problem:

$$\min_{p(z|x)} J_{CEIB} = I(X;Z) - \beta I(Z;(T,Y)) \tag{4.1}$$

where the original output Y in the objective of deep IB is formalized in terms of causal effects of an intervention (T,Y) instead. Likewise, $I(X;Z)$ forms the model encoder for information compression purpose. It indicates how much of the information in the input X has been compressed into the compact representation of Z. For the decoder, $I(Z;(T,Y))$ denotes the quality of treatment effect prediction, which is measured by how much information of the treatment effect can be recovered under these two different interventional regimes.

The rationale behind the CEIB objective can be viewed as a process of "filtering information" as well. This compact representation of Z refers to the "bottleneck". CEIB operates to squeeze all the information regarding "source" X thorough the "bottleneck" for compression purpose, while preserving the "relevance" information about the pair of treatment effect (T,Y) at the same time. Ultimately, the most important information will stay in the "bottleneck".

4.1.4 Estimate ACE with CEIB Model

Our main goal for solving causal inference is to recover ACE building upon the CEIB framework under the setting of the DT model. According to the definition of ACE in Equation 3.109 in which $F_T = t$ is denoting the intervention of the treatment. We assume there are existing two interventional regimes, and they are defined as $F_T = 1$, which indicates that

a patient is offered an active treatment assignment, and $F_T = 0$ implies that a patient who is receiving a control treatment respectively. $F_T = \varnothing$ is the observational regime.

However, we have to confront the challenges in the real world scenario regarding interventional regimes. Evidently, we are only able to collect the patient data on the basis of the observational regime rather than the interventional regimes. A patient can only receive one type of treatment for testing apparently, either with outcome Y_1 for receiving treatment assignment, or Y_0 for control treatment assignment. To solve this problem, we need to expect that the observational ACE is equivalent to the real value of ACE we would like to gain. As discussed in subsection 3.6.9, this can be achieved as long as we make an assumption of conditional independence $Y \perp\!\!\!\perp F_T \mid T$, in order to ignore the single treatment assignment policy (one patient could not offer both controlled and uncontrolled treatment for a particular type of medication). Therefore, the outcome distribution of $Y \mid T$ appears to be identical in the observational regime $F_T = \varnothing$ and intervention regime $F_T = t$. Consequently, the real value of ACE could be inferred from the ACE with the observational regime $F_T = \varnothing$ as well (see Equation 3.110). We are able to compute the average level of the expectation for treatment effect.

4.2 Implementation

In this section, we explain the architecture of the CEIB model in details, and the variational bound of CEIB objective will be examined. Finally, we demonstrate that the reparameterization trick which is involved in the model.

4.2.1 CEIB Model Implementation

The graphical illustration of the CEIB model for implementation is shown in Figure 4.2, which is consist of a pair of DNNs: an encoder and a decoder network. For the encoder network, the distribution of $q(z_i|x_i)$ is employed to approximate the posterior, with the training parameters μ_i and σ_i^2.

$$q(z_i|x_i) = \prod_{j=1}^{D_z} \mathcal{N}\left(\mu_{ij} = \overline{u}_{ij},\ \sigma_{ij}^2 = \overline{\sigma}_{ij}^2\right) \quad \overline{u}_{ij}\ \overline{\sigma}_{ij}^2 = f(x_i) \tag{4.2}$$

x_i denotes the input datapoints, j corresponds to each dimension of x_i, namely the covariate. D_z indicates the dimension of the latent space. $f(\cdot)$ is a deep network of encoder and parametrized by its own parameters.

In the decoder network, since the distribution of the outcome of Y depends on the treatment group T which the patients are assigned in. We introduce an auxiliary distribution to compute the treatment probability of t_i before sampling the new outcome of y_i. More precisely, we employ a Bernoulli distribution of $p(t_i|z_i)$ for evaluating the treatment assignment T (see Equation 4.4). Subsequently, by employing the distribution of $p(y_i|t_i, z_i)$ based on

the result we obtained from the Bernoulli distribution, then the outcome of treatment result y_i can be sampled by Equation 4.5 as following:

$$p(z_i) = \prod_{j=1}^{D_z} \mathcal{N}(z_{ij}|0,1) \tag{4.3}$$

$$p(t_i|z_i) = \text{Bernoulli}(\sigma(g_1(z_i))) \tag{4.4}$$

$$p(y_i|t_i,z_i) = \mathcal{N}(\mu_i = \hat{\mu}_i, \sigma^2 = \hat{v}) \quad \hat{\mu}_i = t_i g_2(z_i) + (1-t_i)g_3(z_i) \tag{4.5}$$

where $p(z_i)$ is the prior; it follows a standard normal distribution. This essential strategy for modelling the continuous latent space we have discussed in subsubsection 3.3.2.2 and section 3.5 respectively. $\sigma(\cdot)$ denotes the logistic function and g_k corresponds to the deep network of the decoder. The discrete probability distribution $p(t_i|z_i)$ of treatment assignment is sampled by the Bernoulli distribution. The continuous outcome of $p(y_i|t_i,z_i)$ is sampled by the Gaussian distribution based on the mean $\hat{\mu}_i$ that is obtained from a TARnet architecture (see Equation 4.5 (Shalit et al. 2017) and the fixed variance $\hat{v}$.

