Auto-encoding Variational Bayes

Franco Caspe

I. Introduction

Variational inference allows the estimation of optimal parameters in probabilistic models for which an exact marginalization cannot be computed due their complexity, by approximating distributions that can be factorized across a set of parametrized distributions.

A variational auto-encoder is a machine learning algorithm that models probability distributions on train data by extracting their parameters from two neural networks that are connected in series. The first model called encoder, receives the input to the auto-encoder and transforms it to a latent representation. In general terms, a latent representation contains all the extracted information of interest from the input. The second model, called decoder, provides the output to the auto-encoder; it receives a latent representation and performs a reconstruction in the domain of the input.

The decoder model provides the parameters to a generative distribution $p(\mathbf{x}|\mathbf{z})$, where $\mathbf{x}$ belongs to the input/output space and $\mathbf{z}$ to the latent representation domain.

In this work, we use variational inference to obtain an optimal set of parameters for a generative distribution $p(\mathbf{x}|\mathbf{z})$ that transforms latent representations that follow a normal distribution across L dimensions, $p(\mathbf{z}) = N(0, 1, I)$ into images of the form $\mathbf{x} \in \{0, 1\}^D$. Furthermore, $\mathbf{z}$ can be sampled from its distribution in order to generate any image. The probabilistic graphical model is shown in Figure 1.

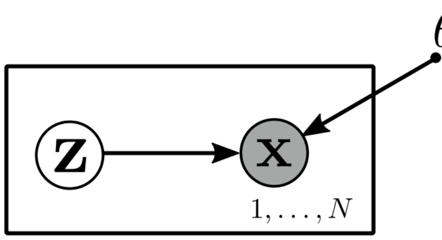

Fig. 1. Graphical model of the generative distribution.

An auto-encoder model is proposed to approximate the distribution and generate the optimal set of parameters.

II. Algorithm Description

The task of interest is to sample $\mathbf{z}$ from its distribution and use the generative model to generate the images, provided that $\mathbf{z}$ has a dimensionality large enough to encode sufficient information.

The generative distribution is modeled as a product of D Bernoulli random variables with activation probability equals to $\sigma(f_j^\theta(\mathbf{z}))$ for $j = 1, ..., D$, where $\sigma(.)$ is the sigmoid activation function and θ are the parameters of the neural network that is represented by the $f(.)$ function, that complies with the decoder of the auto-encoder model.

The estimation of generative distribution's parameters θ cannot be done using Maximum Likelihood Estimation because $p_\theta(\mathbf{X})$ that would maximize the likelihood over a data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \in \{0, 1\}^D$ is untractable due the non linearities of $p(\mathbf{x}|\mathbf{z})$.

Variational inference is then used to approximate the best decoder's parameters by maximizing the lower bound of the log marginal likelihood of our data set, approximating it to $log\ p_\theta(X)$.

The optimization target is shown in Equation 1, on which $q_\phi(\mathbf{z}|\mathbf{x})$ is a parametrized distribution that it is modeled to factorize across multiple Gaussian distributions, and should approximate $p(\mathbf{z}|\mathbf{x})$ to maximize the lower bound.

$$O(\theta, \phi) = \sum_{i=1}^{N} E_{q_\phi(\mathbf{z}|\mathbf{x})}[log\ p_\theta(\mathbf{x}_i|\mathbf{z})] - KL(q_\phi(\mathbf{z}|\mathbf{x}_i)|P(\mathbf{z})) \tag{1}$$

The $q_\phi(\mathbf{z}|\mathbf{x})$ distribution's mean and variance will be given by the output of another neural network: $\mu_j^\phi(\mathbf{z})$ and $\nu_j^\phi(\mathbf{z})$ for $j = 1, ..., L$, respectively. This neural network complies with the encoder part of the auto-encoder model.

Both sets of parameters θ and ϕ can be found at the same time by training both neural networks on a provided data set. This is done by employing stochastic gradient descent on the target, over mini batchs using the Autograd[1] library and the ADAM[2] optimizer.

III. Tasks carried out and results

In this section the tasks carried out and the results obtained are enumerated and discussed.

A. Training Loop Completion

The first two tasks were related to the completion of the training process of the neural networks by means of the implementation of both the optimization target calculation and the training loop employing the ADAM optimizer.

The optimization target was implemented on the function *vae_lower_bound()*, that receives the mini batch and outputs the lower bound estimation on the mini batch. This functionality is performed on three stages.

