DEEP PRIOR, BAYESSIAN MASHINE LEARNING PROJECT

A PREPRINT

Ivan Anokhin
MS student. Data Science
Skoltech
Ivan.Anokhin@skoltech.ru

Egor Nuzhin
MS student. Information Science and Technology
Skoltech
Egor.Nuzhin@skoltech.ru

October 26, 2018

ABSTRACT

The machine learning community is well-practised at learning representations of data-points and sequences. But an efcient learner is one who reuses what they already know to tackle a new problem. For a machine learner, this means understanding the similarities among a datasets. In order to do this, one must take seriously the idea of working with datasets, rather than datapoints, as the key objects to model. Towards this goal, we compared two different models: Deep prior article and Towards a Neural Statistician (TNS).

***K*eywords** ELBO · Deep prior · VAE · Bayesian Inference · Towards a Neural Statistician

1 Problem formulation

1.1 Towards a Neural Statistician [1]

We are given datasets D_i for $i \in I$. Each dataset $D_i = \{x_1 ... x_n\}$ consists of a number of i.i.d samples from an associated distribution p_i over $\mathbb{R}^n$. The task can be split into learning and inference components. The learning component is to produce a generative model $\hat{p_i}$ for each dataset D_i. We assume there is a common underlying generative process p such that $p_i = p(\dots|c_i)$ for $c_i \in \mathbb{R}^l$ drawn from $p(c)$. We refer to c as the context. The inference component is to give an approximate posterior over the context $q(c|D)$ for a given dataset produced by a statistic network. The main goal is to find a posterior generative model - origin distribution of the target dataset $p(x|D_t)$.

In our experimentts we used full TNS model (Figure 1). We use multiple stochastic layers $z_1 ... z_k$ with skip-connections for both the inference and generative networks. The probability of a dataset D is then given by:

$$P(D) = \int p(c) \prod_{x \in D} \int p(x|c, z_{1:L}, \theta) p(z_L|c, \theta) \prod_{i=1}^{L-1} p(z_i|z_{i+1}, c, \theta) dz_{1:L} dc$$

The full approximate postiarior factorizes analogously as:

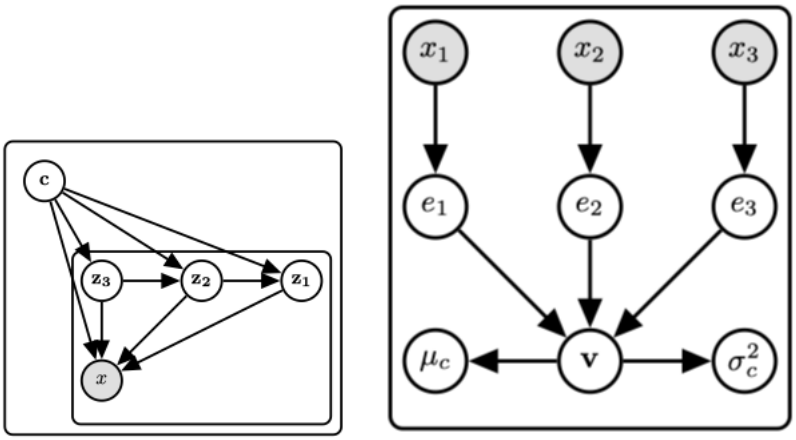
$$q(c, z_{1:L}|D, \phi) = q(c|D, \phi) \prod_{x \in D} q(z_L|x, c, \phi) \prod_{i=1}^{L-1} q(z_i|z_{i+1}, x, c, \phi)$$

To find optimal parameters we optimize ELBO. For convenience we give the variational lower bound as sum of a three parts, a reconstruction term R_D , a context divergence C_D and a latent divergence L: $L = R_D - C_D - L_D$

$$R_D = \mathbb{E}_{q(c|D,\phi)} \sum_{x \in D} \mathbb{E}_{q(z_{1:L}|c,x,\phi} \ln p(x|z_{1:L}, c, \theta)$$

$$C_D = D_{KL}(q(c|D, \phi)||p(c)$$

$$L_D = \mathbb{E}_{q(c,z_{1:L}|D,\phi)} \left[\sum_x D_{KL}(q(z_L, x, \phi)||p(z_L|c, \theta) + \sum_{i=1}^{L-1} D_{KL}(q(z_i|z_{i+1}, c, x, \phi)||p(z_i|z_{i+1}, c, \theta) \right]$$