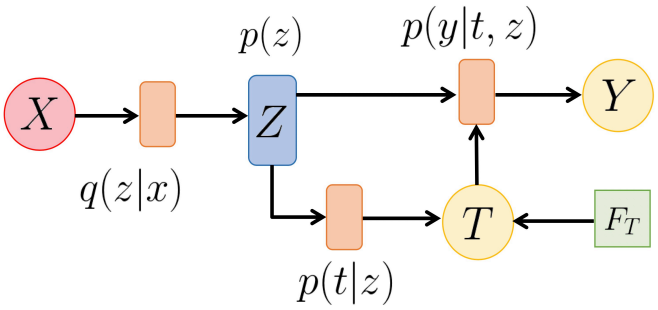


Figure 4.2: Visualize the overall architecture of the model CEIB. $q(\cdot)$ denotes the encoder distribution and $p(\cdot)$ denotes the decoder distribution. X represents an input data, which is involving discrete and continuous data. T is a binary variable that represents whether the patients are treated or not, where the intervention variable F_T with two states (on or off). Y is the outcome under a particular treatment, which is described by a number in a certain range. Z denotes the low-dimensional latent space, it contains the measured confounders. The orange rectangles correspond to draw the samples from the specific distribution with the parameters which are trained by parametrized deterministic DNNs. The circles imply that they are observed variables, the blue rectangle indicates the latent and continuous variables.

Based on the technique that we used for examining the bound of deep IB , therefore, we can rewrite the CEIB objective with the following variational bound according to the deep VIB objective we discussed in Equation 3.93:

$$\begin{aligned} &\min_{p(z|x)} J_{CEIB} = I(X;Z) - \beta I(Z;(T,Y)) \\ &\approx \frac{1}{N}\sum_{i=1}^{N}\left[\int q\left(z_i|x_i\right)\log\frac{q\left(z_i|x_i\right)}{p(z_i)}\,dz - \beta\int q\left(z_i|x_i\right)\log p\left(y_i|t_i,z_i\right)dz\right] \end{aligned} \tag{4.6}$$

Recall that the true probability distributions of both mutual information terms are unsolvable directly without VIB. That is the reason why we use a variational approximation towards the "true" distributions by introducing the auxiliary distributions. Based on the non-negative property of the KL divergence, it leads an inequality and such that we are able to create a bound for CEIB objective, and finally cast the CEIB objective to be an optimization problem. The details discussed in subsection 3.5.1.

Eventually, with the definition of KL divergence in subsubsection 3.3.2.1, we could update the CEIB objective in Equation 4.6 as following:

$$\min_{p(z|x)} J_{CEIB} = \overbrace{\underbrace{D_{KL}\big(q(Z|x_i) \,\|\, p(Z)\big)}_{\text{Encoder}}}^{\textbf{KL divergence}} - \beta \overbrace{\underbrace{E_{z\sim q(z|x)}\left[\log p(y_i|t_i, z_i)\right]}_{\text{Decoder}}}^{\textbf{Log-likelihood}} \tag{4.7}$$

where $p(Z)$ is known as a prior, which is defined as a fixed Z-dimensional spherical Gaussian distribution, $p(Z) = \mathcal{N}(0, I)$. The first KL term represents the encoder network. The second log-likelihood term of $p(y_i|t_i, z_i)$ represents the decoder network, which is denoting the causal effect reconstruction accuracy based on the interventional regimes. We use the decoder network to measure the performance of the trained model. Here β tends to control the trade-off between the causal inference accuracy and the level of complexity (compression) of the model.

4.2.2 Reparameterization Trick for CEIB

CEIB use the same strategy of reparameterization trick (see subsubsection 3.3.2.3) for backpropagation during the training process. Without this methodology, it can only allow us to directly sample from the distribution of $q(z|x)$, which leads to the non-deterministic property of z and the sampling process relies on the parameter of the encoder, thus the partial derivative of each variable is not able to compute for backpropagation.

As mentioned in VAE and deep IB, rather than sampling from $z \sim q(z|x) = \mathcal{N}\left(\mu, \sigma^2\right) = \mathcal{N}\left(z|f_\mu(x), f_\Sigma(x)\right)$, the reparameterization trick introduces an additional noisy term of ϵ (see Figure 4.3). We only draw samples from this auxiliary noise variable $\epsilon \sim \mathcal{N}(0, I)$. Thereby, $E_{\mathcal{N}(z;\mu,\sigma^2)}[f(z)] = E_{\mathcal{N}(\epsilon;0,1)}[f(\mu + \sigma \odot \epsilon)]$ (Alemi et al. 2016), and z becomes a deterministic node in this case with $f(z) = f(\mu + \sigma \odot \epsilon)$, therefore, the expectation turns out to be differentiable and the backpropagation is permitted:

