



VRIJE
UNIVERSITEIT
BRUSSEL



Master thesis submitted in partial fulfilment of the requirements for the degree of
Master of Science In de Ingenieurswetenschappen: Computerwetenschappen

VARIATIONAL GREEDY INFOMAX

Towards indepdnent and interpretable
representations

Fabian Denoodt

2022-2023

Promotor(s): Prof. Dr. Bart de Boer
Science and Bio-Engineering Sciences



VRIJE
UNIVERSITEIT
BRUSSEL



Proefschrift ingediend met het oog op het behalen van de graad van Master of
Science In de Ingenieurswetenschappen: Computerwetenschappen

[DUTCH] VARIATIONAL GREEDY INFOMAX

[Dutch] Towards indepednent and
interpratable representations

Fabian Denoodt

2022-2023

Promotor(s): Prof. Dr. Bart de Boer

Wetenschappen en Bio-ingenieurswetenschappen

Contents

1	Introduction	5
2	Background	7
2.1	Entropy, relative entropy and mutual information	7
2.1.1	Shannon's Entropy	7
2.1.2	Relative entropy and mutual information	8
2.2	Supervised neural networks	8
2.3	Representation learning through reconstruction error	10
2.3.1	Autoencoders	10
2.3.2	Variational autoencoders	11
2.4	Representation learning through Noise-contrastive estimation	14
2.4.1	Contrastive predictive coding	14
2.4.2	Greedy InfoMax	16
3	Variational contrastive predictive coding	19
3.1	Loss function	19
3.2	Learning benefits	20
4	Experiments	21
4.1	Greedy Infomax model	21
4.1.1	Dataset	21
4.1.2	Training and validation during training	24
4.2	Decoders for Greedy Infomax	25
4.3	Feature selection for visualisation	25
4.4	Variational contrastive predictive coding	25
4.5	CHATGPT ANSWERS FOR VAE	26
4.6	Results	27
4.6.1	Results CPC Simple v2 w/ 2 modules	27
4.7	GIM: Activations visualisations	28
4.8	Decoder: predictions on test set	30
5	Related work	35
5.1	Representation learning: explainable	35
5.2	Variational learning	35
5.3	Links I should investigate	35
6	Discussion	37

Chapter 2

Background

2.1 Entropy, relative entropy and mutual information

We first discuss specific definitions from information theory. These concepts will be relevant to understand contrastive predictive coding, which we discuss in a following section. The formal definitions are obtained from the book "Elements of Information Theory" [1]. The equations that contain a log function are assumed to be under base two.

2.1.1 Shannon's Entropy

Entropy measures the average amount of information required to describe a random variable [1]. The entropy $H(X)$ of a discrete random variable X , is formally defined in equation 2.1 shown below.

$$H(X) = - \sum_{x \in \mathcal{X}} p(x) \log p(x) \quad (2.1)$$

The alphabet \mathcal{X} represents the set of events the random variable X can take, formerly known as the *sample space*. Additionally, $p : \mathcal{X} \rightarrow [0, 1]$ denotes the probability density function of X . Hence, given an event $x \in \mathcal{X}$, $p(x)$ corresponds to the probability of event x occurring.

Assume a random variable X with possible events x_1, x_2 . Intuitively, when $p(x_1)$ is low, the surprise when the event x_1 occurs will be high. The surprise for one event is denoted in equation 2.2.

$$-p(x) \log p(x) \quad (2.2)$$

Hence, entropy can also be considered as the sum of "surprise" over each event [2]. To understand why equation 2.2 does indeed correspond to a measure of surprise, consider an event $x \in \mathcal{X}$ with $p(x) = 1$. Note that $\log p(x) = 0$, and thus the surprise is zero. Meanwhile, if $p(x)$ approaches 0, $\log p(x)$ goes to $-\infty$. And hence, by the negation sign in formula 2.1 the surprise is large.

Figure 2.1 displays when entropy reaches its maximum for the case of a random variable with 2 outcomes. We can see that the entropy, and thus, the information is largest when the probability of the two outcomes is equal to each other, namely $p(x_1) = p(x_2) = 0.5$. Note that for a random variable X with more than two events, $H(X)$ can be larger than one.

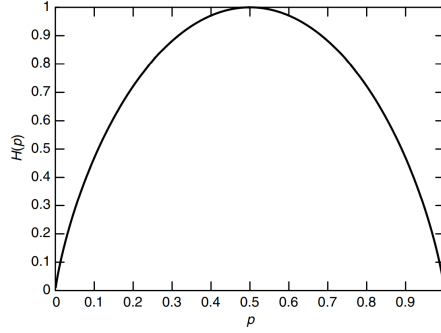


Figure 2.1: $H(p)$ vs p (originates from "Elements of Information Theory", page 16)

2.1.2 Relative entropy and mutual information

Relative entropy, also known as the Kullback Leibler (KL) divergence, is defined in equation 2.3, where p and q denote a probability density function over the same sample space \mathcal{X} [1]. The KL divergence quantifies the "divergence" or "distance" between the two distributions. Note that $D(p||q)$ does not necessarily correspond to $D(q||p)$ and thus the metric is not symmetrical.

$$D_{KL}(q || p) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \quad (2.3)$$

The mutual information (MI) between two random variables X and Y can be computed as the KL divergence between their joint probability distribution, $p_{X,Y}(x,y)$, and the product of their marginal probability distributions, $p_X(x)$ and $p_Y(y)$, which is denoted as $p_X(x)p_Y(y)$ [1]. The equation for mutual information then becomes:

$$I(X;Y) = D_{KL}(p_{X,Y}(x,y) || p_X(x)p_Y(y)) \quad (2.4)$$

As described by Cover and Thomas in their book "Elements of Information Theory" [1], $I(X;Y)$ quantifies the amount of information Y describes about X . A alternative definition for $I(X;Y)$ is illustrated in 2.5. The equation provides us with an intuitive meaning for $I(X;Y)$, corresponding to the surprise caused by X , which is reduced by the knowledge of Y . In a following section, we discuss how these concepts from information theory are applied in representation learning, by maximising the mutual information between latent representations.

$$I(X;Y) = H(X) - H(X|Y) \quad (2.5)$$

2.2 Supervised neural networks

We shall now discuss traditional supervised learning approaches, as these will lay the groundwork for the representation learning approaches discussed in the following section.

