Separation of Foreground and Background Signal in Variational Autoencoders

Albert-Ludwigs-Universität Freiburg

Florian Eichin

02.12.1995

Abstract

Abstract goes here

Dedication

To mum and dad

Declaration

I declare that..

Acknowledgements

I want to thank...

Contents

Chapter 1

Introduction

Chapter 2

For this purpose, many ideas of the following chapter are taken from (XX) and the notation mainly follows the same logic.

2.1 Inference Problem Setup

Before we dive into the technical questions of this thesis, we want to begin with a discussion of the problem we attempt to solve formally. Let $\mathbf{x}, \mathbf{z}$ be random variables with $\mathbf{x}$ observable and $\mathbf{z} \in \mathbb{R}^k$ hidden. Then we are interested in the *latent variable model* with model parameters θ^*

$$p_{\theta^*}(\mathbf{x}, \mathbf{z}) = p_{\theta^*}(\mathbf{x}|\mathbf{z})p_{\theta^*}(\mathbf{z}) \tag{2.1}$$

We further assume that prior $p_{\theta^*}(\mathbf{z})$ and $p_{\theta^*}(\mathbf{x}|\mathbf{z})$ are from parametric families of distributions $p_\theta(\mathbf{z})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ and that they have probability density functions that are differantiable with respect to θ and $\mathbf{z}$ almost everywhere.

To make things clearer, we can look at it in a more practical way: Let $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ be a dataset with N i.i.d. samples of our random variable $\mathbf{x}$. Note, that $\mathbf{x}$ can be a vector of arbitrary dimension encoding all kinds of data such as images, soundwaves etc. If we model our data with the above latent variable model, we suppose the datapoints to be generated with the involvement of $\mathbf{z}$ in the sense, that first a value $\mathbf{z}^{(i)}$ is generated from prior distribution $p_{\theta^*}(\mathbf{z})$ and in the second step, $\mathbf{x}^{(i)}$ is generated from $p_{\theta^*}(\mathbf{x}|\mathbf{z}^{(\mathbf{i})})$.

Usually, $\mathbf{z}$ is assumed to have a much lower dimension and a much simpler distribution than $\mathbf{x}$. Therefore, the $\mathbf{z}$-space can be viewed as a space of encodings, where only relevant information for decoding datapoints into the high-dimensional $\mathbf{x}$-space is retained. This is interesting for us, as we're not only interested in approximations of the posterior inference of $\mathbf{z}$ given a value of $\mathbf{x}$ but also the ...

For a given dataset, there is different approaches for the above scenario. However, we do make additional assumptions, that narrow the list of efficient algorithms significantly [XX]:

1 *Intractability*: the integral of the marginal likelihood $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})d\mathbf{z}$, as well as posterior distribution $p_\theta(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})/p_\theta(\mathbf{x})$ are intractable.

2 *Big dataset*: Batch optimization is too expensive and parameter updates on small minibatches preferable. Sampling-based solutions would be too inefficient [XX].

2.2 Variational Inference

In many probabilistic models inference is intractable and approximation methods are needed. One way of approximating solutions to inference problems is to describe it as an optimization problem. Algorithms involving this approach are called *variational*. Given an intractable probability distribution p and a set of tractable distributions $\mathcal{Q}$, the goal of a variational algorithm is to find $q \in \mathcal{Q}$ that is most 'similar' to p. Subsequently, we can use q instead of p find approximate solutions to inference problems efficiently.
Of course, this rather informal description on variational techniques leaves us with questions. What is the similarity of two distributions q and p? How do we choose an according optimization objective $J(q)$? What are good ways of formulating tractable class of distributions $\mathcal{Q}$ and how can we efficiently solve our optimization problem with respect to $J(q)$?
The (partial) answering to these four questions will be the main motivation for the following sections in order to lay the groundwork for the introduction of the Variational Autoencoder (VAE), a probabilistic model designed for learning latent representations with the help of Deep Neuronal Networks (DNNs).

2.3 Kullback-Leibler divergence

Continuing with our first question, there is a way of quantifying the 'similarity' of two distributions p and q in information theory known as the

Kullback-Leibler (KL) divergence. For p, q continuous, the KL divergence is defined as

$$KL(q||p) = \int_{-\infty}^{\infty} q(x) \log \frac{q(x)}{p(x)} \tag{2.2}$$

In the discrete case, it is analogously

$$KL(q||p) = \sum_{x} q(x) \log \frac{q(x)}{p(x)} \tag{2.3}$$

Note, that for any q, p (continuous or discrete) we can deduce the following properties:

- $KL(q||p) \geq 0$
- $KL(q||p) = 0$ if and only if $q = p$

For a proof consider .