$$\min_{p(z|x)} J_{CEIB} = D_{KL}\left(q\left(Z|x_i\right) \,\|\, p\left(Z\right)\right) - \beta\, E_{\epsilon\sim p(\epsilon)}\left[\log p\left(y_i|f\left(x_i, \epsilon\right), t_i\right)\right] \tag{4.8}$$

where $f(\cdot)$ is the encoder network, based on the setup above, it allows us to estimate the lower bound of CEIB. Eventually, as shown in Figure 4.3, this architecture allows us to learn a low-dimensional latent space of Z, which is turning out to be a multivariate Gaussian distribution due to the prior satisfies $\mathcal{N}(0, I)$, it contains the measured confounders that

we want to obtain. This enables CEIB to approximate the effects of a given T on Y gradually.

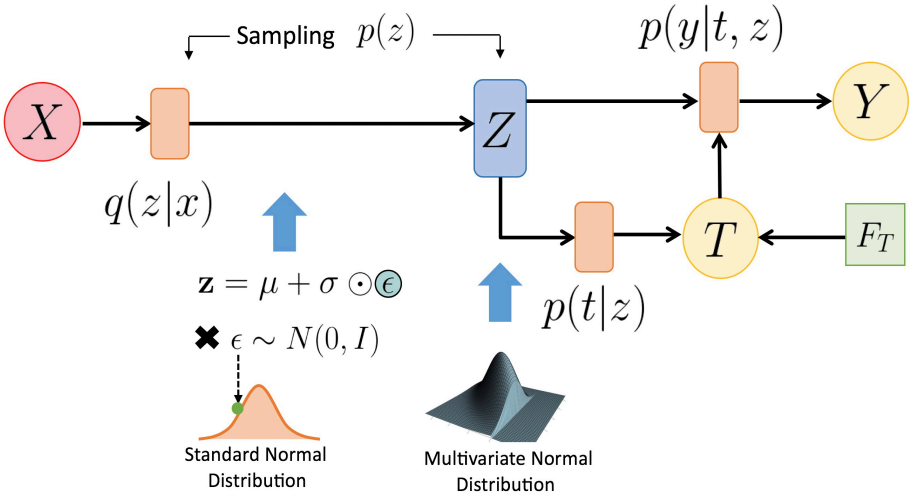


Figure 4.3: Visualize the reparameterization trick for CEIB model. Reparameterization trick adds extra stochastic input of $\mathcal{N}(0, I)$ for sampling latent variables of μ and σ. This trick allows the backpropagation due to deterministic property of $p(Z)$. The latent space obtained by CEIB is a multivariate normal distribution. Here we plot the multivariate normal distribution of the latent space is in two-dimensional case (two covariates).

Chapter 5

Experiments

Evaluating the CEIB usually considered to be a difficult problem due to the lack of ground truth in the real world. In this section, we will evaluate the previously introduced CEIB model to solve a high dimensional real-world task. In order to overcome this barrier, the synthetic or semi-synthetic benchmark dataset from (McCormick et al. n.d.), which is commonly used for the studies of the evaluating causal inference. Our experiment approach based on the semi-synthetic benchmark dataset IHDP (Hill 2011), where the real data is created with the fully known treatment outcomes which correspond to the patients' treatment assignments, such that it is not necessary to model the proxy variables.

For the implementation requirement, we use TensorFlow (Abadi et al. 2016) and a Python library Edward (Tran et al. 2016) for probabilistic modelling. In our experiment, for the architecture of neural networks, we use five hidden layers with ELU activation function. We train the CEIB by Adam optimizer with a learning rate of 0.0001. The dimensionality of the latent space Z is set to be 20.

Infant Health and Development Program The dataset we adopt in our experiment is from the IHDP, which is a randomized control experiment targeting the low birth weight and premature infants who were born between 1985 and 1988 at eight different sites. This experiment aims to study the impact of the childcare service on these premature infants, and the measurements collected from both mothers and infants. The assigned treatment group receives the high-quality and intensive childcare and home visits from a trained specialist, and the other control group is also provided with the home visits to record the test scores. Briefly, we would like to approximate the effect of the given childcare intervention on the outcomes of the cognitive test score.

To construct a "true" individual-level causal effect of the treatment, (Hill 2011) create an observational study with a strategy of treatment assignment "de-randomized". With the extracting features, by throwing away the children with non-white mothers from the treatment group, then a non-random portion of the treated set is removed, and the remaining control group kept intact. Therefore it leads to introduce a biased estimation to the dataset regarding ethnicity, a treated and a control outcome are allowed to be simulated by this artificial

operation of removing a portion of data.