Typical supervised machine learning problems consider the problem where given a training set of tuples (x_i, y_i) , a mapping function $f : \mathcal{X} \rightarrow \mathcal{Y}$ must be learned, where $x_i \in \mathcal{X}$ corresponds to the input sample and $y_i \in \mathcal{Y}$ the label. A good f will then also generalise well to unseen $x_i \in \mathcal{X}$, which were not part of the training set.

Artificial Neural Networks (ANNs) in particular, tackle this problem by defining f as a fixed set of parameters consisting of a series of layers. During inference, at each layer l a transformation

matrix W^l is applied to the output vector from the previous layer a^{l-1} . This is shown in the equation below.

$$z^l = W^l a^{l-1}$$

Secondly, a non-linear function $\sigma : R^d \rightarrow R^d$ is applied to z^l , as shown in equation 2.6. The resulting vector a^l may then again be the input for a following layer.

$$a^l = \sigma(z^l) \quad (2.6)$$

Hence, during inference, the input vector x is propagated through each layer, resulting in a final output \hat{y}^L . The equation for the forward pass of a neural network with L layers is described in equation 2.7.

$$f_{W^1 \dots W^L}(x) = \hat{y}^L = \sigma(\dots \sigma(\sigma(x^T W^1)^T) W^2 \dots)^T W^L \quad (2.7)$$

During training the ANN's parameters $W^1 \dots W^L$, are optimised according to the learning problem. The viability of the parameters is quantified by the loss function \mathcal{L} over a batch of n training samples, as shown in equation 2.8. y_i corresponds to the ground truth label of the single data sample x_i , $e(y_i, \hat{y}_i)$ the error between a ground truth y_i and the estimation \hat{y}_i of a single sample. The error function e can be replaced depending on the task, for instance by mean squared error for regression or cross entropy for classification problems.

$$\mathcal{L}(W) = \frac{1}{n} \sum_{i=1}^n e(y_i, f_W(x_i)) \quad (2.8)$$

The parameters can be optimised by minimising the loss function \mathcal{L} defined above. This can be done algebraically, however, it becomes difficult when the dimension of the input features is large. Gradient descent is used to find an "appropriate" minimum of the cost function, by iteratively adjusting the parameters $W^1 \dots W^L$ via the following learning rule.

$$W_{ij}^l \leftarrow W_{ij}^l - \alpha \frac{\partial \mathcal{L}}{\partial W_{ij}^l} \quad (2.9)$$

Although gradient descent can be successfully applied for finding local minima, neural networks may still contain many parameters which each must be optimised. This is typically resolved by applying the backpropagation algorithm in combination with gradient descent, such that fewer partial derivatives must be calculated, resulting in more efficient optimisation.

Although supervised learning though ANNs is considered successful, their performance is heavily dependent on the choice of the data representation [3], on which they are applied. For complex representations, a more complex architecture may be required, which requires more data to prevent overfitting. When labelled data is scarce, these models may result in seemingly well performance on the training set, but bad generalisation to data outside the training set. As a consequence, a lot of time in the machine learning pipeline is invested in transforming data into better latent representations. Good latent representations tend to disentangle the data into meaningful features, such that data can be more easily be separable into classes. In the following section we discuss how part of the labour of finding good representations can be relieved with unsupervised learning approaches, which may learn to find good latent representations. These representations could then be used as the input for supervised predictors.

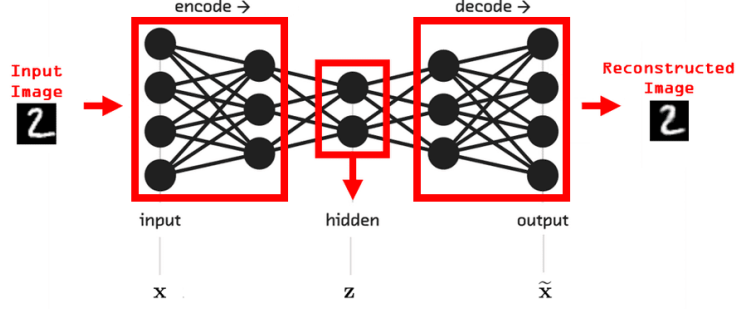


Figure 2.2: Autoencoder neural network architecture adapted from [7].

2.3 Representation learning through reconstruction error

One of the challenges in supervised learning is the constant need of large amounts of labelled data. Hence, when a labelled dataset is small, we would like to leverage a larger unlabelled dataset as basis for learning. By doing so, a mapping can be learned from the raw input data to a representation which makes downstream tasks easier. This process of learning representations from data is commonly referred to as representation learning [4]. Supervised learning algorithms can then learn directly from these disentangled latent representations with fewer labelled data.

