Latent variable models

Lesson No. 04

João Victor da Silva Guerra [*]

1 Deep generative models

In unsupervised learning, a generative model is a powerful tool to learn any type of data distribution. The model learns the true data distribution with a training procedure and after is able to generate new data points with some variations. However, it is not always possible to learn implicity or explicity the distribution of our data and then the model learns a distribution as similar as possible to the true distribution.

Deep generative models includes:

- Latent variable models (e.g. Variational autoencoders [1]);
- Implicit generative models (e.g. Generative adversarial networks [2]);
- Exact likelihood models (e.g. PixelCNN [3], RealNVP [4], Glow [5], Flow++ [6]).

2 Latent variable models

A latent variable model (LVM) is a statistical model that contains latent, i.e. unobserved, variables (z) that relates to a set of observable, i.e. manifest, variables (x). When all random variables of our model are observed (Fig. 1a), we should work with exact likelihood models, such as autoregressive and flows. However, when some random variables are unknown (Fig. 1b), we should work with LVMs.

(a) All random variables are observable.

(b) Some random variables are hidden.

Figure 1. Bayes newtorks of random variables.

The design of a LVM could be made by understanding the causal process that generates the data distribution. However, the latent variables (z) and their interactions with observation (x) are not usually known in a scientific process. Nowadays, the most popular models make little assumptions about what are the latent variables and the best procedure to define them is an active area of resear ch.

2.1 Example of latent variable model

Here, we introduce a simple LVM that is a Bernoulli LVM (Fig. 2). The latent space (z) is a set of K binary variables and distributed based on a Bernoulli distribution $p(z|\beta)$ as follows:

[*]RA: 117410 - j117410@g.unicamp.br

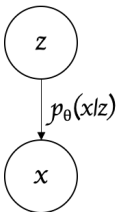

Figure 2. Bernoulli latent variable model

$$z = (z_1, ..., z_K) \sim p(z|\beta) = \prod_{k=1}^{K} \beta_k^{z_k} (1-\beta_k)^{1-z_k} \tag{1}$$

The observation space (x) is also a set of L binary variables and distributed based on a Bernoulli distribution as follows:

$$x = (x_1, ..., z_L) \sim p_\theta(x|z) = \text{Bernoulli}(x_i|\text{DNN}(z)) \tag{2}$$

where DNN is a Deep Neural Network. The output of this DNN or any other function approximator is the coefficients that parametrize the Bernoulli distribution of the observation space.

In this model, the generation procedure consists of:

1. sample $z \sim p(z|\beta)$;
2. map z through $\text{DNN}(z)$;
3. generate $x \sim p(x|z)$.

2.2 Training LVM

A LVM has a set of parameters θ. The training procedure consists of optimizating these parameters with maximum likelihood. The marginal likelihood $p(x)$ is not explicitly defined and it is evaluated as follows:

$$\log p(x) = \log\left(\sum_z p(x|z)\, p(z)\right) \tag{3}$$

$$p(z|\phi(x)) = \prod_{k=1}^{K} \phi_k^{z_k} (1-\phi_k)^{1-z_k} \tag{4}$$

The trained parameters will be the parameters that maximize the marginal likelihood $p(x)$ as follows:

$$\theta \leftarrow \text{argmax}_\theta \left[\log p_\theta(x)\right] = \text{argmax}_\theta \left[\log \sum_z p_\theta(x|z)\, p(z|\phi(x))\right] \tag{5}$$

However, the main problem with this procedure is that it is not very scalable in how complex is the latent space. For a string that has 32 binary characters in the latent space, the summation in Eq. (3) occur over 4 billion terms, which is not very tractable.

2.3 Variational Inference

Currently, the calculations of the prior $p(x)$ and the conditional probability $p(x|z)$ have $O(1)$ complexity. The search of a efficient procedure to calculate the marginal probability $p(x)$ is still necessary; however, it is challenging due to its intractable sum.