The IHDP dataset consists of 747 subjects in total: the remaining 139 children samples in the treatment group, and 608 in the comparison control group. Each unit (sample) including 25 covariates, which is involving both continuous and binary features. These covariates are collected from the infants and their mothers. Data related to the children such as gender, birth order, head circumference, neonatal health index. The dataset regarding mothers collected when they gave birth, which is including such as marital status, age, educational attainment, and whether she worked during pregnancy. Each sample unit is also composed of the observed outcomes of the test sore y corresponding to the treatment group t. Additionally, the mean of the true potential outcomes, they are denoted by μ_0 and μ_1 regarding two treatment assignments, respectively. With $\mu_1 - \mu_0$, it allows us to compute the "true" ACE by measuring the difference between the outcome distributions of y_1 and y_0, this can be adopted by evaluating the performance of our predicting ACE afterwards.

5.1 Information Curve

In order to analyze the interpretation referred to the latent confounding of IB, we would like to measure how the value of the tradeoff parameter β changing could potentially influence the performance of the CEIB model. According to the CEIB objective (in Equation 4.1), we conduct an experiment to plot the information curve for demonstrating the relationship between two terms of $I(X;Z)$ and $I(Z;(T,Y))$ in the latent space. We set the number of epochs to be 250000, and the initial value of β equal to 1. During the training process, we gradually increase β-value by multiplying 1.01 with the step size of each 200 epochs. Notice that during the implementation of collecting information of $(Z;(T,Y))$, we need to consider the entropy of the ground truth label Y due to Equation 3.79:

$$(Z;(T,Y)) = E\left[\log p\left(y_i|t_i,z_i\right)\right] + H(Y) \tag{5.1}$$

$$\text{with} \quad H(Y) = -E_{p(Y)} \log p(Y). \tag{5.2}$$

5.1.1 Results

We plot the information curve based on the CEIB model that we designed, which is shown in Figure 5.1. We observe that the relationship between these two terms of mutual information under the condition of raising the value of β. It turns out to be the accuracy reconstruction output $I(Z;(T,Y))$ increase with the growth of the latent confounding information of $I(X;Z)$. The information curve grows smoothly until saturating at the point where the β-value is around 16671, along with the $I(Z;(T,Y))$ reaches around 0.7006 and $I(X;Z)$ with around 603.

Forming the information curve can be separated into two phases. In the initialization phase, the compression term of $I(X;Z)$ and prediction term of $I(Z;(T,Y))$ both tend to approach 0. When $I(X;Z)$ approximates to 0, which is indicating and the latent space Z becomes completely independent of the input X; as a result, it is unquestionable that there is no useful

information in latent confounding for predicting cause-effects at this point, the reconstruction accuracy $I(Z;(T,Y))$ also stay around 0 before $I(X;Z)$ starts to increase. This is implying that the latent confounding of the model has not started learning any information in this initialized training phase.

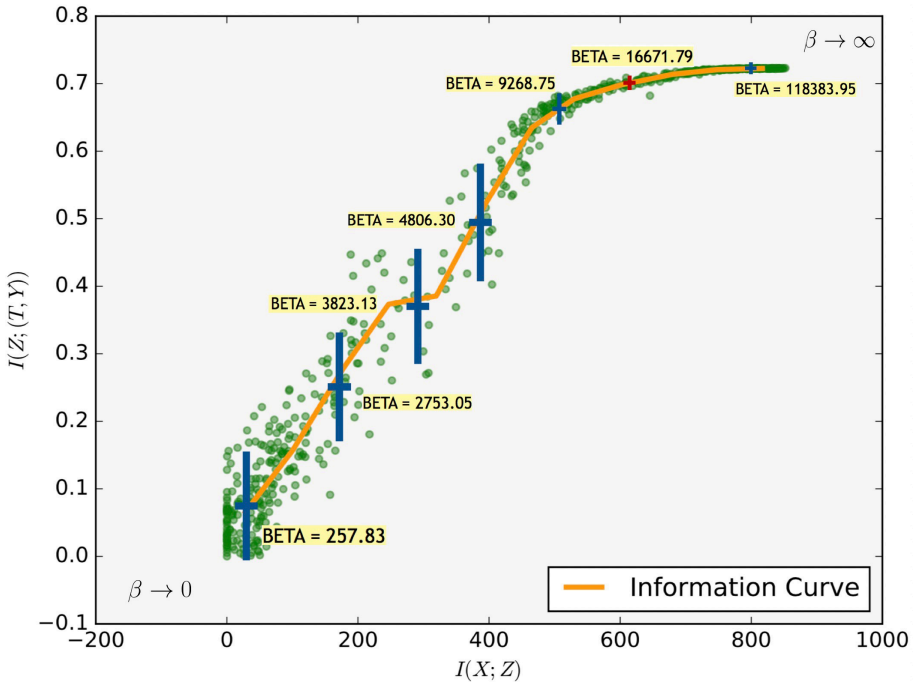


Figure 5.1: Illustrate the plot of the information curves regarding $I(X;Z)$ and $I(Z;(T,Y))$ with de-randomized IHDP dataset. The blue stars denote the corresponding β-value for a specific point on the information curve. The red star represents the information curve achieves the best performance with the converging point of $I(X;Z) \approx 603$ and $I(Z;(T,Y)) \approx 0.7$ with β-value around 16671. Note that the higher value of $I(X;Z)$ indicates less compression on the latent space. The higher value of $I(Z;(T,Y))$ indicates more accuracy for IBCE to predict the ACE performance.