In the following two sections we discuss two paradigms of representation learning with ANNs. The first paradigm is learning representations by minimising a reconstruction error. The second learns its representations by contrasting them against noise. These two paradigms will lay the basis for our own contributions in section three.

2.3.1 Autoencoders

Autoencoders were introduced in 1986 by D. Rumelhart et al. [5] as a means to learn compressed representations [6]. This is achieved through a neural network architecture consisting of two blocks. The first block is the encoder E and receives input data which it encodes into a lower dimensional representation. The second block, called the decoder D , receives as input the latent representation and is tasked to reconstruct the original input. The two blocks are simultaneously optimised by *minimising the reconstruction error* shown in the following equation:

$$\mathcal{L} = \sum_{i=1}^N l(\mathbf{x}^{(i)}, D(E(\mathbf{x}^{(i)}))) \quad (2.10)$$

where l refers to the error for a single data point, for instance the l_2 -norm and N the number data points. An example autoencoder architecture is depicted in figure 2.2. $\mathbf{x} \in \mathcal{D}$ corresponds to an input vector which is encoded into latent representation \mathbf{z} . The decoding of \mathbf{z} corresponds to $\tilde{\mathbf{x}} = D(E(\mathbf{x}))$. The dimension of \mathbf{z} is typically bottlenecked to be smaller than the original dimension of \mathbf{x} . This results in the encoder having to define encodings that are as "informative" as possible to reconstruct the original data [6]. \mathbf{z} may then for instance be used for downstream tasks such as classification or clustering.

While capable of learning compressed representations, autoencoders do not pose any restrictions on the latent space they define (the space of latent representations). As a result, the representations may be meaningful to computers, but non-interpretable to humans. For instance, given the left image depicted in figure 2.3 which depicts an autoencoder's two dimensional latent

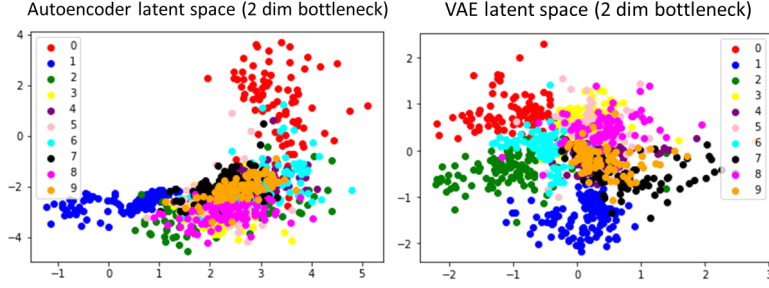


Figure 2.3: Both images represent a the two-dimensional latent space, learned from the MNIST dataset [8]. This is a dataset consisting of images of handwritten numbers between 0 and 9. Each image is associated to a class label, referring to the number. The left image consists of the space learned from a classical autoencoder, the right of a variational autoencoder (VAE). Both autoencoders have not received any explicit information of the labels of the dataset, yet learned to define representations which equal numbers are closer to each other. The VAE’s latent space, depicted in the right image, is optimised to be standard normally distributed. The latent vectors \mathbf{z} are distributed according to the two-dimensional normal distribution $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, where μ is the two dimensional zero vector. \mathbf{I} is the 2×2 covariance matrix with ones on the diagonal and zeroes elsewhere.

space, it is very difficult to know what the resulting images would be when interpolating between the latent representations of 0 (red) and 1 (blue). Answering this question becomes even more infeasible for higher dimensional latent vectors.

2.3.2 Variational autoencoders

Similar to traditional autoencoders, variational autoencoders (VAE) learn representations that contain the important information that is necessary to reconstruct the data, *however*, an additional constraint is applied to the latent space [9, 10, 11, 12]. The data points from the latent space are defined in such a way that they conform to a certain distribution, typically, the standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ [10]. This behaviour can be observed in the right plot of figure 2.3. Since the latent representations conform a two-dimensional standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$, the data samples are more likely to be near the center at (0, 0).

Simulating distributions through neural networks

In variational autoencoders the latent representations of a data point $\mathbf{x}^{(i)}$ does not simply consist of a fixed deterministic vector $\mathbf{z}^{(i)}$, as was the case in the traditional autoencoder. Instead, given $\mathbf{x}^{(i)}$, its latent representation corresponds the following distribution $q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})$. A concrete vector $\mathbf{z}^{(i)}$ can be obtained taking a sample $\mathbf{z}^{(i)} \sim q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})$. As result a single $\mathbf{x}^{(i)}$ will correspond to multiple latent vectors $\mathbf{z}^{(i)}$. The latent representation $q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})$ is modelled as $\mathcal{N}(\mu^{(i)}, \text{diag}(\sigma^{(i)}))$. This means that all the required information to model $q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})$ is a mean vector $\mu^{(i)}$ and a covariance matrix $\text{diag}(\sigma^{(i)})$. Thus, given $\mathbf{x}^{(i)}$, a deterministic neural network can simulate $\text{diag}(\sigma^{(i)})$ by predicting two vectors, namely, $\mu^{(i)}$ and $\sigma^{(i)}$. The latent vectors $\mathbf{z}^{(i)}$ can then be obtained by randomly sampling from the distribution. And thus, multiple $\mathbf{z}^{(i)}$ ’s may correspond to a single $\mathbf{x}^{(i)}$. This method is depicted in figure 2.4. Finally, a sample $\mathbf{z}^{(i)}$ can be obtained as follows:

