An Introduction to variational autoencoders (VAEs)

May 22, 2022

The idea here is to understand just enough to explain what these things are and to understand how and why they're used where they are used. No detailed knowledge is required at the moment of writing.

1 Introduction

1.1 Motivation

Discriminative modelling aims to learn a predictor given the observations, while generative models has the more general objetive of learning a joint distribution over all the variables. Generative models are in line with making theories in science. They enable hypothesis testing. They require stronger assumptions than purely discriminative models, which often leads to a higher asymptotic bias when the model is wrong.

The goal is to find abstactions of the world which can be used for multiple prediction tasks downstream. More concretely, the goal is to find disentagled, semantically meaningful, statistically independent and causal factors of variation in data — this is called unsupervised representation learning. VAEs are employed for this purpose. This can also be viewed as an implicit form of regularization: by forcing the representations to be meaningful for data generation, we bias the inverse of that process, i.e. mapping from input to representation, into a certain mould.

VAEs can be split in two coupled, but independently parametrized parts: the encoder, or the recognition model, and the decoder, or the generative model. One advantage of the VAE framework, relative to ordinary variational inference (VI), it that the recognition model (also called inference model) is now a (stochastic) function of input variables, making in more efficient on large data sets: the recognition model uses one set of parameters to model the relation between input and latent variables and as such is called "amortized inference". This is reasonably fast because by construction the recognition model can be done using a single forwardpass from input to latent variables. The price paid for this is that sampling induces sampling nois in the gradients required for

learning. The key VAE contribution is that this variance can be counteracted by what is known as the "reparametrization trick" (reorganized gradient computation which reduces variance).

VAEs marry graphical models and deep learning. The generative model is a Bayesian network of form $p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$, or in the case of multiple stochastic layers, a hierarchy such as: $p(\boldsymbol{x}|\boldsymbol{z}_L)p(\boldsymbol{z}_L|\boldsymbol{z}_{L-1})\cdots p(\boldsymbol{z}_1|\boldsymbol{z}_0)$. Similarly, the recognition model is also a conditional Bayesian network of form $p(\boldsymbol{z}|\boldsymbol{x})$ which can also be a hierarchy of stochastic layers. Inside each conditional may be a deep neural network, e.g. $\boldsymbol{z}|\boldsymbol{x} \sim f(\boldsymbol{x}, \boldsymbol{\epsilon})$ with f being the neural network mapping and $\boldsymbol{\epsilon}$ a noise random variable. Its learning algorithm is a mix of classical (amortized, variational) expectation maximization, but with the reparametrization trick ends up backpropagating through the many layers of the deep neural networks embedded inside it. You can play with VAEs by adding attention layers, extending them to dynamical models etc. GANs create great realistic images, but lack full support of the data. VAEs, like other likelihood-based models, generate more dispersed samples, but are better density models in terms of likelihood creation.

1.2 Probabilistic models and variational inference

Probabilistic models are a formalization of knowledge and skill. The degree and nature of uncertainty is specified in terms of (conditional) probability distributions. Let $\boldsymbol{x}$ be the vector represeting the set of all observed variables whose joint distribution we would like to model. We assume $\boldsymbol{x}$ is a random sample of an unknown distribution $p^*(\boldsymbol{x})$. We attempt to approximate this process with a chosen model $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. *Learning* is most comonly the process of searching for values of parameters $\boldsymbol{\theta}$ such that for any observed $\boldsymbol{x}$:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx p^*(\boldsymbol{x}) \tag{1}$$

1.2.1 Conditional models

Often we're actually interested in learning not $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, but $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ that approximates $p^*(\boldsymbol{y}|\boldsymbol{x})$. Here $\boldsymbol{x}$ is the input, ex. and image, and $\boldsymbol{y}$ is something we're interested in like a class label. To simplify notation, unconditional modelling is assumed, but it is almost always applicable to conditional modelling.

1.3 Directed graphical models and neural networks

We can parametrize conditional distributions with neural networks. VAEs in particular work with *directed* probabilistic models, also know as *probabilistic graphical models* (PGMs) or *Bayesian networks*. The joint distribution over the variables of such models factorizes as a product of prior and conditional distributions:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M) = \prod_{j=1}^{M} p_{\boldsymbol{\theta}}(\boldsymbol{x}_j|Pa(\boldsymbol{x}_j)) \tag{2}$$

where $Pa(\boldsymbol{x}_j)$ is the set of parent variables of node j in the directed graph. For root nodes the parents are an empty set, i.e. that distribution is unconditional. Before you'd parametrize each conditional distribution with ex. a linear model, and now we do it with neural networks:

$$\boldsymbol{\eta} = \text{NeuralNet}(Pa(\boldsymbol{x})) \tag{3}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|Pa(\boldsymbol{x})) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{\eta}) \tag{4}$$

Training Let dataset be $\mathcal{D}$. In fully observed models you know what you'd do:

$$\log p_{\boldsymbol{\theta}}(\mathcal{D}) = \sum_{\boldsymbol{x}\in\mathcal{D}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{5}$$

1.3.1 Maximum likelihood and minibatch SGD

The most common criterion of probabilistic models is maximum likelihood (ML). Maximization of the log-likelihood criterion is equivalent to minimization of KL divergence betwen data and model distributions. Chain rule, minibatch, etc, you know it. For the sake of notation, dataset has size $N_{\mathcal{D}}$, minibatch is called $\mathcal{M}$. Then

$$\frac{1}{N_{\mathcal{D}}}\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathcal{D}) \simeq \frac{1}{N_{\mathcal{M}}}\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathcal{M}) = \frac{1}{N_{\mathcal{M}}} \sum_{\boldsymbol{x}\in\mathcal{M}} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\S) \tag{6}$$

1.3.2 Bayesian inference

From a Bayesian perspective, ML can be improved through *maximum a posteriori* (MAP) estimation, or even further, inference of a full approximate posterior distribution over parameters.