During the IB curve forming phase, after $I(X;Z)$ achieving a small number (close to 0) for a while, both values of the mutual information terms start to grow from a certain point until β-value reaches around 250 with the iteration of 111800 in our case. From this period, the information curve begins to climb up from the lower-left corner. $I(X;Z)$ is measured by the KL divergence in practice. Since the latent space in CEIB model gradually "learning" information, there are more and more measured confounders, or so-called the useful features being extracted during this phase. It leads the decoder of $I(Z;(T,Y))$ tends to reduce the reconstruction error as well. More precisely, with the growing demands on the log-likelihood of $I(Z;(T,Y))$, $I(X;Z)$ in the encoder is also required to become higher in order to force latent space Z to reduce the compression degree.

Discussion We use the process of constantly compression and reconstruction (information) for gaining generalization ability for our model. In other words, by compressing the resource information in order to obtain the relevant factors about the treatment effects in the output, it encourages the "bottleneck" in the model to gain a generalized and continuous representation for predicting, this representation turns out to be the latent confounding we would like to obtain.

Above all, since the compression parameter β keeps increasing, the level of model compression becomes smaller with the increasing of reconstruction accuracy. However, according to the subsection 3.1.4, the best performance of Z requires the model to minimize the value of $I(X;Z)$ (maximize the compression level) and maximize $(Z;(T,Y))$ in the meantime. It confirms that along with the growth of β, the model will achieve a suitable performance at the convergence point on the IB curve.

Theoretically, under the extreme condition of $\beta \to \infty$, the model desires to preserve all the trivial information about treatment effect in the decoder and completely ignore the compression by the encoder. The compression term of $I(X;Z)$ tends to exceed the "minimum barrier" at a certain point, along with the maximum treatment effects accuracy, which would cause a performance drop-off after curve converging, since we want to gain the representation of $I(X;Z)$ as compact as possible under the condition that the CEIB model owns the best performance. In contrast, with another extreme condition of $\beta \to 0$, $I(Z;(T,Y))$ of the decoder is driven to equal to 0 as well. Without its role in accuracy prediction, the encoder would perform data compression to the largest possible extent such that $I(X;Z) = 0$. Thereby, the latent confounding is not containing any information in this case.

5.2 IHDP Performance

In this experiment, the same neural network configuration and the IHDP datasets in the first experiment were adopted for model training. With the CEIB model established, our experiment data split into 60/30/10% into training/validation/testing sets, along with 1000 replications of the simulated outcome.

Based on the real outcomes y_1 and y_0 corresponding to $t = 1$ and $t = 0$ provide by IHDP, we evaluate the CEIB performance by computing the average treatment effect:

$$ACE = \left| \frac{1}{N} \sum_{i=1}^{N} \left(\hat{y_1} - \hat{y_0} \right) - \frac{1}{N} \sum_{i=1}^{N} \left(y_1 - y_0 \right) \right| \tag{5.3}$$

where $\hat{y}_1, \hat{y}_0$ are the estimating results of our model. We train our model with a fixed parameter of $\beta = 16671$, which is the convergence point obtained from information curve based upon the first experiment.

5.2.1 Results

After running the dataset average over 10 train/validation/test splits, we compare the ACE performance of our model to several existing baselines (see Appendix B). The results are shown in table 5.1. It turns out that with the aid of IB curve, CEIB obtains an optimal confounding space under the sufficient statistical analysis of the β-value. CEIB offers state-of-the-art performance; it outperforms the CEVAE with the fixed $\beta = 1$, while achieving the best estimation among the existing baselines on both within-sample and out-of-sample ACE.

Method	$\epsilon_{\text{ACE}}^{\text{within-s}}$	$\epsilon_{\text{ACE}}^{\text{out-of-s}}$
OLS-1	$.73 \pm .04$	$.94 \pm .06$
OLS-2	$\mathbf{.14 \pm .01}$	$.31 \pm .02$
KNN	$\mathbf{.14 \pm .01}$	$.79 \pm .05$
BLR	$.72 \pm .04$	$.93 \pm .05$
TARnet	$.26 \pm .01$	$.28 \pm .01$
BNN	$.37 \pm .03$	$.42 \pm .03$
RF	$.73 \pm .05$	$.96 \pm .06$
CFRW	$.25 \pm .01$	$.27 \pm .01$
CEVAE	$.34 \pm .01$	$.46 \pm .02$
CEIB ($\beta \approx 16671$)	$\mathbf{.14 \pm .046}$	$\mathbf{.26 \pm .089}$

Table 5.1: Within-sample (training) and out-of-sample (testing) mean and standard errors for the metrics for the following models on the IHDP dataset. The smaller value implies better performance, as the desired predicting outcome will come closer to the true one. The bold values correspond to the model with the best performance.