Before we dive deeper into how to utilize the KL divergence for our problem, let us gain a better understanding of why it is a sound choice for measuring 'similarity'.

2.4 Variational Lower Bound (ELBO)

As we discussed previously, $p(\mathbf{x})$ as well as$p(\mathbf{z}|\mathbf{x})$ are supposed to be intractable. We have thus no way of retrieving either of the two out of the other. This is where the variational component from two sections before comes into play. In order to approximate $p_\theta(\mathbf{z}|\mathbf{x})$ we introduce a tractable *parametric inference model* $q_\phi(\mathbf{z}|\mathbf{x})$ We will optimize the so called *variational parameters* ϕ of this model such that $p_\theta(\mathbf{z}|\mathbf{x}) \approx q_\phi(\mathbf{z}|\mathbf{x})$. Derived from Bayes' rule, we also have

$$p_\theta(\mathbf{x}) = \frac{p_\theta(\mathbf{z}|\mathbf{x}) p_\theta(\mathbf{z})}{p_\theta(\mathbf{x}|\mathbf{z})} \approx \frac{p_\theta(\mathbf{z}|\mathbf{x}) p_\theta(\mathbf{z})}{q_\phi(\mathbf{x}|\mathbf{z})} \tag{2.4}$$

It is clear, that for our model to fit the true distribution of our data well, we are interested in the following two things:

1. Maximization of the marginal likelihood $p_\theta(\mathbf{x})$ for our data to improve our generative model.

2. Minimization of the KL divergence between $p_\theta(\mathbf{x})$ and $q_\phi(\mathbf{x})$ to improve the approximation of $q_\phi(\mathbf{x})$.

Since the logarithm to base 2 (here abbreviated as log) is monotonous, maximizing $p_\theta(\mathbf{x})$ is equivalent to maximizing $\log p_\theta(\mathbf{x})$. For an arbitrary choice of $q_\phi(\mathbf{z}|\mathbf{x})$ we can consider the following derivation:

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right)\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right)\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \left(\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right)\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right)\right] + KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))
\end{aligned} \tag{2.5}$$

Where the right term in the last row is the KL divergence of $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x},\mathbf{z})$. If we rearrange the equation, we have the following:

$$\log p_\theta(\mathbf{x}) - KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{x},\mathbf{z})) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right)\right] \tag{2.6}$$

And since $KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{x},\mathbf{z})) \geq 0$, the right hand side is a lower bound for $\log p_\theta(\mathbf{x})$. It is also referred to as *variational lower bound* or *evidence lower bound* (ELBO)

$$\begin{aligned}
\mathcal{L}_{\theta,\phi}(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right)\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right]
\end{aligned} \tag{2.7}$$

With the above derivation in mind, we can identify another interpretation of $KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{x},\mathbf{z}))$ besides being the KL divergence of approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and true posterior $p_\theta(\mathbf{x},\mathbf{z})$): It is also the gap between ELBO $\mathcal{L}_{\theta,\phi}(\mathbf{x})$ and $\log p_\theta(\mathbf{x})$. If $q_\phi(\mathbf{z}|\mathbf{x})$ approximates the true $p_\theta(\mathbf{z}|\mathbf{x})$ 'better', the gap gets smaller.
Let $\mathbf{X}$ be the dataset of i.i.d. samples from before and $N_\mathbf{X} = |\mathbf{X}|$. If we want

to fit our model on $\mathbf{X}$, the ELBO yields us an optimization objective we were asking for, namely the average of ELBOs of single datapoints $\mathbf{x} \in \mathbf{X}$:

$$\mathcal{L}_{\theta,\phi}(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}} \frac{\mathcal{L}_{\theta,\phi}(\mathbf{x})}{N_{\mathbf{X}}} \tag{2.8}$$

If we maximize $\mathcal{L}_{\theta,\phi}(\mathbf{x})$ with respect to parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ for our data, we will approximately maximize $p_\theta(\mathbf{x})$ and minimize $KL(p_\theta(\mathbf{x})||q_\phi(\mathbf{x}))$ just like the goals we formulated in the beginning of this section.

2.5 Auto-encoding Variational Bayes

2.5.1 Batch Gradient Descent

With the means of the ELBO, we now have an objective to optimize the model parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ for. A naive solution, also known as *Batch Gradient Descent*, is to initialize the parameters randomly and then to estimate the gradients $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{X})$ and $\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{X})$ and adjust $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ into their respective directions until convergence. With each step of adjusting the parameters for the gradients, also called an *epoch*, we expect $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{X})$ to improve until we have reached a local maximum.