1.4 Learning and inference in deep latent variable models

Fully observed directed models can be extended into directed models with latent variables. *Latent variables* are variables that are a part of the model, but which we don't observe (are not it dataset). Typical label is $\boldsymbol{z}$. In the case of unconditional modelling of $\boldsymbol{x}$, the directed graphical model would then represet a joint $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$. The marginal is then:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} \tag{7}$$

This is also called the *marginal likelihood* or the *model evidence* when taken as a function of $\boldsymbol{\theta}$.

1.4.1 Deep latent variable models

Deep latent variable models (DLVM) denote a latent variable model $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ whose distributions are parametrized with neural networks. Can be conditioned on some context, giving $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{y})$. Cool thing is that while the directed model is relatively simple, $p(\boldsymbol{x})$ can be wild.

1.4.2 Intractabilities

The main difficulty with maximum likelihood learning in DLVMs is that the marginal probability of data under the model is typically intractable. This is due to $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z}$ which does not have an analytic solution or efficient estimator. Hence you can't differentiate it w.r.t its parameters and optimize it. $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is intractable due to the intractability of $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$. While $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ is efficient to compute,

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})} \tag{8}$$

gives problem in DLVMs. Hence we need approximate inference techniques.

2 Varitional autoencoders

2.1 Encoder or approximate posterior

To solve intractabilities, we introduce a parametric *inference model* $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. This model is called the *encoder* or *recognition model*/ $\boldsymbol{\phi}$ are called the *variational parameters*. They are optimized s.t.:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \approx p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \tag{9}$$

Like a DLVM, the inference model can be almost any directed graphical model:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = q_{\boldsymbol{\phi}}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M|\boldsymbol{x}) = \prod_{j=1}^{M} q_{bm\phi(\boldsymbol{z}_j|Pa(\boldsymbol{z}_j),\boldsymbol{x})} \tag{10}$$

This can also be a neural network. In this case, parameters $\boldsymbol{\phi}$ include the weights and biases, ex.

$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) = \text{EncoderNeuralNet}_{\boldsymbol{\phi}}(\boldsymbol{x}) \tag{11}$$

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma})) \tag{12}$$

Typically, one encoder is used to perform posterior inference over all of the datapoints in the dataset. The strategy used in VAEs of sharing variational parameters across datapoints is also called *amortized variational inference*.

2.2 Evidence lower bound (ELBO)

The optimization objetive of the VAE, like other variational methods, is the *evidence lower bound (ELBO)*. It is typically derived with Jensen's inequality, but here it isn't:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\right] \tag{13}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right] \tag{14}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right] \tag{15}$$

$$= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right]}_{=\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})\text{ELBO}} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{z})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right]}_{=\mathcal{D}_{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}(\boldsymbol{z}|\boldsymbol{x})})} \tag{16}$$

D_{KL} is non-negative and 0 iff $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ equals the true distribution. The first term is called the *variational lower bound* or *evidence lower bound*. Due to non-negativity of D_{KL}, ELBO is a lower bound on the log-likelihood of the data:

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) = \log p_{\theta}(\boldsymbol{x}) - D_{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})) \tag{17}$$

$$\leq \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{18}$$

So, $D_{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$ determines two distances:

1. by definition the approximate posterior from the true posterior
2. the gap between ELBO $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}$ and the marginal likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$: this is also called the *tightness of the bound*

Thus, by maximizing ELBO w.r.t. parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$, we'll concurently optimize:

1. approximately maximize the marginal likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, which means that the generative model will become better
2. minimize the KL divergence of the approximation $q_{\boldsymbol{\phi}(\boldsymbol{z}|\boldsymbol{x})}$ from the true posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$

We do this optimization with SGD. Given a dataset, ELBO is the sum (or average) of individual point ELBO's. Individual-datapoint ELBO is intractable in general, but there are good unbiased estimators of $\tilde{\nabla}_{\boldsymbol{\theta},\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})$.

Grad of ELBO w.r.t. generative model parameters $\boldsymbol{\theta}$ are simple to obtain:

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}}\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}(\boldsymbol{x},\boldsymbol{z})} - \log q_{\boldsymbol{\phi}(\boldsymbol{z}|\boldsymbol{x})}\right] \quad = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\nabla_{\boldsymbol{\theta}}\left(\log p_{\boldsymbol{\theta}(\boldsymbol{x},\boldsymbol{z})} - \log q_{\boldsymbol{\phi}(\boldsymbol{z}|\boldsymbol{x})}\right)\right] \simeq \nabla_{\boldsymbol{\theta}}\left(\log p_{\boldsymbol{\theta}(\boldsymbol{x}}\right. \tag{19}$$

where we used a Monte-Carlo estimator of the second line and $\boldsymbol{z}$ is a random saple from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$.

2.3 Reparametrization trick

Unbiased gradiest w.r.t. variational parameters $\boldsymbol{\phi}$ are more difficult to obtain. For this we'll use the reparametrization trick. This is just a change of variable which is cool over the gradient of expectation and we can get a Monte-Carlo estimator. It essentially "externalizes" the noise in $\boldsymbol{z}$. I'm not going to write this out. Point is you get your estimate and can do gradient descent and it's all easy to implement in say PyTorch. Usually the reparametrization is done with a normally distributed variable. It's all interesting and you should go read it.