Discussion In this setting, we modify CEIB with $\beta = 1$ in order to test the causal effect performance under the VAE framework. The score of CEIB turns out to be: 0.93 ± 0.264 and 0.61 ± 0.404 for within-sample and out-of-sample ACE. This performance is worse than CEVAE ($\beta = 1$), and the performance drawback is caused by the KL divergence of $I(X; Z)$ in our CEIB model still stays close to 0 when $\beta = 1$; it leads to an extremely small amount of information is existing in the latent space for predicting ACE. Therefore, CEIB could only achieve an overall average performance against the existing baselines under the VAE ($\beta = 1$) framework.

Using IB method to infer the causal effect of treatment is more robust and flexible compared with the VAE approach since VAE is only a particular case of the IB method. The information curve can be treated as a "preview window" for displaying all the CEIB model's performances with different β-values. It intuitively reveals the entire procedure of ACE performance changing under the DT framework, offering us a sufficient statistic for the convergence region on information curve regarding the optimal latent confounding. Therefore, we are allowed to choose a β-value to obtain an optimal tradeoff between encoder and decoder

networks.

Chapter 6

Conclusion

In this paper, we have introduced a model which is attempting at combining two ideas together: the causal inference and the deep IB method. The original IB principle is inspired by the rate distortion theory related to information compression, which is extended to the concept of deep IB with DNNs. The latent variable Z in the model represents the latent confounding we aim to construct, this low-dimensional continuous representation enables the most useful features among a high-dimensional input to be identified, which is known as the measured confounders for affecting both the treatment effect Y and the interventional treatment T. This latent variable can be interpreted as playing the role of "bottleneck" in order to squeeze out all the information in an input X through it, furthermore, it is leaving us only the most relevant parts about the output of Y. More specifically, deep IB behaves in the way that the encoder tends to compress the input as much as possible, on the other hand, the decoder aims to preserve the relevant information about Y to the largest extent possible. In the perspective of machine learning point of view, IB method becomes more powerful and robust due to the improvement of the generalization ability via the controlling with β.

Solving the cause-effects problem with a DT approach, we apply the practice-oriented variational approach to deep IB model, which is known as deep VIB, in order to ultimately approximate the effects of a given T on Y. CEIB describes as how much information about the treatment effect $I(Z;(T,Y))$ we are willing to lose by compressing X into Z. This rationale can be interpreted as the original information in the input is mapped to a more compact latent space, although we sacrifice a certain amount of prediction accuracy, and as a result, the generalization ability of model will be improved correspondingly due to the reduction of model's complexity. Hence the overall performance of the CEIB model should improve significantly. In the application of evaluating the ACE, we introduce a causal inference benchmark using data of IHDP. We also compared our method regarding VAE framework with the model of the CEVAE with additional proxy variables. The results show that CEIB makes state-of-the-art predictions of the ACE regarding measured confounding.

Besides, we examined the CEIB model performance based on the information bottleneck curve that is gained by gradually increasing the β-value. Each point for forming this curve with a specific β-value, which is demonstrating the relationship between corresponding pair of $I(X;Z)$ and $I(Z;(T,Y))$. Information curve is a powerful tool for the IB model; it

shows us a "shortcut" for accessing the optimal model performance under the controlling of β. Along with increasing β, the causal model achieves the best performance when the information curve is tending to converge. CEIB owns the best generalization ability within this saturate region since it contains the $minI(X;Z)$ and $maxI(Z;(T,Y))$ simultaneously. Consequently, we are able to analyze the relationship between the compression parameter of β and the performance of the IB model in order the obtain the optimal score of ACE.

Finally, we are allowed to draw a connection between two frameworks of VAE and deep IB based on the causal inference. In the application of both models, we have shown that VAE turns out to be a special case of deep IB approach with a fixed β-value ($\beta = 1$). The IB framework provides more flexibility for the model with modifying the tradeoff-controlling parameter, and hence allows us to pick a suitable β-value to obtain the CEIB model with the best performance. Additionally, the optimal performance appears to be a sufficient statistic, which provides a guarantee for performance reliability.

Future Work Solving the problem of extracting the relevant aspects from data is a vital research topic in machine learning. Not only for obtaining measured confounders, but CEIB can also be improved by including the hidden confounding in future work. Similar to CEVAE, the additional proxy variable can be involved in the encoder network, in order to implicitly infer the hidden factors beyond the input dataset that could influence the cause-effects.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M. et al. (2016), 'Tensorflow: Large-scale machine learning on heterogeneous distributed systems', *arXiv preprint arXiv:1603.04467* .

Alemi, A. A., Fischer, I., Dillon, J. V. & Murphy, K. (2016), 'Deep variational information bottleneck', *arXiv preprint arXiv:1612.00410* .

Amine, M. (2018), 'Deep learning for natural language processing (nlp) using variational autoencoders (vae)'.

Blundell, C., Cornebise, J., Kavukcuoglu, K. & Wierstra, D. (2015), 'Weight uncertainty in neural networks', *arXiv preprint arXiv:1505.05424* .