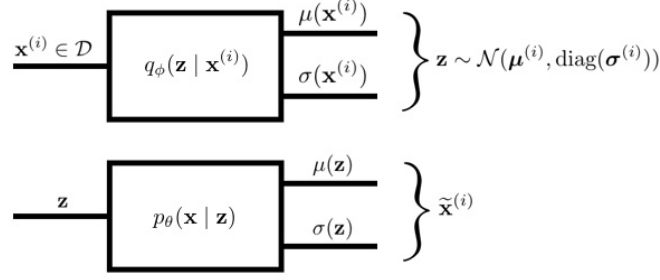


Figure 2.4: High level view of a variational autoencoder, depicting how a data point $\mathbf{x}^{(i)}$ is encoded into a latent distribution and reconstructed as $\tilde{\mathbf{x}}^{(i)}$. Both blocks depict a neural network. The upper block is the encoder and the lower block the decoder. The upper block receives a data points \mathbf{x} and produces the parameters of $q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})$. Since we choose to model $q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})$ as a Gaussian with independent components, the covariance matrix Σ is zero everywhere except for the diagonal. This way the diagonal values, representing the standard deviations, can be represented via a single vector $\sigma^{(i)}$. The vectors $\mu^{(i)}$ and $\sigma^{(i)}$ are $\mu(\mathbf{x}^{(i)})$ and $\sigma(\mathbf{x}^{(i)})$, respectively. These are the output of the encoder block and form the parameters for $q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})$. A single neural network with parameter weights ϕ is used to simulate $q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})$ for every $\mathbf{x}^{(i)} \in \mathcal{D}$. This strategy of sharing ϕ across data points is referred to as "amortised variational inference" [12].

$$\mathbf{z}^{(i)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(i)} \quad (2.11)$$

where $\epsilon^{(i)}$ corresponds to a sampled value $\epsilon^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and \odot is element-wise multiplication. Computing $\mathbf{z}^{(i)}$ through $\epsilon^{(i)}$, rather than directly sampling from $\mathbf{z}^{(i)} \sim \mathcal{N}(\mu^{(i)}, \text{diag}(\sigma^{(i)}))$ is referred to as the parametrisation trick and allows for gradients to freely backpropagate through the layer [10].

The learning objective

So far we have discussed how predictions of a neural network can emulate predicting a distribution $q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})$, by predicting the distribution's parameters $\mu^{(i)}$ and $\sigma^{(i)}$. However, no constraints have been set on the quality of the distributions. We will discuss this now. The representations are optimised to minimise two measurements: one, the reconstruction error, and secondly, the distance from the latent distributions to $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

The loss function to be optimised for a single data point $\mathbf{x}^{(i)}$ is shown in the equation below.

$$\mathcal{L} = \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})} \left[-\log p_\theta(\mathbf{X}^{(i)} | \mathbf{z}) \right] + D_{KL} \left(q_\phi(\mathbf{Z} | \mathbf{x}^{(i)}) \parallel p_\theta(\mathbf{Z}) \right) \quad (2.12)$$

Although the equation may seem daunting at first, we will decompose its components. The loss function is made up of two terms, the left term corresponds to the reconstruction error, while the second term poses constraints on the latent space. Let us focus on the reconstruction term first.

$$\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})} \left[-\log p_\theta(\mathbf{X}^{(i)} | \mathbf{z}) \right]$$

The distribution $p_\theta(\mathbf{X}^{(i)} | \mathbf{z})$ is a distribution over $\mathbf{x}^{(i)}$ where \mathbf{z} is instantiated. What comes out are thus probabilities. Notice \mathbf{z} is sampled from the Gaussian distribution $\mathbf{z} \sim q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})$.

So the \mathbf{z} 's close to the mean are more likely to be sampled than the ones further away. So for these \mathbf{z} 's, we'd like their probability of corresponding to an actual \mathbf{x} to be high as well. Adding a log to this equation does not change the parameters to obtain a maximum. Finally, by adding a negative sign in front, maximising the equation is equivalent to minimising the negative. In practice this term is approximated through mini-batches with the mean squared error.

Optimising the second term of equation 2.12 poses the constraints on the latent distributions $q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})$. Again, this metric should be minimised. As we discussed in the chapter on Entropy, the KL divergence can be considered as a (non-symmetric) distance measure between two distributions. Hence, this value is small when the two distributions are similar. As we discussed earlier $p_\theta(\mathbf{Z})$ is often replaced by $\mathcal{N}(\mathbf{0}, \mathbf{I})$ as shown below. Minimising this equation will result in moving each distribution $q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})$, corresponding to a value $\mathbf{x}^{(i)}$, close to $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

$$D_{KL} \left(q_\phi(\mathbf{Z} | \mathbf{x}^{(i)}) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I}) \right)$$

One can algebraically prove, that optimising this equation is equivalent to optimising the following equation [11]:

$$D_{KL} \left(\mathcal{N}(\boldsymbol{\mu}^{(i)}, \text{diag}(\boldsymbol{\sigma}^{(i)})) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I}) \right) = \frac{1}{2} \sum_{k=1}^D \left(-\log(\sigma_k^{(i)})^2 - 1 + (\sigma_k^{(i)})^2 + (\mu_k^{(i)})^2 \right) \quad (2.13)$$

where $\sigma_k^{(i)}$ and $\mu_k^{(i)}$ correspond to the components of the predicted vectors $\boldsymbol{\sigma}^{(i)}$ and $\boldsymbol{\mu}^{(i)}$, respectively.