Assume an oracle $p(z|x)$, the marginal probability $p(x)$ is:

$$p(x) = \frac{p(x,z)}{p(z|x)} = \frac{p(x|z)\, p(z)}{p(z|x)} \tag{6}$$

Given the graphical model described in Fig. 2, we need to define the true prosterior distribution of the latent variable; however, we do not have access to the true posterior. To overcome this, the variational inference

approach approximate the true posterior $p(z|x)$ with a variational distribution $q_x(z)$, which is achieved by an optimization problem that minimizes the KL divergence between these two distributions.

$$\begin{aligned} D_{KL}[q_x(z)||p(z|x)] &= \mathbb{E}_{z\sim q_x(z)}[\log q_x(z) - \log p(z|x)] \\ &= \mathbb{E}_{z\sim q_x(z)}\left[\log q_x(z) - \log \frac{p(x|z)p(z)}{p(x)}\right] \\ &= \mathbb{E}_{z\sim q_x(z)}[\log q_x(z) - \log p(x|z) - \log p(z) + \log p(x)] \\ &= \mathbb{E}_{z\sim q_x(z)}[\log q_x(z) - \log p(x|z) - \log p(z)] + \log p(x) \end{aligned} \tag{7}$$

For all these terms that can be calculated efficiently, we should approximate posterior and steer it towards the true posterior with scalable stochastic optimization, which will minimize the expectation terms that only depends on z.

2.3.1 Variational Lower Bound

From the previous equation, we have an objective function amenable to stochastic optimization. But rearranging it in the form:

$$\log p(x) = \mathbb{E}_{z\sim q_x(z)}[\log p(z) + \log p(x|z) - \log q_x(z)] + D_{KL}[q_x(z)||p(z|x)] \tag{8}$$

The expectation term is the variational lower bound (VLB). Further, the optimal $q_x(z)$ of VLB is the true posterior $p(z|x)$, at which point VLB is tight (i.e. equal to the log of the marginal probability $p(x)$).

To start fitting the model, we train it by maximizing the VLB under a data distribution $x \sim p_{\text{data}}$. In this condition, we have

$$\mathbb{E}_{x\sim p_{\text{data}}}[\mathbb{E}_{z\sim q_x(z)}[\log p(z) + \log p(x|z) - \log q_x(z)]] \leq \mathbb{E}_{x\sim p_{\text{data}}}[\log p(x)] \tag{9}$$

2.3.2 Wake-sleep algorithm

In the Bernoulli example setting (Eqs. (1) and (2)), the VLB becomes:

$$\mathbb{E}_{z\sim q_x(z)}\left[\log p(z|\beta) + \log p(x|z,\theta) - \log q_x(z|\phi(x))\right] \tag{10}$$

To optimize the expectation from which z is drawn, there is an algorithm that is called wake-sleep algorithm, which has two steps: wake phase and sleep phase.

The wake phase (train prior and conditional probability) consists of:

- sample $x \sim p_{\text{data}}$ and $z \sim q(z|\phi(x))$;
- maximize VLB w.r.t. parameters θ and β.

The sleep phase (train approximate posterior) consists of:

- sample $z \sim p(z|\beta)$ and $x \sim p(x|z,\theta)$;
- minimize reverse KL divergence w.r.t. coefficients of ϕ, but having samples from p_{model}.

The first phase maximize the VLB with $x \sim p_{\text{data}}$ and the last phase minimize the reverse KL divergence with $x \sim p_{\text{model}}$. The sleep phase does not guarantee the bound is tight due to the optimization using a data generated from a model instead of the actual data, which does not minimize the same objective. Further, this algorithm is not scalable to large datasets because it would be necessary to store an approximate posterior per data point.

Finally, taken together the variational inference, variational lower bound and wake-sleep algorithm, we described the Helmholtz Machine [7].

References

[1] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2013. arXiv: 1312.6114 [stat.ML].

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial networks*, 2014. arXiv: 1406.2661 [stat.ML].

[3] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, *Conditional image generation with pixelcnn decoders*, 2016. arXiv: 1606.05328 [cs.CV].

[4] L. Dinh, J. Sohl-Dickstein, and S. Bengio, *Density estimation using real nvp*, 2016. arXiv: 1605.08803 [cs.LG].

[5] D. P. Kingma and P. Dhariwal, *Glow: Generative flow with invertible 1x1 convolutions*, 2018. arXiv: 1807.03039 [stat.ML].

[6] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel, *Flow++: Improving flow-based generative models with variational dequantization and architecture design*, 2019. arXiv: 1902.00275 [cs.LG].

[7] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel, "The helmholtz machine," *Neural Comput.*, vol. 7, no. 5, pp. 889–904, Sep. 1995, ISSN: 0899-7667. DOI: 10.1162/neco.1995.7.5.889. [Online]. Available: https://doi.org/10.1162/neco.1995.7.5.889.