Bre, F., Gimenez, J. M. & Fachinotti, V. D. (2018), 'Prediction of wind pressure coefficients on building surfaces using artificial neural networks', *Energy and Buildings* **158**, 1429–1441.

Buzzoni, M. (2014), 'The agency theory of causality, anthropomorphism, and simultaneity', *International Studies in the Philosophy of Science* **28**(4), 375–395.

Chechik, G., Globerson, A., Tishby, N. & Weiss, Y. (2005), 'Information bottleneck for gaussian variables', *Journal of machine learning research* **6**(Jan), 165–188.

Cover, T. M. & Thomas, J. A. (2012), *Elements of information theory*, John Wiley & Sons.

Dawid, A. P. (1979), 'Conditional independence in statistical theory', *Journal of the Royal Statistical Society: Series B (Methodological)* **41**(1), 1–15.

Dawid, A. P. (2015), 'Statistical causality from a decision-theoretic perspective', *Annual Review of Statistics and Its Application* **2**, 273–303.

Geiger, D., Verma, T. & Pearl, J. (1990), d-separation: From theorems to algorithms, *in* 'Machine Intelligence and Pattern Recognition', Vol. 10, Elsevier, pp. 139–148.

Hill, J. L. (2011), 'Bayesian nonparametric modeling for causal inference', *Journal of Computational and Graphical Statistics* **20**(1), 217–240.

Hornik, K., Stinchcombe, M. & White, H. (1989), 'Multilayer feedforward networks are universal approximators', *Neural networks* **2**(5), 359–366.

Kingma, D. P. & Welling, M. (2013), 'Auto-encoding variational bayes', *arXiv preprint arXiv:1312.6114* .

Louizos, C., Shalit, U., Mooij, J. M., Sontag, D., Zemel, R. & Welling, M. (2017), Causal effect inference with deep latent-variable models, *in* 'Advances in Neural Information Processing Systems', pp. 6446–6456.

McCormick, M. C., Brooks-Gunn, J. & Buka, S. L. (n.d.), 'Infant health and development program, phase iv, 2001-2004 [united states]. 2013. doi: 10.3886', *ICPSR23580. v2* .

McCormick, M. C., Brooks-Gunn, J., Buka, S. L., Goldman, J., Yu, J., Salganik, M., Scott, D. T., Bennett, F. C., Kay, L. L., Bernbaum, J. C. et al. (2006), 'Early intervention in low birth weight premature infants: results at 18 years of age for the infant health and development program', *Pediatrics* **117**(3), 771–780.

Messerli, F., Sarmadi, B., Aminuddin, F., Hamid, M., Saari, N., Abdul-Hamid, A. & Ismail, A. (2012), 'Chocolate and your health', *N Engl J Med* **367**(16), 1562–4.

Minsky, M. & Papert, S. A. (2017), *Perceptrons: An introduction to computational geometry*, MIT press.

Mohamed, S. & Lakshminarayanan, B. (2016), 'Learning in implicit generative models', *arXiv preprint arXiv:1610.03483* .

Montgomery, M. R., Gragnolati, M., Burke, K. A. & Paredes, E. (2000), 'Measuring living standards with proxy variables', *Demography* **37**(2), 155–174.

Parbhoo, S., Wieser, M. & Roth, V. (2018), 'Cause-effect deep information bottleneck for incomplete covariates', *stat* **1050**, 8.

Pearl, J. (2000), *Causality: models, reasoning and inference*, Vol. 29, Springer.

Roberts, A., Engel, J., Oore, S. & Eck, D. (2018), Learning latent representations of music to generate interactive musical palettes., *in* 'IUI Workshops'.

Roberts, S., Guilford, T., Rezek, I. & Biro, D. (2004), 'Positional entropy during pigeon homing i: application of bayesian latent state modelling', *Journal of theoretical biology* **227**(1), 39–50.

Russell, S. J. & Norvig, P. (2016), *Artificial intelligence: a modern approach*, Malaysia; Pearson Education Limited,.

Schmidhuber, J. (2015), 'Deep learning in neural networks: An overview', *Neural networks* **61**, 85–117.

Shalit, U., Johansson, F. D. & Sontag, D. (2017), Estimating individual treatment effect: generalization bounds and algorithms, *in* 'Proceedings of the 34th International Conference on Machine Learning-Volume 70', JMLR. org, pp. 3076–3085.

Shannon, C. E. (1948), 'A mathematical theory of communication', *Bell system technical journal* **27**(3), 379–423.

Slonim, N. (2002), The information bottleneck: Theory and applications, PhD thesis, Citeseer.

Spirtes, P., Glymour, C. N., Scheines, R., Heckerman, D., Meek, C., Cooper, G. & Richardson, T. (2000), *Causation, prediction, and search*, MIT press.

Tishby, N., Pereira, F. C. & Bialek, W. (2000), 'The information bottleneck method', *arXiv preprint physics/0004057* .