[TODO] Relation with variational inference / derivation of ELBO loss

$$\begin{aligned} -\mathcal{L}_{ELBO} &= -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})} \left[\log p(\mathbf{Z}, \mathbf{x}^{(i)}) - \log q(\mathbf{z} | \mathbf{x}^{(i)}) \right] \\ &= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{Z} | \mathbf{x}^{(i)})} \left[-\log p_\theta(\mathbf{X}^{(i)} | \mathbf{z}) \right] + D_{KL} \left(q_\phi(\mathbf{Z} | \mathbf{x}^{(i)}) \parallel p_\theta(\mathbf{Z}) \right) \end{aligned} \quad (2.14)$$

Where the first term is reconstruction error and second is regularisation. Hence minimising the KL divergence between the two posteriors is equivalent to minimising the divergence between the approximate posterior $q(\mathbf{z} | \mathbf{x}^{(i)})$ and the **marginal or prior?** $p_\theta(\mathbf{Z})$.

$p(z)$ in the equation is usually chosen to be the standard normal distribution $\mathcal{N}(0, I)$, such that a closed form solution exists. As such, when $p(z)$ corresponds to the standard normal, and $q(\mathbf{z} | \mathbf{x}^{(i)})$ is a multidimensional Gaussian with mean vector $\boldsymbol{\mu}^{(i)}$ and covariance matrix with independent dimensions, such that the diagonal corresponds of a vector standard deviations $\boldsymbol{\sigma}^{(i)}$, then the KL divergence has the following closed form:

$$D_{KL} \left(\mathcal{N}(\boldsymbol{\mu}^{(i)}, \text{diag}(\boldsymbol{\sigma}^{(i)})) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I}) \right) = \frac{1}{2} \sum_{k=1}^D \left(-\log(\sigma_k^{(i)})^2 - 1 + (\sigma_k^{(i)})^2 + (\mu_k^{(i)})^2 \right) \quad (2.15)$$

For the equation above, gradients can easily be back propagated through machine learning libraries such as Tensor Flow. We still require a method for computing the gradient of the first term in \mathcal{L}_{ELBO} .

2.4 Representation learning through Noise-contrastive estimation

2.4.1 Contrastive predictive coding

In what follows next, we discuss Contrastive Predictive Coding (CPC), a representation learning approach that we use as the basis for our own experiment in the following chapter. CPC is an unsupervised learning approach, again with the objective of learning (lower dimensional) representations from high dimensional data [13]. While the objective is thus the same as for the autoencoders discussed in the previous section, CPC achieves its representations entirely differently. An autoencoder’s objective is to define a compressed representation from which the original data can be recovered. However, when working with sequential data, compressing patches of the sequence without considering the relation with nearby patches, will result in lost information as the context between patches is not encoded into the representation.

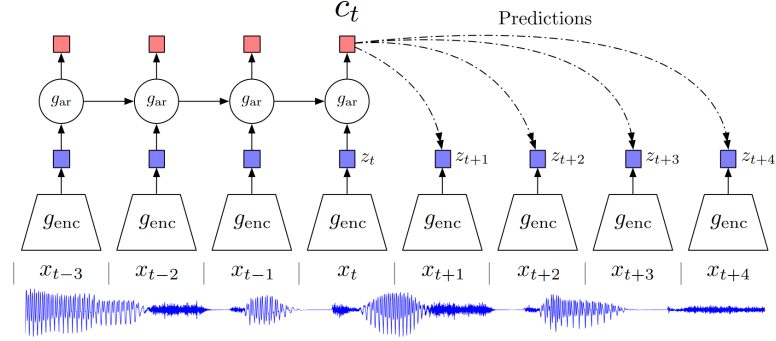


Figure 2.5: Overview of Contrastive Predictive Coding, originates from [13]

CPC deals with these context issues by maximising the shared information between the extracted representations of temporally nearby patches [14]. We will discuss this concept in more detail in a following section. For now, we would like to draw the readers attention to figure 2.5. The figure depicts a high level view of how this idea is achieved by producing two latent representations \mathbf{z}_i and \mathbf{c}_i . An audio sequence of undefined length is split up into patches $\mathbf{x}_1 \dots \mathbf{x}_n$ where each \mathbf{x}_i is a vector of fixed length, containing for instance 10ms of speech audio. Each patch \mathbf{x}_i is encoded into latent representation \mathbf{z}_t , defined as follows:

$$\mathbf{z}_t = g_{enc}(\mathbf{x}_t).$$

$g_{enc}(\cdot)$ is for instance a convolutional fully connected ANN. The latent representations $\mathbf{z}_1 \dots \mathbf{z}_n$ are obtained independently from each other and do not yet contain any information on an historic context. This historic context is achieved through $g_{ar}(\cdot)$, an auto-regressor, which encodes all previous $\mathbf{z}_1 \dots \mathbf{z}_t$ into a single representation \mathbf{c}_t :

$$\mathbf{c}_t = g_{ar}(\mathbf{z}_1 \dots \mathbf{z}_t)$$

Either \mathbf{z}_t or \mathbf{c}_t could be used as latent representation for downstream tasks. Oord et al. suggest to use \mathbf{c}_t for tasks where context about the past is useful, for instance speech recognition, and \mathbf{z}_t when historic context is not useful [13]. As shown in figure 2.5, the encodings from sequential data of undefined length, may correspond a series of latent representations $\mathbf{c}_1, \mathbf{c}_2, \dots$

or $\mathbf{z}_1, \mathbf{z}_2, \dots$. In the case of downstream tasks which require a single representation vector, Oord et al. propose to pool the sequence of vector representations into a single vector.

