Generative and Self-Supervised ML

Latent Variables Models

Javier Béjar - UPC

16/1/2026

BSC AI Factory Partners

Affiliated entities

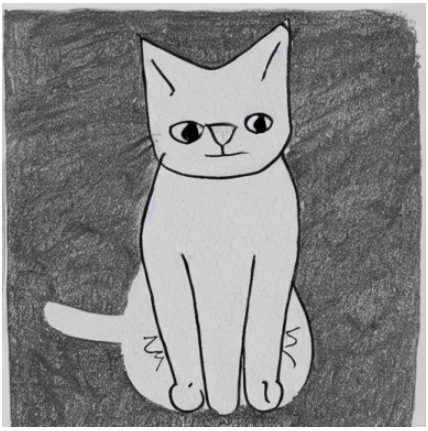
Introduction

Latent Variables Models

- For autoregressive and flow models we observe directly all the variables
- Latent Variables Models assume that exists a set of hidden variables that generate the observed variables
- These hidden variables possibly are less than the observable ones (compression) and are more interpretable, holding some semantic information (hidden representation)
- LVMs try to identify that representation from the observed data

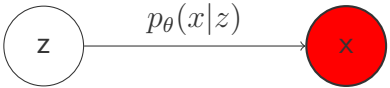
Latent Variables Models

Cat in Pixel Space



Cat in Latent space

- LVMs define the relationship among latent and observed variables as probabilistic models



- Sampling: $z \sim p_Z(z)$, $x \sim p_\theta(x|z)$
- Evaluate Likelihood for samples, given that $p(x, z) = p(x|z)p(z)$, marginalizing by z

$$p_\theta(x) = \sum_z p_Z(z) p_\theta(x|z)$$

- Train

$$\max_\theta \sum_i \log p_\theta(x_i) = \max_\theta \sum_i \log \sum_z p_Z(z) p_\theta(x_i|z)$$

Variational Inference

- In order to compute this model we need to obtain the probabilities that allow finding the latent variables, basically:

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

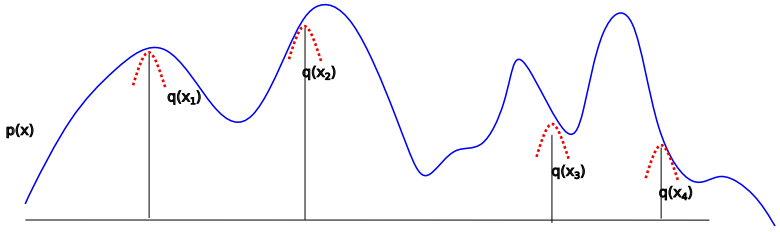
- We need to compute $p(x)$, if z is a vector of continuous values this means to integrate

$$\int_z p(x|z)p(z)dz$$

- $p(z)$ is a multidimensional distribution that needs multiple integration that is usually intractable
- An approach to approximate $p(z|x)$ is Variational Inference

Distance between PDFs

- Variational inference substitutes an intractable probability distribution p by another well-behaved distribution q
- The goal is to transform the distribution q so it is as close as possible to p at least locally for a sample, tuning its parameters ϕ
- Basically we need a distribution q_ϕ that is close to p_θ that we can use as surrogate distribution



- We can measure the closeness between two probability distributions using the Kullback-Leibler Divergence (KL)

$$KL(q||p) = -\sum_x q(x) \log \frac{p(x)}{q(x)} = \sum_x q(x) \log \frac{q(x)}{p(x)}$$

- This is not a symmetric measure, so it is not a proper distance, but can help to develop a loss function for this problem

- We are going to choose a distribution $q_\phi(z|x)$ as a substitute for $p_\theta(z|x)$
- In order to choose the parameters of q_ϕ we need to minimize the KL Divergence

$$KL(q_\phi(z|x)||p_\theta(z|x)) = \sum_{z \sim q_\phi(z|x)} q_\phi(z|x) \log \frac{q_\phi(z|x)}{p_\theta(z|x)}$$

