Paper Replication Report

Shuying Wang

The Concrete Distribution: a Continuous Relaxation of Discrete Random Variables

By Chris J. Maddison, Andriy Mnih, & Yee Whye The

DeepMind, London, United Kingdom

University of Oxford, Oxford, United Kingdom

Summary

This paper aims to find a practical way for auto differentiation (AD) to deal with the discrete random variables in neural networks. The authors introduced a new family of distributions: concrete distribution, which is the abbreviation of "continuous relaxations of discrete random variables". The concrete methods solve the problem that discrete random variables lack appropriate reparameterizations, so that enable the application of reparameterization tricks to SCGs with discrete stochastic nodes. In this way, we can take gradients backwards the neural networks and optimize the parameters by gradient descent. The authors tested the effectiveness of this method by VAE architectural neural networks with discrete latent random variables, and they substitute all the discrete stochastic nodes with concrete ones during training. As a result, they found that concrete approximation can be used effectively in the neural networks with discrete stochastic nodes.

Key words: reparameterization trick, concrete distribution, neural network, VAE model,

Theoretical Basis

Reparameterization Trick

Reparameterization trick is used to get low variance unbiased estimators of the gradients of parameters in stochastic computation graphs. The main idea of this trick is to refactor a random variable into a differentiable function of its parameters and a random variable with fixed distribution.

Figure 1: an example of SCG.

For example, in a simple SCG with a Normal distributed stochastic node:

$$x|\phi \sim \text{Normal}(\mu, \sigma^2)$$

We cannot directly get the gradient of ϕ, because the objective function $L(\theta, \phi) = E_{X\sim p_\phi}[f_\theta(x)]$ is an expectation depend on ϕ. We can use methods like importance sampling, but the estimator of the gradient may suffer from high variance. To use reparameterization trick, we can refactor x as a function of its parameters and a standard normal distribution,

$$x = g_\phi(z) = \sigma z + \mu$$

so that, we can express the objective function as an expectation with respect to z:

$$\nabla_\phi L(\theta, \phi) = E_{Z\sim q(z)}\left[\nabla_\phi f_\theta\left(g_\phi(Z)\right)\right] = E_{Z\sim q(z)}\left[f_\theta{}'\left(g_\phi(Z)\right)\nabla_\phi g_\phi(Z)\right]$$

In this way, we can get the low variance, unbiased estimator of the gradient.

Concrete distribution

For a discrete random variable $D \sim \text{Discrete}(\alpha)$, where $\alpha \in (0, \infty)^n$, $D \in \{0, 1\}^n$, $\sum_{k=1}^{n} D_k = 1$, and $P(D_k = 1) = \frac{\alpha_k}{\sum_{i=1}^{n} \alpha_i}$, it can be reparametrized as an argmax function of parameter α and standard Gumbel random variables G_k (Gumbel – Max trick). However, the derivative of the argmax function is either 0 or undefined, so it cannot be used in the SCGs with auto differentiation. Therefore, the authors introduced the concrete distribution, which relaxes the state of a discrete random variable from the vertices to the interior of the space. The concrete random variable $x \sim$ Concrete (α, λ) can also be refactored as a function of parameters and standard Gumbel random variables G_k:

$$X_k = \frac{\exp((\log \alpha_k + G_k)/\lambda)}{\sum_{i=1}^{n} \exp((\log \alpha_i + G_i)/\lambda)}$$

The density function of the concrete distribution is:

$$p_{\alpha,\lambda}(x) = (n-1)!\, \lambda^{n-1} \prod_{k=1}^{n} \left(\frac{\alpha_k x_k^{-\lambda-1}}{\sum_{i=1}^{n} \alpha_i x_i^{-\lambda}} \right)$$