(a) (b)

Figure 1: Full neural statistician model (a) and the statistic network (b)

Figure 2: Full neural statistician model

As statistic network $q(c|D, \phi)$ we use a feedforward neural network consisting of three main element (Figure 1):

- An instance encoder E that takes each individual datapoint x i to a vector $e_i = E(x_i)$
- An exchangeable instance pooling layer that collapses the matrix $(e_1, ..., e_k)$ to a single pre-statistic vector v. Examples include elementwise means, sums, products, geometric means and maximum. We use the sample mean for all experiments.
- A final post-pooling network that takes v to a parameterization of a diagonal Gaussian.

1.2 Deep prior model [2]

In the paper Deep Prior Alexandre Lacoste et al. consider a hierarchical Bayes approach across N tasks. Each task has its own parameters w_j , with $W = \{w_j\}_{j=1}^N$

Author's goal was to learn a prior from previous tasks by learning a probability distribution $p(w|\alpha)$ over the weights w of a network, parameterized by α

For simplicity, in this work, they consider a point estimation of $p(\alpha|D)$ - its MLE estimation. This can be justify by considering scenarios where we have a lot of samples to learn α across many tasks while the uncertainty we truly care about is the uncertainty over w_j for new tasks.

To find MLE estimation of α they have used ELBO approximation across join distribution of w and new latent variable z. (See the paper for the detailed explanation)

Authors use following following assumption on true distribution:

$$p(z_j) = N(0, I)$$
$$p(z_j, w_j|\alpha) = p(z_j)\delta_{h_\alpha(z_j)-w_j}$$
$$p(z_j, w_j|\alpha, S_j) = p(z_j|\alpha, S_j)\delta_{h_\alpha(z_j)-w_j}$$

where S_j is dataset j

And on family of distribution q:

$$q(w_j, z_j) = q(z_j|S_j)\delta_{h_\alpha(z_j)-w_j}$$

Using the assumption above the ELBO can be derived:

$\ln \text{p(D)} \geq E_{q(w_j,z_j)} \sum_{j=1}^{N} \sum_{i=1}^{N} \ln p(y_{ij}|x_{ij}, w_j) - KL(q||p) =$
$\sum_{j=1}^{N} E_{q_{\theta_j}(z_j|S_j)} \sum_{i=1}^{N} \ln p(y_{ij}|x_{ij}, h_\alpha(z_j)) - \sum_{j=1}^{N} E_{q(w_j,z_j)} \ln \frac{q_{\theta_j}(z_j|S_j)\delta_{h_\alpha(z_j)-w_j}}{p(z_j)\delta_{h_\alpha(z_j)-w_j}} =$
$\sum_{j=1}^{N} E_{q_{\theta_j}(z_j|S_j)} \sum_{i=1}^{N} \ln p(y_{ij}|x_{ij}, h_\alpha(z_j)) - \sum_{j=1}^{N} KL(q_{\theta_j}(z_j|S_j)||p(z_j))$

Now authors notice that there is no longer need to explicitly calculate the KL on the space of w, we can simplify the likelihood function to the following $p(y_{ij}|x_{ij}, z_j, \alpha)$, which can be a deep network, parameterized by α taking both x_{ij} and z_j as inputs. This contrasts with the previous formulation where $h_\alpha(z)$ produces all the weights of a network, yielding a really high dimensional representation and slow training.

2 Implementation details

2.1 Towards a Neural Statistician

During implementation we faced with:

1. Ambiguity of Latent decoder network. Authors didn't provide architecture of the network for experiments on simple 1-D distributions and MNIST dataset. We supposed that architecture has the same structure like for OMNIGLOT dataset.
2. Authors didn't provide any information about training procedure. In our experiments we used usual Adam optimizer and didn't apply ant regularization techniques.
3. We didn't manage to learn our model on MNIST dataset. Probably, it caused by undefined training procedures or low computational resources.

2.2 Deep Prior model

During implementation of the paper several issue have emerged:

1. There was no details on encoder architecture but authors only point out that they uses IAF for the best performance - we implemented standard reparametrisation trick encoder firstly but it did not produce good result. So we implement planar flow[3] along with IAF [4] as well.
2. Authors claim to use 12 layers (Linear, Normalization, Relu) in the decoder, but in this case our model seems to struggle to learn anything. We have used only 4 layers in decoder.
3. The main issue is our inability to reproduce result from the paper (to reproduce well enough). The reason for this may lay in the previous two issues or in insufficient training time and insufficient size of the dataset.