- With this we will have an optimization objective

$$
\begin{aligned}
\sum_{z\sim q_\phi(z|x)} q_\phi(z|x)\log\frac{q_\phi(z|x)}{p_\theta(z|x)} &= \sum_{z\sim q_\phi(z|x)} q_\phi(z|x)\log\frac{q_\phi(z|x)}{p_\theta(x,z)/p_\theta(x)} \quad (product\ rule)\\
&= \sum_{z\sim q_\phi(z|x)} q_\phi(z|x)\left[\log\frac{q_\phi(z|x)}{p_\theta(x,z)}p_\theta(x)\right]\\
&= \sum_{z\sim q_\phi(z|x)} q_\phi(z|x)\left[\log\frac{q_\phi(z|x)}{p_\theta(x,z)}+\log p_\theta(x)\right]\\
&= \sum_{z\sim q_\phi(z|x)} q_\phi(z|x)\log\frac{q_\phi(z|x)}{p_\theta(x,z)} + \sum_{z\sim q_\phi(z|x)} q_\phi(z|x)\log p_\theta(x)\\
&= \sum_{z\sim q_\phi(z|x)} q_\phi(z|x)\log\frac{q_\phi(z|x)}{p_\theta(x,z)} + \log p_\theta(x)\sum_{z\sim q_\phi(z|x)} q_\phi(z|x)\\
&= \sum_{z\sim q_\phi(z|x)} q_\phi(z|x)\log\frac{q_\phi(z|x)}{p_\theta(x,z)} + \log p_\theta(x)
\end{aligned}
$$

- So we have that

$$KL(q_\phi(z|x)||p_\theta(z|x)) = \sum_{z\sim q_\phi(z|x)} q_\phi(z|x) \log \frac{q_\phi(z|x)}{p_\theta(x,z)} + \log p_\theta(x)$$

- That can be rewritten as:

$$\begin{aligned} \log p_\theta(x) &= KL(q_\phi(z|x)||p_\theta(z|x)) - \sum_{z\sim q_\phi(z|x)} q_\phi(z|x) \log \frac{q_\phi(z|x)}{p_\theta(x,z)} \\ &= KL(q_\phi(z|x)||p_\theta(z|x)) + \sum_{z\sim q_\phi(z|x)} q_\phi(z|x) \log \frac{p_\theta(x,z)}{q_\phi(z|x)} \end{aligned}$$

- $\log p_\theta(x)$ is the likelihood of the distribution of the samples

Variational Lower Bound (VLB)/Evidence Lower Bound (ELBO)

- The second term is called the Variational Lower Bound (VLB) or Evidence Lower Bound (ELBO)
- The KL divergence is always larger or equal to 0, so the ELBO is a lower bound of the probability distribution $p(x)$
- if $KL(q_\phi(z|x)||p_\theta(z|x)) = 0$ then the second term is exactly the likelihood of $p(x)$
- The KL divergence is defining actually two quantities
  - How close are the two distributions
  - What is the tightness between the ELBO and the likelihood of the marginal distribution

- The variational can be defined as expectations, and given that we are going to work with samples from the distributions we can use gradient descent for optimization

$$\mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right] + \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right]$$

- Maximizing the ELBO we maximize the likelihood of the marginal $p(x)$ and minimize the divergence among the two conditional distributions

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right]$$

- To train the parameters ϕ and θ of the distributions from data we will have

$$\mathcal{L}_{\theta,\phi}(\mathcal{D}) = \sum_{x \in \mathcal{D}} \mathcal{L}_{\theta,\phi}(x)$$

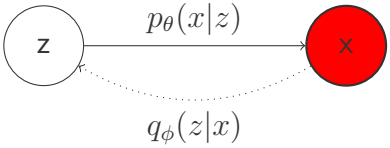
- For a datapoint x we have:

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

- We can perform alternate gradient descent for this functions computing its derivative respect each group of parameters fixing one set at a time

Variational AutoEncoders

- The goal of all these estimations is to have probability distributions that can map from the original data space to the latent space and back