Where $x \in (0,1)$, $\alpha \in (0, \infty)^n$, and another parameter $\lambda \in (0, \infty)$ is called the temperature of the concrete distribution. The concrete distribution has several nice properties so that it can be used to approximate the discrete distributions. Firstly, each value of X_i is between 0 and 1, and the sum of X_i's is 1. Secondly, $P(X_k > X_i \; for \; i \neq k) = \alpha_k / (\sum_{i=1}^{n} \alpha_i)$, which means the probability of X_k being the maximum value among X_i's is the same as the probability of D_k being 1. Thirdly, $P(\lim_{\lambda \to 0} X_k = 1) = \alpha_k / (\sum_{i=1}^{n} \alpha_i)$, so the concrete distribution converges to its corresponding discrete ones (with the same parameter α) as λ goes to 0. Besides, $p_{\alpha,\lambda}(x)$ is log – convex in x, if $\lambda \leq (n-1)^{-1}$.

Experiment

To evaluate the effectiveness of the gradients of concrete relaxations for optimizing SCGs with discrete nodes, the authors used datasets with 28×28 images of handwritten digits (MNIST) and letters (Omniglot). The Concrete density is used during training in place of the discrete mass. At test time, the graph with discrete nodes is evaluated. They conducted two experimental tasks: the structured output prediction and density estimation. They compared their results with VIMCO for optimizing the multi-sample variational objective (m > 1, IWAE model) and NVIL for optimizing the single-sample one (m = 1, VAE model).

The experiment results in this paper shows that the concrete relaxation is an effective way to deal with discrete stochastic nodes in neural networks, and it has a better performance in non-linear models compared to other methods.

Replication

Protocol

I replicate the density estimation and generative model task with the MNIST dataset. The VAE (Variational Auto-encoder) generative model I choose is denoted by (200H ~ 784V) in the paper, which means when generate simulated data, first independently sample 200 latent variables $z \sim p_\theta(z)$, and then sample 784 data points $x \sim p_\theta(x|z)$. The "~" in (200H ~ 784V) means that the transformation $\theta(z)$ to compute parameters for sampling x is non-linear.

VAE Model

The true posterior $p_\theta(z|x)$ is typically intractable, so we will use an approximated $q_\alpha(z|x)$ during training. We want to maximize the log likelihood and at the same time minimize the KL divergence between $q_\alpha(z|x)$ and $p_\theta(z|x)$, so we are going to maximize the Evidence lower bound (ELBO) as the objective in training model.

We assume that the stochastic latent variables in our generative model are discrete random variables, $D \sim$ Discrete (α), and we replace it with the corresponding concrete random variables $Z \sim$ Concrete (α), so the objective function ELBO is:

$$\mathcal{L}_1(\theta, a, \alpha) \overset{\text{relax}}{\rightsquigarrow} \mathop{\mathbb{E}}_{Z \sim q_{\alpha,\lambda_1}(z|x)} \left[\log \frac{p_\theta(x|Z) p_{a,\lambda_2}(Z)}{q_{\alpha,\lambda_1}(Z|x)} \right]$$

Where a and θ are the true parameters, α is the variational parameter of the approximated posterior, which depends on the input x. During the training, we compute α through a non-linear transformation $\alpha(x)$. While $q_{\alpha,\lambda_1}(z\,|x)$ is the relaxed concrete distribution of the posterior, $p_{a,\lambda_2}(z)$ is the relaxed concrete distribution of the prior. Temperatures λ_1 and λ_2 are hyper-parameters fixed during the training