Slowly varying features

The temporally nearby patches \mathbf{z}_{t+1} and \mathbf{c}_t are optimised to preserve shared information, while discarding differences. Before we discuss how to obtain such representations, we first motivate why defining representations in this fashion makes sense.

Consider we would like to define useful representations for sequential data such as speech signals. Then it not unlikely to believe that the conveyed information at time step t and $t+k$ contains some redundant information, such as pitch, frequency, tone, etc. [15]. Meanwhile, large changes of the signal in a small time window, may be the result of noise. Sequential data which poses these slowly varying features, are commonly referred to as "slow-features" [16]. CPC leverages these slowly varying features, by encoding the underlying shared information between different patches, while at the same time discarding low-level information and noise that is more local [13].

The learning objective

CPC will learn to preserve information between temporally nearby representations, by solving another task. In particular, CPC learns to discriminate subsequent *positive* samples \mathbf{z}_{t+k} from *negative* random samples \mathbf{z}_j . This is achieved through a similarity function $f_k(\cdot)$, which scores the similarity between two latent representations [14]. It is defined as a log bilinear model as follows:

$$f_k(\mathbf{z}_j, \mathbf{c}_t) = \exp(\mathbf{z}_j^T W_k \mathbf{c}_t) \quad (2.16)$$

where W_k is a weight matrix which is learned. $f_k(\mathbf{z}_j, \mathbf{c}_t)$ thus quantifies how likely the context \mathbf{c}_t corresponds to a random vector \mathbf{z}_j . Due to the slowly varying data assumption, a good representation for successive representations \mathbf{z}_{t+k} and \mathbf{c}_t is one where $f_k(\mathbf{z}_{t+1}, \mathbf{c}_{t+1})$ is high and $f_k(\mathbf{z}_j, \mathbf{c}_t)$ is small for random \mathbf{z}_j . Or equivalently, maximising the shared information between temporally nearby patches, while discarding the temporal noise results in large values $f_k(\mathbf{z}_{t+1}, \mathbf{c}_{t+1})$.

The InfoNCE loss, used to optimise g_{enc} , g_{ar} and W_k simultaneously is shown below.

$$\mathcal{L}_N = - \sum_k \mathbb{E}_X \left[\log \frac{f_k(\mathbf{z}_{t+k}, \mathbf{c}_t)}{\sum_{\mathbf{z}_j \in X} f_k(\mathbf{z}_j, \mathbf{c}_t)} \right] \quad (2.17)$$

where X corresponds to the set $\{\mathbf{z}_{t+k}, \mathbf{z}_1, \mathbf{z}_2, \dots\}$. Notice that there exists exactly one $\mathbf{z}_{t+k} \in X$, which corresponds to a positive sample and all other $\mathbf{z}_j \in X$ correspond to negative samples (with respect to the context \mathbf{c}_t). Hence good representations for \mathbf{z}, \mathbf{c} and $f_k(\cdot)$, will result in a large similarity score for positive samples and approximate 0 for negative samples, resulting in a minimum fraction equal to 1. This would then be cancelled out by the $\log(\cdot)$ function. Meanwhile, \mathcal{L}_n is large when the denominator is large, indicating in a large high similarity score for negative samples.

Ties with mutual information

Earlier we argued that CPC's encodings will preserve shared information between temporally nearby patches, while discarding the local noise. Oord et al. make this claim even stronger by making ties with mutual information, which we discussed in a previous chapter. In particular, Oord et al. proof that optimising InfoNCE is equivalent to maximising the mutual information between \mathbf{c}_t and \mathbf{z}_{t+1} [13].

$$I(\mathbf{z}_{t+1}; \mathbf{c}_t) = \sum_{\mathbf{z}_{t+1}, \mathbf{c}_t} p(\mathbf{z}_{t+1}, \mathbf{c}_t) \log \frac{p(\mathbf{z}_{t+1} | \mathbf{c}_{t+1})}{p(\mathbf{z}_{t+1})} \quad (2.18)$$

This proof is available in their appendix. Although, we do not repeat the proof here, we give a high level overview.

The first step in proving the relation between the InfoNCE loss and mutual information is to model $f_k(\mathbf{z}_{t+k}, \mathbf{c}_t)$ in a probabilistic manner. The InfoNCE loss is in fact the categorical cross-entropy of classifying the positive sample correctly with $\frac{f_k}{\sum_x f_k}$ as the predicted model [13]. Since this equation may take values between zero and one, it can be considered as a probability. In particular, the optimal probability for the loss can then be written as

$$p(i | X, \mathbf{c}_t)$$

where X corresponds the set of samples $\{\mathbf{z}_{t+k}, \mathbf{z}_1, \mathbf{z}_2, \dots\}$ as discussed in the InfoNCE loss, and i corresponds to indicator that sample \mathbf{z}_i is the "positive" sample. By doing so, one can eventually obtain a proportionality relation to the density distribution presented below.

$$f_k(\mathbf{z}_{t+k}, \mathbf{c}_t) \propto \frac{p(\mathbf{z}_{t+k} | \mathbf{c}_t)}{p(\mathbf{z}_{t+k})} \quad (2.19)$$

Oord et al. utilise this proportionality relation to reformulate $-\mathcal{L}_N$ as a lower bound on the mutual information between \mathbf{z}_{t+1} and \mathbf{c}_t as follows [14, 13]:

$$I(\mathbf{z}_{t+1}; \mathbf{c}_t) \geq \log(N) - \mathcal{L}_N \quad (2.20)$$

Since number of samples N is a constant, the mutual information between \mathbf{z}_{t+1} and \mathbf{c}_t becomes greater when \mathcal{L}_N becomes smaller. Additionally, when the number of samples N increases, the bound becomes tighter.