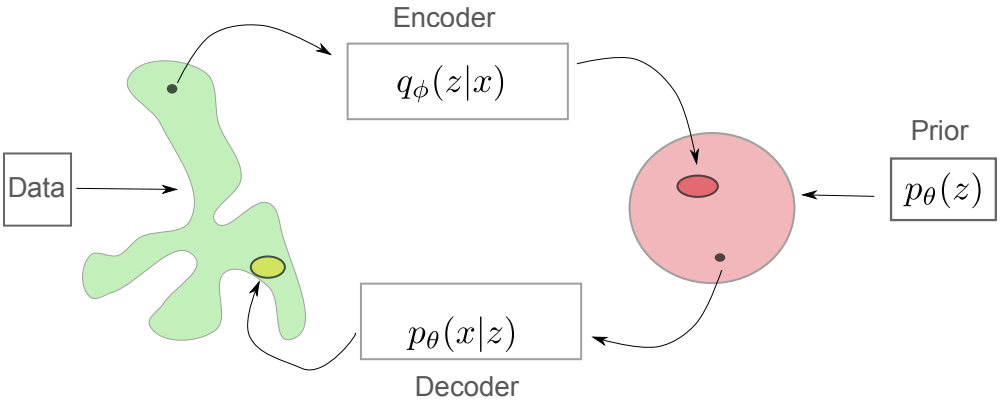


- This can be implemented as a type of autoencoder: Variational AutoEncoder (VAE)

Variational Autoencoder (Google)

- A Variational AutoEncoder is composed by two networks
- The encoder learns the function $q_\phi(z|x)$ obtaining an estimate for its parameters, this is also called the inference model or recognition network
  - From the parameters estimate of the encoder random samples can be drawn
- The decoder learns the function $p_\theta(x|z)$ that corresponds to a generative model
  - The samples generated from the result from the encoder are passed through the decoder
- After training we keep the decoder, we can obtain new samples processing samples from a prior distribution $p_\theta(z)$

Variational Autoencoder: Mapping



- In order to implement the VAE we need to pick a distribution for q that is easy to work with
- For continuous variables an evident choice is to use the gaussian distribution

$$q_\phi(z|x) = \mathcal{N}(z, \mu, \Sigma)$$

- To make things simpler we can use a gaussian with diagonal covariance (all latent variables are independent)

$$q_\phi(z|x) = \mathcal{N}(z, \mu, diag(\sigma)) = \prod_i q_\phi(z_i|x) = \prod_i \mathcal{N}(z_i, \mu, \sigma_i)$$

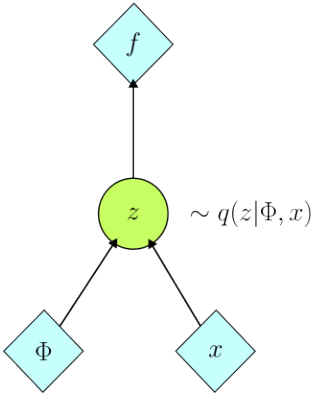
- So the encoder will estimate μ and σ (actually $\log(\sigma)$) and a sample will be obtained from the distribution to run the decoder

- Given that the sampling is a probabilistic operation, we can not use it with backpropagation
- In order to avoid this problem we can assume that we have a source of gaussian noise that provides us with samples
- This gaussian distribution is chosen $\epsilon \sim \mathcal{N}(0, 1)$
- So we can obtain a sample computing