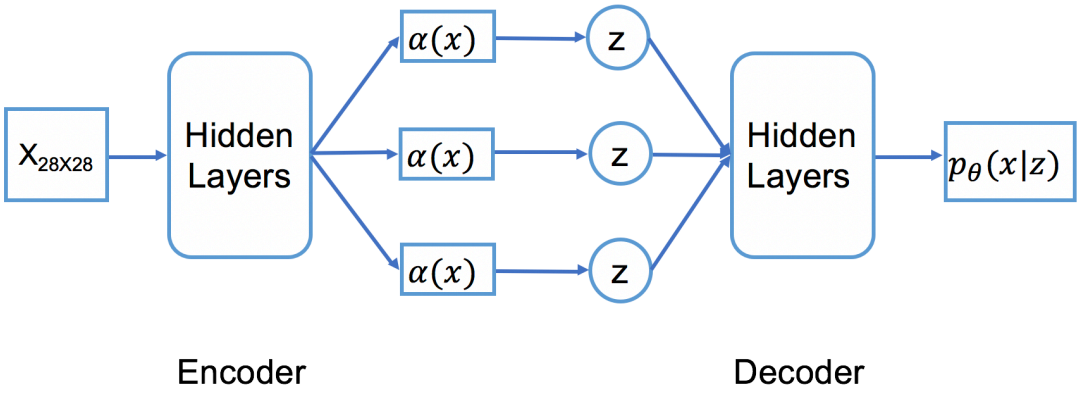

Figure 2: a simple illustration of VAE model. There are 200 latent variable nodes in the actual model

Concrete Reparameterization

To take the gradients of parameters by reparameterization trick, we sample z's by first sample Gumbel random variables:

$$\nabla_\alpha \mathcal{L}(\theta, a, \alpha) = E_{G\sim Gumbel}\left[\nabla_\alpha \log \frac{p_\theta(x|g(G))p_{a,\lambda_2}(g(G))}{q_{\alpha,\lambda_1}(g(G)|x)}\right]$$

$$\cong \frac{N}{M}\sum_{i=1}^{M}\left\{\frac{1}{L}\sum_{l=1}^{L}\left[\nabla_\alpha \log \frac{p_\theta(x_i|g(G_l))p_{a,\lambda_2}(g(G_l))}{q_{\alpha_i,\lambda_1}(g(G_l)|x_i)}\right]\right\}$$

Sample L G_l 's per datapoint x_i. Sample M x_i's per minibatch, and average over each minibatch to get the estimator of gradient for each step of gradient descent

Algorithm

Initialize parameters (θ, a, α)

Repeat

Random sample minibatch of M data points

Repeat

Input one data $X_{28\times 28}$

Pass through the 4 convolutional hidden layers to compute $\alpha(x)$

Random sample L Gumbel variables and compute $z = g_\alpha(G)$

Pass through the 4 convolutional hidden layers to compute $\theta(x)$ and ELBO.

Compute the gradient estimator $\nabla_\alpha \mathcal{L}(\theta, a, \alpha)$

Until input all the data in a minibatch

Update parameters using the gradient estimator of minibatch by one step

Until converge of parameters (θ, a, α)

Return (θ, a, α)

Results

I trained the (200H ~ 784V) VAE models with 200 binary or 4-ary discrete latent variables, compared with models with 100 bivariate normal latent variables and 100 binary or 4-ary discrete latent variables.

NLL	200 Discrete	100 Discrete & 100 Normal
binary	133.1747	128.7401
4-ary	127.4705	125.6739