Algorithm 1: Batch Gradient Descent

while *not converged* **do**
  Estimate gradients $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{M})$, $\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{M})$
  Update parameters $\boldsymbol{\theta} \to \boldsymbol{\theta} + \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})$, $\boldsymbol{\phi} \to \boldsymbol{\phi} + \eta\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})$
end

Note, how η is a hyperparameter, that determines the amount of influence the gradients have on the parameters when updating. It is therefore also called the *learning rate*. It is an important to choose η properly as it highly affects the convergence of the algorithm: If we choose η too small, we will only move slowly in the direction of local maxima. However, if we choose it to big, our steps get too big and the algorithm can not converge.

2.5.2 Estimation of the gradients

Because in general analytical computations of the gradients of the ELBO are intractable, we have to estimate them. For $\nabla_{\boldsymbol{\alpha}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{X})$ with $\boldsymbol{\alpha} = \boldsymbol{\theta}$ or $\boldsymbol{\phi}$, it is

sufficient to estimate $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})$ for $\mathbf{x} \in \mathbf{X}$ because since (XX) we have

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{X}) = \nabla_{\boldsymbol{\alpha}} \sum_{\mathbf{x} \in \mathbf{X}} \frac{\mathcal{L}_{\theta,\phi}(\mathbf{x})}{N_{\mathbf{X}}} = \frac{1}{N_{\mathbf{X}}} \sum_{\mathbf{x} \in \mathbf{X}} \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\theta,\phi}(\mathbf{x}) \tag{2.9}$$

For gradients of the ELBO with respect to generative model parameters $\boldsymbol{\theta}$, it is simple to obtain an unbiased Monte Carlo estimator:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\nabla_{\boldsymbol{\theta}} (\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})) \right] \\ &\simeq \nabla_{\boldsymbol{\theta}} (\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})) \\ &= \nabla_{\boldsymbol{\theta}} \log p_\theta(\mathbf{x}, \mathbf{z}) \end{aligned} \tag{2.10}$$

With the $\mathbf{z}$ in the last two lines being a random sample from $q_\phi(\mathbf{z}|\mathbf{x})$. For unbiased gradients with respect to $\boldsymbol{\phi}$, things are a bit more difficult. This is due to the fact, that in general, we have:

$$\begin{aligned} \nabla_{\boldsymbol{\phi}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right] \\ &\neq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\nabla_{\boldsymbol{\phi}} (\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) \right] \end{aligned} \tag{2.11}$$

For $\mathbf{z}$ continuous and $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ differentiable (the case we're focussing on), we can reparameterize $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ to circumvent this problem. We can choose f as some differentiable transformation and introduce another random variable $\boldsymbol{\epsilon}$ independent of $\boldsymbol{\phi}$ and $\mathbf{x}$ such that

$$\mathbf{z} = f(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x}) \tag{2.12}$$

This is also referred to as *reparameterization trick*. With (2.12) we can rewrite our gradient as follows:

$$\begin{aligned} \nabla_{\boldsymbol{\phi}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right] \\ &= \nabla_{\boldsymbol{\phi}} \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[\log p_\theta(\mathbf{x}, f(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})) - \log q_\phi(f(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})|\mathbf{x}) \right] \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[\nabla_{\boldsymbol{\phi}} (\log p_\theta(\mathbf{x}, f(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})) - \log q_\phi(f(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})|\mathbf{x})) \right] \\ &\simeq \nabla_{\boldsymbol{\phi}} (\log p_\theta(\mathbf{x}, f(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})) - \log q_\phi(f(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})|\mathbf{x})) \end{aligned} \tag{2.13}$$

And the last row is again the Monte Carlo estimator for a single sample $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. It is unbiased, because

$$\begin{aligned} &\mathbb{E}_{p(\boldsymbol{\epsilon})} \left[\nabla_{\boldsymbol{\phi},\boldsymbol{\theta}} \log p_\theta(\mathbf{x}, f(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})) - \log q_\phi(f(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})|\mathbf{x}) \right] \\ =&\nabla_{\boldsymbol{\phi},\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[\log p_\theta(\mathbf{x}, f(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})) - \log q_\phi(f(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})|\mathbf{x}) \right] \end{aligned} \tag{2.14}$$

2.5.3 Minibatch Stochastic Gradient Descent

Vanilla Gradient Descent has several shortcomings, that we still have to address. For one, it is usually very expensive to estimate $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{X})$ on big datasets $\mathbf{X}$, violating our goal for efficiency with large data. Also, the algorithm usually only finds the closest local maximum and has no means to escape before convergence. There is several solutions to those problems. One that tackles both is called *Minibatch Stochastic Gradient Descent (SGD)*.