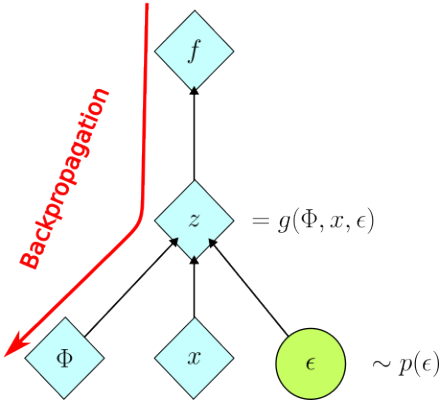
$$z = \mu + \sigma \odot \epsilon$$

The reparameterization trick

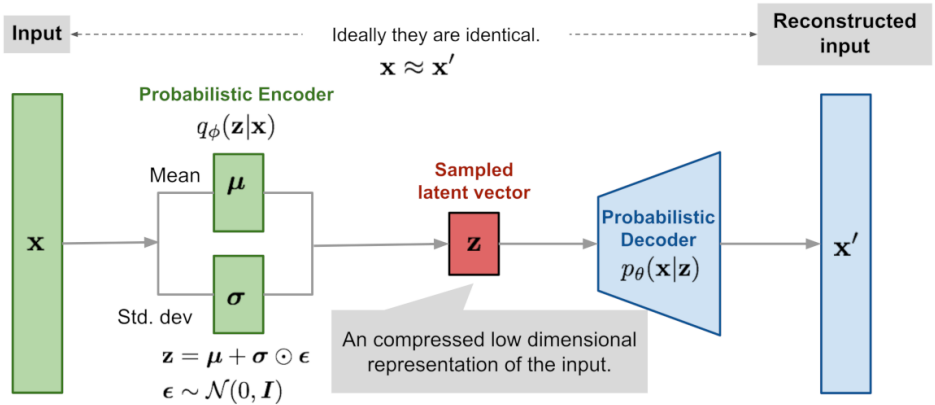
Original Computational Graph



Reparameterised



VAE Architecture



Problems when sampling from VAEs

- Even when we are trying to adjust the latent distribution to an a priori distribution, we can end with a mismatch between their densities
- It can happen that sampling high probability areas from the a priori distribution (gaussian) the decoded samples have low probability in the posterior distribution, but we obtain them frequently
- It can also happen the opposite, that the a priori distribution is not dense where the posterior distribution is dense, so these samples are hard to obtain

Problems when sampling from VAEs: Solutions

- Fit after training a more complex a priori distribution for the latents that assigns better the weights of the different probability densities (e.g.: a gaussian mixture)
- Use a more complex distribution than the gaussian
- Fit jointly a normalizing flow between encoder and decoder, so the distribution that is learned is more complex
- Use a hierarchical VAE where each level adjusts better to the data complexity

VAEs for Vector Quantisation

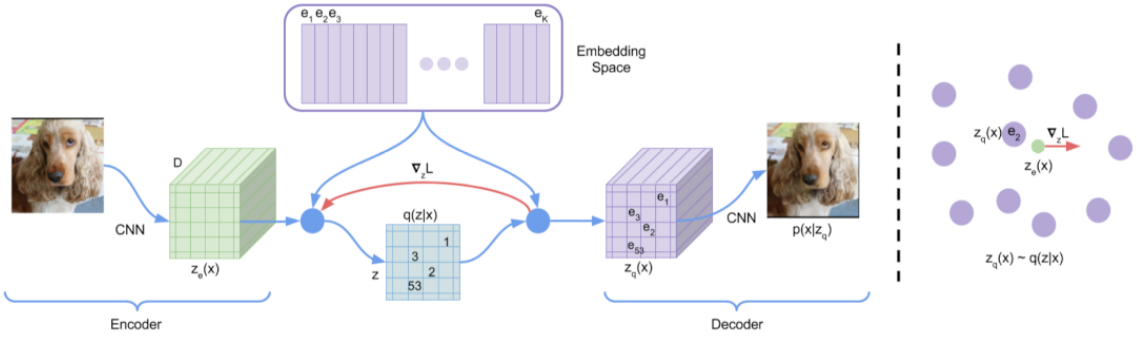
VQ-VAE (Google)

- VQ-VAE (Vector Quantised-Variational AutoEncoder) learns discrete latent variables, this can be more adequate for many problems (NLP, speech...)
- Vector quantization is a method for mapping K-dimensional vectors to a finite code, a set of vector centroids
- The architecture maintains a codebook of k vectors with dimensionality D
- The distribution q is then a discrete distribution

VQ-VAE: Architecture

- The encoder outputs a vector of dimension D for each variable in x
- The euclidean distance for the k elements in the codebook is computed for each variable and the nearest code is obtained to obtain the encoded sample
- Each variable of the encoded sample corresponds to a vector from the code
- The encoded sample is passed through the decoder to generate the output
- Codes are updated to reduce the reconstruction error

VQ-VAE: Architecture



This Python Notebook shows examples of Variational Autoencoders

- Variational Autoencoders Notebook (click here to open the notebook in colab)

If you download the notebook you will be able to use it locally (run jupyter notebook to open the notebooks)

BSC AI Factory

AI Factories

Co-funded by the European Union

Plan de Recuperación, Transformación y Resiliencia

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101234399 and Spain, Portugal, Romania and Türkiye. The JU receives support from the European Union's Horizon Europe Programme.

BSC AI Factory Partners

Affiliated entities