3 Experiments

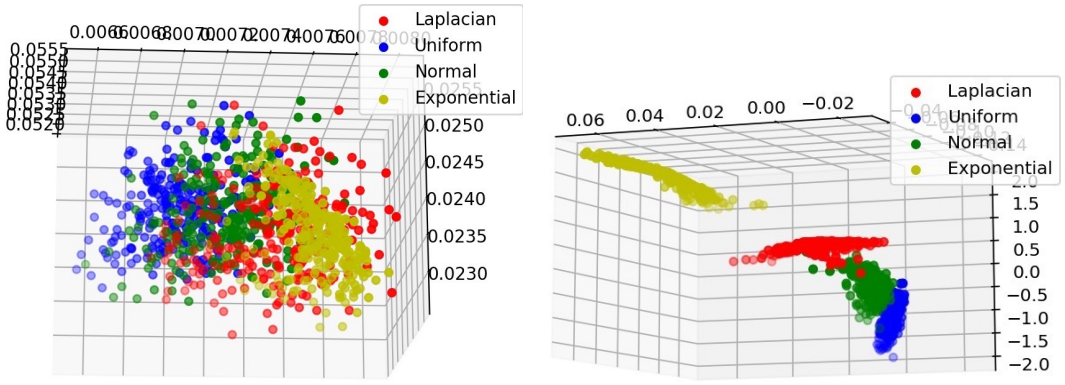
3.1 Towards a Neural Statistician

In our experiment we wanted to know if the neural statistician will learn to cluster synthetic 1-D datasets by distribution family. We generated a collection of synthetic 1-D datasets each containing 200 samples. Datasets consist of samples from either an Exponential, Gaussian, Uniform or Laplacian distribution with equal probability.

The architecture for this experiment contains a single stochastic layer with 32 units for z and 3 units for c, . The model $p(x|z, c;)$ and variational approximation $q(z|x, c;)$ are each a diagonal Gaussian distribution with all mean and log variance parameters given by a network composed of three dense layers with ReLU activations and 128 units. The statistic network determining the mean and log variance parameters of posterior over context variables is composed of three dense layers before and after pooling, each with 128 units with Rectied Linear Unit (ReLU) activations. Additionally we computed Silhouette score to estimate clasterization performance.

First experiment was performed on untrained network. On Figure 3 each point is the mean of the approximate posterior over the context $q(c|D;)$ where $c \in R^3$. Each point is a summary statistic for a single dataset with 200 samples. As expected we don't see clear clusters. And silhouette score reinforce our observations: 0.11.

Second one was performed on trained network (Figure 3). Now clusters are expressed much more explicit and silhouette score is significantly bigger: 0.57.



(a) (b)

Figure 3: Mean context of datasets using untrained (a) and trained (b) statistic network

Figure 3 shows 3-D scatter plots of the summary statistics learned. Notice that the different families of distribution cluster.

We also generated new samples subject given dataset according to Towards a Neural Statistician article. Results are shown on Figure 4.

As you can see, generated datasets are very similar to conditioned.