Figure 3: NLL of models trained after 15 epochs

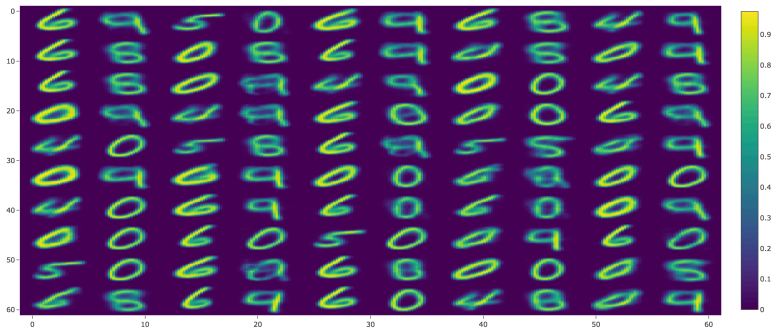

Figure 4: 200 binary discrete latent variables

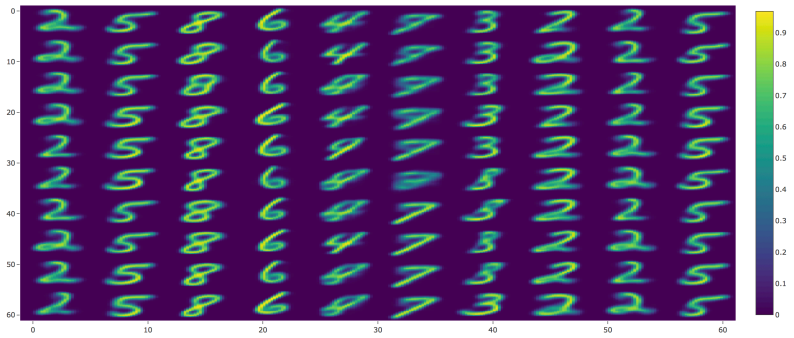

Figure 5: 100 binary discrete latent variables & 100 bivariate normal latent variables

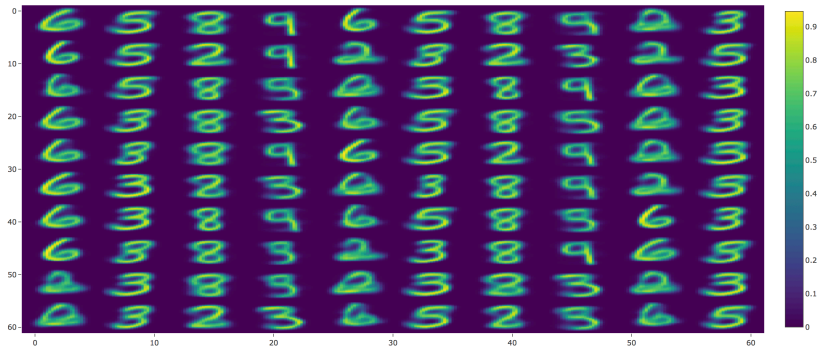

Figure 6: 200 4-ary discrete latent variables

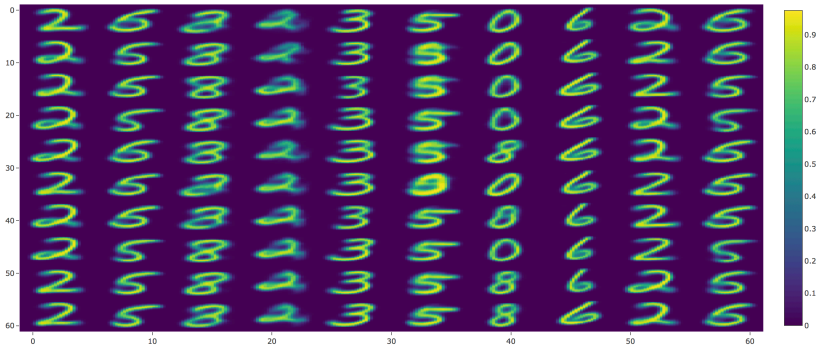

Figure 7: 100 4-ary discrete latent variables & 100 bivariate normal latent variables

Discussion

I was not able to generate results as good as the original paper, maybe due to the limited numbers of epochs trained. The optimization of parameters didn't reach convergence after 15 epochs of training. Most of the images generated by the (200H ~ 784V) VAE models can be recognized as a handwritten digit, but some of them are still vague. Generally the 4-ary models outperformed the binary models, while in the original paper there's no significant difference between this two kinds of models. The 100 discrete & 100 normal 4-ary model performs similarly to the 200 discrete 4-ary model, while the 100 discrete & 100 normal model is better in the binary case.

References

EmilienDupont. (2017, September 22). EmilienDupont/vae-concrete. Retrieved from https://github.com/EmilienDupont/vae-concrete

Kingma, P, D., & Max. (2014, May 01). Auto-Encoding Variational Bayes. Retrieved from https://arxiv.org/abs/1312.6114

Maddison, C. J., Mnih, A., & The, Y. W. (2017). The Concrete distribution: A continuous relaxation of discrete random variables. *ICLR*, 2017.

Schulman, John, Nicolas, Weber, Abbeel, & Pieter. (2016, January 05). Gradient Estimation Using Stochastic Computation Graphs. Retrieved from https://arxiv.org/abs/1506.05254