Tran, D., Kucukelbir, A., Dieng, A. B., Rudolph, M., Liang, D. & Blei, D. M. (2016), 'Edward: A library for probabilistic modeling, inference, and criticism', *arXiv preprint arXiv:1610.09787* .

van Gerven, M. & Bohte, S. (2018), *Artificial neural networks as models of neural information processing*, Frontiers Media SA.

Appendix

A Calculation of KL divergence of $-D_{KL}(q_\phi(z|x) \parallel p_\theta(z))$ (Gaussian case).

The KL divergence is one of the variational lower bound (ELBO) component.

Given: The prior $p_\theta(z)$ and the posterior approximation $q_\phi(z|x)$ are both Gaussians.

$$p_\theta(z) = \mathcal{N}(0, I)$$
$$q_\phi(z|x) = \mathcal{N}(z; u_I(x, \phi), \sigma_I^2(x, \phi))$$

Calculation: $-D_{KL}(q_\phi(z|x) \parallel p_\theta(z)) = \int q_\phi(z|x) \log p(z) dz - \int q_\phi(z|x) \log q_\phi(z|x) dz$

$$\begin{aligned}
\int q_\phi(z|x) log\, p(z) dz &= \int \mathcal{N}(z; \mu, \sigma^2) \log \mathcal{N}(z; 0, I) dz \\
&= E_{z\sim\mathcal{N}(\mu,\sigma^2)} \left[\log \mathcal{N}(z; 0, I)\right] \\
&= E_{z\sim\mathcal{N}(\mu,\sigma^2)} \left[\log(\frac{1}{\sqrt{2\pi}} e^{\frac{z^2}{2}})\right] \\
&= -\frac{1}{2} \log 2\pi - \frac{1}{2} E_{z\sim\mathcal{N}(\mu,\sigma^2)} \left[z^2\right] \\
&= -\frac{1}{2} \log 2\pi - \frac{1}{2}(\mu^2 + \sigma^2)
\end{aligned}$$

And:

$$\begin{aligned}
\int q_\phi(z|x) \log q_\phi(z|x) dz &= \int \mathcal{N}(z; \mu, \sigma^2) \log \mathcal{N}(z; \mu, \sigma^2) dz \\
&= E_{z\sim\mathcal{N}(\mu,\sigma^2)} \left[\log \mathcal{N}(z; \mu, \sigma^2)\right] \\
&= E_{z\sim\mathcal{N}(\mu,\sigma^2)} \left[\log(\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(z-\mu)^2}{2\sigma^2}})\right] \\
&= -\frac{1}{2} \log 2\pi - \frac{1}{2} \log \sigma^2 - \frac{1}{2\sigma^2} E_{z\sim\mathcal{N}(\mu,\sigma^2)} \left[(z-\mu)^2\right] \\
&= -\frac{1}{2} \log 2\pi - \frac{1}{2}(\log \sigma^2 + 1)
\end{aligned}$$

Therefore:

$$-D_{KL}(q_\phi(z|x) \parallel p_\theta(z)) = \frac{1}{2}\sum_{j=1}^{J}\left[1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2\right]$$

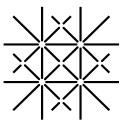
With:

$$\mu_{I_i}(x, \phi) = W^{(l)}_{\mu_{(I_i)}} h^{(l)} + b^{(l)}_{\mu_{I_i}}$$
$$\sigma_{I_i}(x, \phi) = W^{(l)}_{\sigma_{(I_i)}} h^{(l)} + b^{(l)}_{\mu_{I_i}}$$

where I denotes the last layer of encoder network, and μ_j and σ_j denote the j-th element of these vectors.

B Infant Health and Development Program: Baselines

The existing baselines we use to compare with the performance of the CEIB model is adapted from (Louizos et al. 2017). OLS-1 stands for the ordinary least squares, which is a least squares regression. OLS-2 is by using two least squares regressions for the two groups of treatment groups, respectively. KNN is the k-nearest neighbors algorithm. BLR denotes balancing linear regression method. TARnet is a feedforward network that we introduced in chapter 2. BNN stands for balancing neural network. RF is random decision forests are an ensemble learning method regression. CFRW is a counterfactual regression by using Wasserstein distance metric. CEVAE stands for causal effect variational autoencoder method.



Natural Science Faculty of the University of Basel
Department of Mathematics and Computer Science
Biomedical Data Analysis

Title of work:

Estimating Cause-Effects with the Deep Information Bottleneck

Thesis type and date:

Master Thesis, November 15, 2019

Examiner:

Prof. Dr. Volker Roth

Supervision:

Mario Wieser and Sonali Parbhoo

Student:

Name: Yue Xu
E-mail: yue.xu@unibas.ch
Matr.-Nr.: 2015-062-904

Statement regarding plagiarism:

I hereby declare that this submission is my own work and that I have fully acknowledged the assistance received in completing this work and that it contains no material that has not been formally acknowledged. I have mentioned all source materials used and have cited these in accordance with recognised scientific rules.

Basel, November 15, 2019: _____