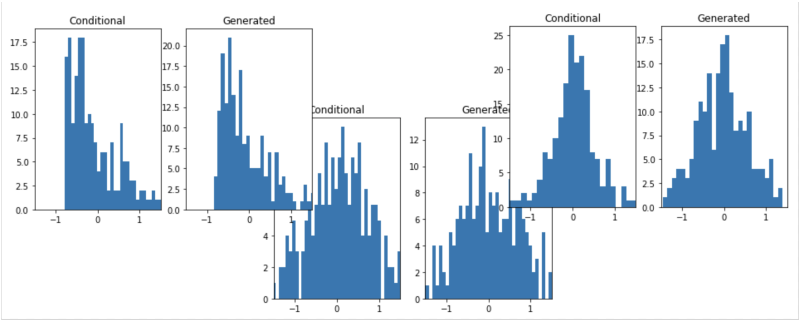


Figure 4: Generated samples

3.2 Deep Prior model

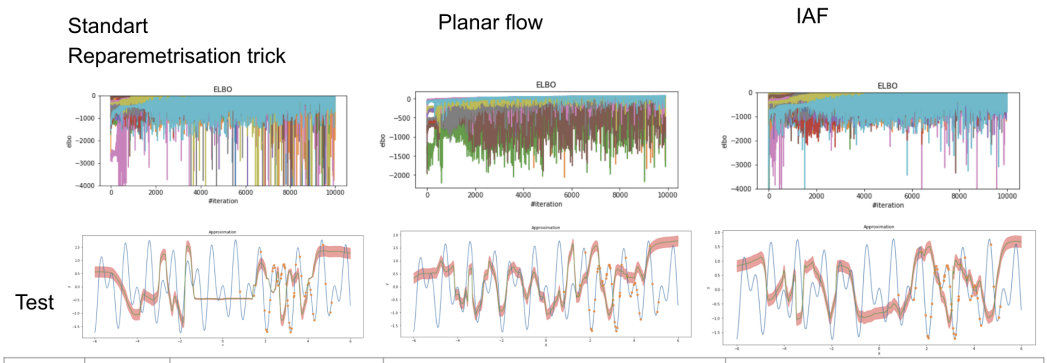
In the paper authors use regression task on the signal dataset to test their model. They demonstrate the ability of our model to learn a good prior on a dataset of periodic signals. The model successfully generalizes the periodic structure to unseen signals, while maintaining appropriate uncertainty about which periodic signal the data is sampled from.

We concentrate on regression task as well.

We compare exclusivity of three version of our encoder. The first one uses reparametrization trick, the second one uses Planar flow and the last one uses Inverse Autoregressive flow.

It can be seen from the figure 5 that IAF produces the best result, Reparametrization trick produces the worst result and Planar flow is in the middle.

We conduct same experiments as in the deep prior paper. We have try to see how our approximation works depending on the number of point in the test dataset. The number of point that we used are 1, 2, 8 and 256. We can see in figure 6 that results of approximation in the paper is much better than we could achieve. The reason for this may be insufficient training time and insufficient datasets size.



MSE	Test	2.689	2.484	2.315
	Train	4.851	4.093	4.165

Figure 5: Different flows comparison

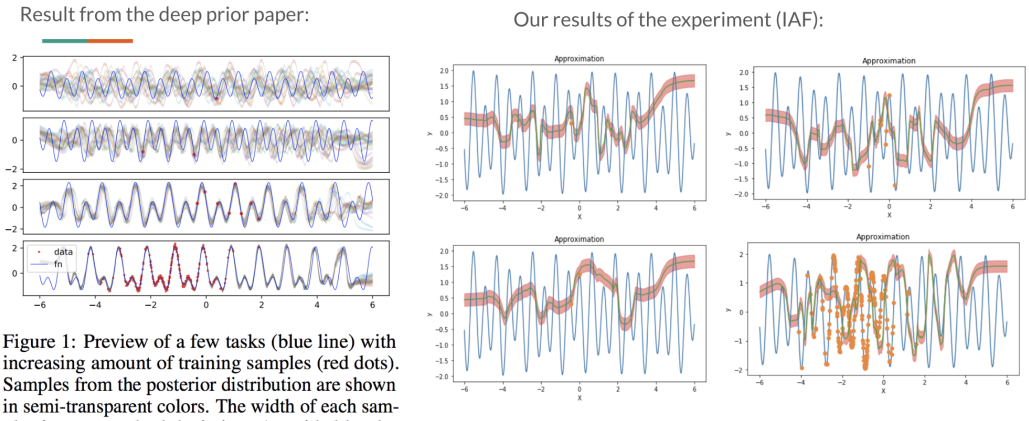


Figure 1: Preview of a few tasks (blue line) with increasing amount of training samples (red dots). Samples from the posterior distribution are shown in semi-transparent colors. The width of each samples is two standard deviations (provided by the predicted heteroskedastic noise).

Figure 6: Paper results compared with implementation results

References

[1] Harrison Edwards and Amos Storkey. Towards a neural statistician. *arXiv preprint arXiv:1606.02185*, 2016.

[2] Alexandre Lacoste, Thomas Boquet, Negar Rostamzadeh, Boris Oreshki, Wonchang Chung, and David Krueger. Deep prior. *arXiv preprint arXiv:1712.05016*, 2017.

[3] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[4] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.