DFL - Week 6

Scribed by Jean-Michel Amath Sarr

7 November 2019

1 Context

Week 6 was the final week of the Depth First Learning Group on Normalizing Flows. In the following, I report the main points of the discussions that took place on Wednesday 30th of October after reading [4] and [1]. The participants were: Steve Kroon, Scott Cameron, Lood van Niekerk, James Allingham, Suvarna Kadam, Bharathi Srinivasan, Witold Szejgis, and Jean-Michel Amath Sarr. We also had a guest: Rianne van den Berg. The question raised at the beginning of the session were:

- Why do we say that normalizing flows are not volume-preserving? What is the difference with a volume-preserving transformation?
- Why do we talk about the free energy bound?
- Are we using auxiliary variables in Normalizing Flows (NF)?
- What is the relationship between the free parameters λ outputted by the NF and the variational parameter ϕ in NICE [2]?
- What is the significance of preconditioning the optimizer?
- What would be an interpretation of the reparametrization for the planar flows?
- Why to do this at all? What NF gives us that regular variational autoencoders don't give us?
- Is there some tradeoff between richer prior and faster sampling?

2 Discussion on Normalizing Flows

2.1 Volume-preservation

We started to talk about the non-volume preserving question. James answers that it is simply because the log determinant is different from 1. Steve added that in case of non-linear transformations, a point and a volume around are

going to be transformed concerning the change of variable formula (equation 5 of [4]) and the amount of transformation is computed by the determinant of the Jacobian, which is a measure of volume transformation.

As a matter of comparison, James pointed out that the determinant in the NICE paper was always 1. Another example of non-volume preserving flows are NFP [3].

2.2 Relation between NF and NICE

About the relationship between the free parameter outputted by NF and NICE. According to Steve, in the NICE paper, they are doing mainly density estimation, but in case they wanted to perform variational inference, they would require to output the parameter of the neural network. Scott noticed that even if the authors of [4] compared NF with NICE (fig 3), they did not specify the architecture they used with their NICE implementation. Indeed, it is a bit strange that NICE underperform so much, while in principle you can allow arbitrary complexity by combining layers.

2.3 Implementation matters and code

The implementation of planar flow to get the result for fig 3 in [4] is very difficult according to Rianne, she mentioned the two moon as an example. It is unstable because it quickly gets stuck in one mode. Replicate this figure requires learning to take place with optimizing a KL, whereas, in figure one, the parameters are random.

Steve said that we can interpret planar flow by considering the expression $w^T z + b$ as a hyperplane in the general case, h is a strictly increasing function (a bijection, this is why the authors used tanh instead of ReLU) You then need to reparametrize the parameters at each iteration to keep having an invertible flow.

One question that was part of the exercise was: why do we output the flow parameters for each input rather than outputting static parameters. Well as Steve said, not outputting the flow parameter would we similar to use a variational autoencoder (VAE).

James asked about the significance of reconditioning, Rianne used Adamax, and report that sometimes with VAE Adamax is more stable empirically than Adam. Even if SGD/SGD with momentum can perform better, they are much more difficult to stabilize. Still, on the optimization side, Steve asked about the annealing of the log-posterior which is opposite than what Kingma et al suggested in VAEs. Rianne answered that the two schemes are not exact opposites, in fact, they are doing something similar. In a lot of schemes, the KL term between the prior and the posterior is annealed. In equation 20 of

[4], one of these terms is actually in the modified bound. After some trying, she finds that the normal annealing works slightly better than what is suggested.

2.4 Relation with HMC

Suvarna asked about the auxiliary variable technique. Steve recalls that the technique was introduced in the VAE tutorial from week 5 and that it is a technique to enrich the latent space. It is mentioned in [4] because in some cases, Hamiltonian Monte Carlo (HMC) can be seen as NF. HMC comes from physics, and this explains the origin of the term free energy.

Scott added that in physics, the log of the normalizing constant is called free energy, and that's pretty much similar to the log evidence in statistics. There is another connection between variational inference and statistical physics. In lots of systems, we use a mean-field formulation and the Jensen inequality to maximize the lower bound of the free energy. The maths and the equations are the same. Steve mentioned that we should distinguish the free energy from the free energy bound, the later is the quantity we usually maximize.

2.5 Value added by NF in comparison to VAE

We made a transition and Steve wanted the discussion to port on the value added of NF. He asked: if we are not using a variational parameter per example, then we can use the flow part as an extension of the decoder. So if we are going to output the flow parameters, why not encoding that into a higher dimensional Gaussian, and use it as a base density. He added, that in principle with a powerful enough encoder and decoder you should not need NF.
James mentioned the issue of overfitting that has been touched in [1], where the authors noticed that with too much capacity in your network, you could overfit on your validation set. He suggested that maybe using the flow parameters provides a more robust approach because they are not optimized, at least not directly with backpropagation.
In Scott's opinion, it is really hard for a VAE to overfit, except when the inference net is over-parametrized. If the latent space is bigger, overfitting must come from a too large decoder.

Lood sees NF as almost skip connections between the encoder and the decoder, except that you recover densities instead of transferring information like in a skip connection. In his view, the advantage of NF is that it may allow more interpretable and meaningful latent space.