TODO: if time permits: compare against VAE, so doesn't have a decoder layer so simpler architecture + allows for sequential data.

2.4.2 Greedy InfoMax

So far we discussed how CPC encodes latent representations by maximising the mutual information between temporally nearby patches of data. This method has shown great success in recent years and is considered state of the art in self-supervised learning for encoding sequential data [17]. Additionally, CPC has been successfully applied to multiple use cases [17, 18, 19, 20, 21, 22]. This is achieved by minimising the InfoNCE loss discussed earlier in equation 2.17. Through this *global* loss function all parameters are optimised end-to-end via backpropagation. Although, backpropagation is considered effective, it still suffers from multiple constraints. Löwe et al. group the constraints in two categories: biological and computational.

Biological constraints of backpropagation [TODO]

- **Local Error Representation, Weight Symmetry, ...** its performance is still uncompanionable to

When comparing its performance against how humans learn, humans can still learn much faster from only a few examples...

The human brain does not appear to have a global objective function, which is optimized by backpropagating an error signal [23]. In the following section we discuss the changes Löwe et al. have made to make CPC more biologically plausible.

Computational constraints of backpropagation [TODO]

- - **memory overhead, synchronous training, vanishing gradients.**

Löwe et al. work further on the representation learning method proposed in CPC by taking a biologically inspired approach. This is done via the introduction of Greedy InfoMax (GIM), an extension on CPC. As in CPC, mutual information between latent representations is maximized by optimising the InfoNCE loss function. However, neurons in the biological brain primarily learn from local information and there does not appear to be a global error function that is optimized [24]. Hence, end-to-end backpropagation cannot be used as an accurate learning model for the brain.

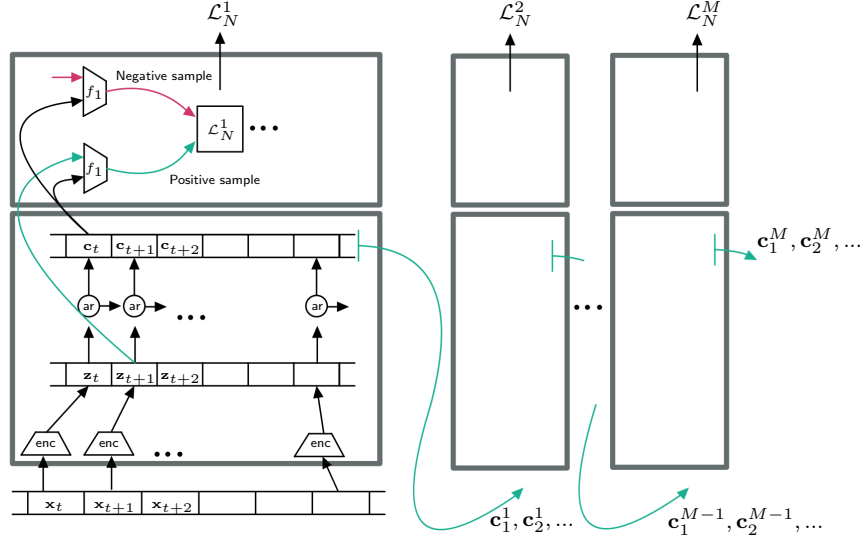


Figure 2.6: Overview of Greedy InfoMax

Towards greedy learning

Instead of optimising a single loss function (as in end-to-end backpropagation), Löwe et al. split the neural network up into "modules". Each module is optimised with its own personal instance of the InfoNCE Loss function \mathcal{L}_N described in equation 2.17. This idea is depicted in figure 2.6, which displays M modules, each trained with their own instance of the InfoNCE loss. Each "module" can be considered as an instance of the CPC framework where high dimensional data goes in and a lower dimensional representation comes out. The output representations of a one module serve as the input for the next module. One can thus think of Greedy InfoMax as "stacking" multiple ANN's, which were each trained with the InfoNCE loss on top of each other. A module may consist of one or more layers of the neural network.

Optimising Greedy InfoMax

By splitting up the neural network into modules and allowing them to optimise their own loss, modules can be trained asynchronously, improving upon training time when multiple GPUs are used. Note that similar to tradition neural networks, during forward propagation each module (or layer) requires the output of the previous layer as its input. Hence, modules are still dependent

on previous modules for training, however, they can learn without the gradients of subsequent modules.

We now revisit the function f_k and NCE Loss, and describe it in the context of GIM. Instead of a single f_k and L_N , we now have a function f_k^m for each module m , each with their own respective loss \mathcal{L}_N^m . The attentive reader may notice $f_k^m(\cdot)$ no longer receives as input c_t , but instead a second z_t . Löwe omits the autoregressive function **TODO: $\mathbf{F}(\mathbf{X}) = \mathbf{CT\ OF\ ZO}$** to obtain c_t and immediately applies gives opted for this function instead, as she did not notice any improvement in her preliminary results.

$$f_k^m(\mathbf{z}_{t+k}^m, \mathbf{z}_t^m) = \exp(\mathbf{z}_{t+k}^{m\top} W_k^m \mathbf{z}_t^m)$$

$$\mathcal{L}_N^m = - \sum_k \mathbb{E}_X \left[\log \frac{f_k^m(\mathbf{z}_{t+k}^m, \mathbf{z}_t^m)}{\sum_{\mathbf{z}_j^m \in X} f_k^m(\mathbf{z}_j^m, \mathbf{z}_t^m)} \right]$$

Chapter 6

Discussion

while GIM argues to resolve memory constraints, not entirely true. In fact we even countered the opposite as containing multiple neural networks, each with their own personal loss function (the loss function is based on `fk` which contains parameters that must be learned), and thus for early layers where the sequence is still long, a lot of memory is required. We went for a compromise on GIM by splitting up the architecture in merely two modules, significantly reducing the memory constraints.