Algorithm 2: Minibatch Stochastic Gradient Descent

while *not converged* **do**
1. Randomly draw a minibatch $\mathbf{M} \in \mathbf{X}$;
2. Estimate gradients $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{M})$, $\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{M})$
3. Update parameters $\boldsymbol{\theta} \to \boldsymbol{\theta} + \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{M})$, $\boldsymbol{\phi} \to \boldsymbol{\phi} + \eta\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{M})$

end

The idea behind Minibatch SGD is again intuitive: Instead of estimating the gradients on the whole dataset $\mathbf{X}$, we randomly draw a subset $\mathbf{M} \subset \mathbf{X}$ of size $N_{\mathbf{M}}$. Since for $\boldsymbol{\alpha} = \boldsymbol{\theta}$ or $\boldsymbol{\phi}$ we have:

$$\nabla_{\boldsymbol{\alpha}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{X}) = \frac{1}{N_{\mathbf{X}}}\sum_{\mathbf{x}\in\mathbf{X}}\nabla_{\boldsymbol{\alpha}}\mathcal{L}_{\theta,\phi}(\mathbf{x}) \simeq \frac{1}{N_{\mathbf{M}}}\sum_{\mathbf{x}\in\mathbf{M}}\nabla_{\boldsymbol{\alpha}}\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \nabla_{\boldsymbol{\alpha}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{M}) \tag{2.15}$$

We can thus get the following unbiased gradient estimators from combining (2.10) with (2.12) and (2.13) for $\mathbf{M}' = \{(\mathbf{x}, \boldsymbol{\epsilon})|\mathbf{x} \in \mathbf{M}, \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})\}$ a set of tuples of each datapoint in $\mathbf{M}$ and an according sample of $\boldsymbol{\epsilon}$:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{M}) &\simeq \frac{1}{N_{\mathbf{M}}}\sum_{(\mathbf{x}^{(i)},\boldsymbol{\epsilon}^{(i)})\in\mathbf{M}'}\nabla_{\boldsymbol{\theta}}\log p_\theta(\mathbf{x}, f(\boldsymbol{\epsilon}^{(i)}, \boldsymbol{\phi}, \mathbf{x})) \\
\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{M}) &\simeq \frac{1}{N_{\mathbf{M}}}\sum_{(\mathbf{x}^{(i)},\boldsymbol{\epsilon}^{(i)})\in\mathbf{M}'}\nabla_{\boldsymbol{\phi}}(\log p_\theta(\mathbf{x}^{(i)}, f(\boldsymbol{\epsilon}^{(i)}, \boldsymbol{\phi}, \mathbf{x})) - \log q_\phi(f(\boldsymbol{\epsilon}^{(i)}, \boldsymbol{\phi}, \mathbf{x}^{(i)})|\mathbf{x}^{(i)}))
\end{aligned} \tag{2.16}$$

Another problem is, that even though Minibatch SGD might converge in local maxima, the parameters it yields might be far from optimal. There is several techniques to overcome this problem and escape local maxima, but the discussion of those is not in the scope of this thesis. Consider [XX] for an in depth look.

Minibatch SGD is a rather straightforward algorithm, with only two hyperparameter $N_{\mathbf{M}} = |\mathbf{M}|$, the size of a minibatch $\mathbf{M} \subset \mathbf{X}$ of our dataset and learning rate η. For $N_{\mathbf{M}} = 1$ it is just called *Stochastic Gradient Descent*.

2.5.4 Auto-Encoding Variational Bayes

2.6 The Variational Auto-encoder (VAE)

2.6.1 Neuronal Networks

one way of parameterizing distributions $q(z|x)$
how do they work?
how to calc gradients? -¿ diffbare fkt

In the context of Variational Autoencoders, Neuronal Networks, which shall be discussed in the following, are the most promising way of realizing said

2.6.2 Neuronal Networks for Parameterizing blbla

choice of posterior q as NN vae algorithm

Broadly speaking, Variational Autoencoders (VAE) are an instance of the AEVB algorithm described in the previous section.

2.7 Convolutional Neural Networks (CNN)

While standard, Fully Connected Neuronal networks offer a great way to realize the encoder and decoder of our VAE, there are approaches that work better on certain kinds of data. Convolutional Neural Networks (CNN) are a NN structure, that offers good results when applied on high-dimensional image data. The main idea with CNNs is, that images consists of recurring combinations of shapes that can be learned by lower dimensional filters.

Appendix A

Appendix Title