Finally, Steve gave his view on the matter. He said that at first he thought that NF should be something to plug in the middle of a VAE when you don't know the structure of the latent space, but it is not the only use case. In Deep Latent Gaussian Models (DLGM) assuming that parameters are generating the data, you can also use NF as a way to get a better interpretable latent space, or

in other words to get a better representation of the generating factor of the data.

Rianne thought that if you make the flow parameter not input dependent (i.e let's make them global learnable parameters with SGD), it is not similar to transferring those layers directly to the decoder. She added that if you have a very good decoder, you might not need a non-gaussian encoder. The problem happens when your latent space becomes meaningless in the sense that the posterior collapse the prior and you lose the autoencoder structure because the encoder does nothing. It can be problematic if you need your encoder for another downstream task. But there are indeed other ways that don't require putting the flow on the posterior. Some people put it on the prior, some people put ut at the end of the decoder to predict $p(z|x)$. So at the end where you put the flow depends on your downstream task.

Steve asked if you can do similar things with a regular VAE and a NF.

Rianne thought that it is possible, but it would require a much more powerful decoder because there is a difference between what you can theoretically achieve (global optimum) and what you achieve during numerical optimization. And in case you have a powerful decoder, you could easily get to a local minimum where your encoder does nothing.

3 Discussion on Inference Suboptimality

One of the questions we have been working in the weekly exercise and that James brought up was to try to figure out where the true posterior comes from in [1] figure 2. Rianne answered that the author fixed the generative model, i.e the decoder and the prior. With that in mind, you can compute the posterior through Bayes rule. She finds the term true posterior to be misleading. In the paper, the true posterior refers to the posterior you compute with a fixed decoder.

James commented the figure 3, indeed, these figures illustrate overfitting if the encoder or the decoder is too big, and that can be reflected in the amortization gap.

4 General discussion on Generative Models

4.1 Guideline to train Generative Models ?

Steve asked a question to Rianne about the dimensionality of the latent space in her implementation of NF: "If you are going to use a planar flow, how is the depth of the flow interact with the size of the latent dimension, and what about the type of the flow?"

Rianne responded that it is similar to a neural network. Mainly if you have a 2 hidden layer neural network with a large input/output and a small hidden layer, then the model is going to be less expressive. But you don't need to be to restrict the hidden dimension, (i.e no bottleneck). For instance, this is the case of Sylvester flows.

Steve brought up another question about Rianne's implementation. The question was about gated convolutional layer, why do we need them anyway? Rianne answered that they tried many architectures, another one was a ResNet kind of encoder/decoder that overfits less quickly, and appear better in retrospect.

Witold asked what dimension in the latent space is right for generative models in the context of information theory.
From Steve's perspective, the dimension does not matter as long as there is enough space. He reformulated the question and asked if there is a good set of guidelines to know how big you make your encoder/decoder, how complicated to make your flow, and how big to make the dimensionality of the latent space. In Rianne's view, there are no such guidelines. There are hierarchical VAEs, with dimensions exceeding even the dimension of the input. It depends on the task you are interested in. The idea is to take inspiration from previous papers on the subject. In practice, there is a trade-off with regards to the dimensionality of the latent space. You want to compress it as much as possible to get a nice compressed representation. But too much compression can arm your generative model, so in the end, it depends on your goal.

Then Steve asked about what to do when your generative model (wether a NF or a VAE) is not working. What do you start playing with?

Rianne said that it depends on what it means by saying that it doesn't work. For instance, in many instances, it is really hard to make your VAE overfit. You need to correctly diagnose the issue with your VAE, in that matter, the results from [1] are the tip of the iceberg. For example, when you work with images, sampling from VAE can result in blurry images, and there are many conjectures on why it's the case. So understanding VAEs or taming it is not done, it's not finished.

4.2 What makes a good representation?

Furthermore, James asked Rianne her opinion on what makes a good representation. From last week's discussion, we discussed the view that a good representation easy models the data. But the definition was not good enough, maybe it also depends on the downstream task. For instance, if the task is classification, then a good feature of the latent space is that it is easily separable.

Rianne also agreed that the downstream task is important in deciding what makes a good representation. She gave two additional examples. You can use a VAE for compression, and in that case, you don't care about separability. You can also use a VAE to uncover ground truth factors, for instance in modeling

proteins, you can use the latent space to model orientation or rotation given that you feed your model with images. In that case, you are interested in the physical sense of your latent space.

James then asked Rianne about her opinion why for example in the NICE paper the author adopted the view that a good representation should be easy to learn.

Rianne guessed that maybe it is useful to have an easily learnable representation to walk around in the latent space. It might also be useful to have a representation that is well organized in a semantically meaningful way. For example in MNIST, to have separable latent factors.

Steve then asked if people use NF in other settings. For instance, are people using NF in probabilistic graphical models to learn more about the hidden variables behind the generating distribution of a dataset? Rianne report that some people use NF in reinforcement learning to learn policies.

Jean-Michel asked about the best generative model to learn a rich latent space with a small dataset. In Steve's view, a good option is to start with a simple model. Survana added that you can use a feature learner with high capacity with a related dataset. For example, it can be the encoder of a VAE, then use the small dataset. It is an instance of transfer learning.

References

[1] C. Cremer, X. Li, and D. Duvenaud. Inference suboptimality in variational autoencoders. *arXiv preprint arXiv:1801.03558*, 2018.

[2] L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

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

[4] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.