- Explainability of latents is dependent on the performance of the decoder.
- Intermediate loss function with `kld` resulted in similar behaviour as batch normalisation. Resulting in faster convergence than without `kld`.
- We observed no quality loss in the learned representations. Data was equally easily separable.

xx does what [14] yes! idk [1] cool

Bibliography

- [1] T. M. Cover and J. A. Thomas, *ELEMENTS OF INFORMATION THEORY*.
- [2] “Ali Ghodsi, Lec : Deep Learning, Variational Autoencoder, Oct 12 2017 [Lect 6.2].” [Online]. Available: <https://www.youtube.com/watch?v=uaaqyVS9-rM>
- [3] Y. Bengio, A. Courville, and P. Vincent, “Representation Learning: A Review and New Perspectives,” vol. 35, pp. 1798–1828.
- [4] P. H. Le-Khac, G. Healy, and A. F. Smeaton, “Contrastive Representation Learning: A Framework and Review,” vol. 8, pp. 193 907–193 934.
- [5] D. Rumelhart, G. Hinton, and R. Williams, “Learning Internal Representations by Error Propagation,” in *Readings in Cognitive Science*. Elsevier, pp. 399–421. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/B9781483214467500352>
- [6] D. Bank, N. Koenigstein, and R. Giryes. Autoencoders. [Online]. Available: <http://arxiv.org/abs/2003.05991>
- [7] S. Karagiannakos. How to Generate Images using Autoencoders. AI Summer. [Online]. Available: <https://theaisummer.com/Autoencoder/>
- [8] Papers with Code - MNIST Dataset. [Online]. Available: <https://paperswithcode.com/dataset/mnist>
- [9] C. Doersch. Tutorial on Variational Autoencoders. [Online]. Available: <http://arxiv.org/abs/1606.05908>
- [10] David Foster, “3. Variational Autoencoders,” in *Generative Deep Learning, 2nd Edition*, 2nd ed. [Online]. Available: <https://www.oreilly.com/library/view/generative-deep-learning/9781098134174/>
- [11] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. [Online]. Available: <http://arxiv.org/abs/1312.6114>
- [12] —, “An Introduction to Variational Autoencoders,” vol. 12, no. 4, pp. 307–392. [Online]. Available: <http://arxiv.org/abs/1906.02691>
- [13] A. van den Oord, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. [Online]. Available: <http://arxiv.org/abs/1807.03748>
- [14] S. Löwe, P. O’Connor, and B. S. Veeling. Putting An End to End-to-End: Gradient-Isolated Learning of Representations. [Online]. Available: <http://arxiv.org/abs/1905.11786>

- [15] S. S. Rao. Understanding the Gradient-Isolated Learning of Representations and intuition to the Greedy... Analytics Vidhya. [Online]. Available: <https://medium.com/analytics-vidhya/understanding-the-gradient-isolated-learning-of-representations-and-intuition-to-the-greedy-cb6c3598e317>
- [16] Z. Zhang and D. Tao, "Slow Feature Analysis for Human Action Recognition," vol. 34, no. 3, pp. 436–450.
- [17] K. Stacke, C. Lundström, J. Unger, and G. Eilertsen, "Evaluation of Contrastive Predictive Coding for Histopathology Applications," in *Proceedings of the Machine Learning for Health NeurIPS Workshop*. PMLR, pp. 328–340. [Online]. Available: <https://proceedings.mlr.press/v136/stacke20a.html>
- [18] P. de Haan and S. Löwe. Contrastive Predictive Coding for Anomaly Detection. [Online]. Available: <http://arxiv.org/abs/2107.07820>
- [19] M. Y. Lu, R. J. Chen, J. Wang, D. Dillon, and F. Mahmood. Semi-Supervised Histology Classification using Deep Multiple Instance Learning and Contrastive Predictive Coding. [Online]. Available: <http://arxiv.org/abs/1910.10825>
- [20] S. Bhati, J. Villalba, P. Želasko, L. Moro-Velazquez, and N. Dehak. Segmental Contrastive Predictive Coding for Unsupervised Word Segmentation. [Online]. Available: <http://arxiv.org/abs/2106.02170>
- [21] S. Deldari, D. V. Smith, H. Xue, and F. D. Salim, "Time Series Change Point Detection with Self-Supervised Contrastive Predictive Coding," in *Proceedings of the Web Conference 2021*, ser. WWW '21. Association for Computing Machinery, pp. 3124–3135. [Online]. Available: <https://dl.acm.org/doi/10.1145/3442381.3449903>
- [22] O. Henaff, "Data-Efficient Image Recognition with Contrastive Predictive Coding," in *Proceedings of the 37th International Conference on Machine Learning*. PMLR, pp. 4182–4192. [Online]. Available: <https://proceedings.mlr.press/v119/henaff20a.html>
- [23] A. H. Marblestone, G. Wayne, and K. P. Kording, "Toward an Integration of Deep Learning and Neuroscience," vol. 10. [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fncom.2016.00094>
- [24] N. Caporale and Y. Dan, "Spike timing-dependent plasticity: A Hebbian learning rule," vol. 31, pp. 25–46.