The first stage, implemented in the function *sample_latent_variables_from_posterior()* encodes every instance b of the mini batch into a particular distribution

$q_\phi(\mathbf{z}|x_b)$ and then performs one sampling for each distribution (Monte Carlo approach) generating latent representations.

The second stage, uses the generative model to generate images from the latent representations and then computes the log probability of the output given the data, in the function *bernoulli_log_prob()*.

The third stage, *compute_KL()* calculates the KL divergence that will be subtracted from the log probability to obtain an estimate of the lower bound by averaging across the mini batch.

Finally, in the *main* section of the Python script, the training loop over mini batchs was completed by implementing the ADAM optimizer, for maximizing the target using the noisy approximation of the gradient of the target computed from the mini batch.

The maximization for the lower bound across 30 training epochs is shown in Figure 2.

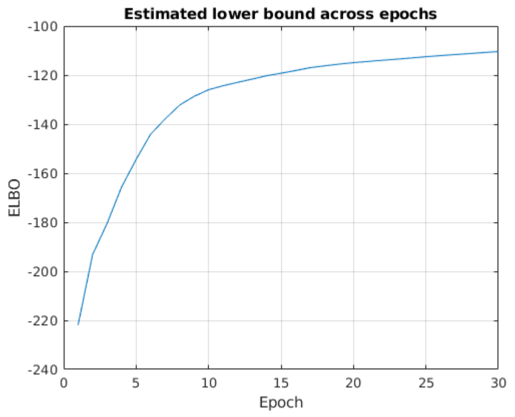

Fig. 2. Evolution of the Estimated Lower Bound.

B. Image generation using the generative model

The next subsection describes the set of tests that were conducted in order to evaluate the performance of the generative model obtained from training the auto encoder. To obtain the graphical representations from the generative model, a sigmoid activation function was applied to the output of the decoder, obtaining a probability image, or probability map for each pixel, as shown in Equation 2. The images are plotted using gray scale, with intensities from 0 to 1.

$$\sigma(f_j^\theta(\mathbf{z}_b)) = p(x_j = 1|\mathbf{z}_b)\ j = 1, .., D; b = 1, ..., 25. \quad (2)$$

The first test consisted on drawing samples from the latent representation distribution, $p(\mathbf{z})$. 25 latent representations were sampled from the normal distribution and used as the input of the decoder network. The results are shown in Figure 3.

The second test consisted on assessing the representation capabilities of the auto-encoder by inputting a set of images on the encoder network, to obtain for each instance the parameters of the factors of the distribution $p(\mathbf{z}|\mathbf{x})$. The means of the factors then was used as latent representation to

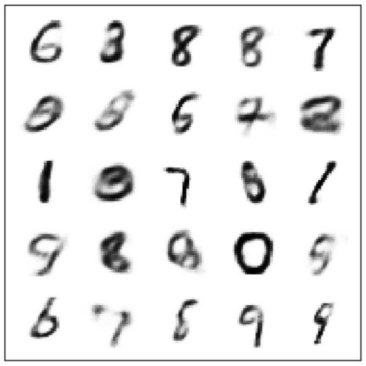

Fig. 3. Images sampled from random prior.

re-synthesize the images as close as possible to the originals using the decoder. The results are shown in Figure 4 on which 10 original images are shown above the respective 10 reconstructions from probability maps.

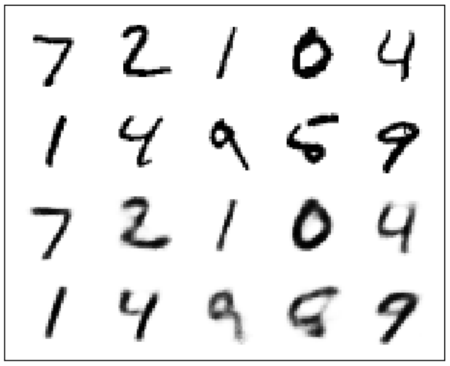

Fig. 4. Original and reconstructed images.

The final test consisted on perform a graphical evaluation of the probability maps generated on ambiguous latent variables. It was performed by firstly generating optimal latent representations by processing pairs of images through the encoder as in the previous test, obtaining distributions from which the mean of latent variables was taken.

The pair of optimal latent representations was then linearly interpolated on 25 steps, generating different linear combinations of the vectors, with different weights for the first and the second image in the pair. Five pairs were sampled from the test set. The output probability images are shown in Figure 5 and 6, on which it can be seen that there exist an increase in the grey areas of the output probability maps, depicting a pixel probability close to 0.5. This signifies a decrease in the confidence of the output due the ambiguity of the latent representation.

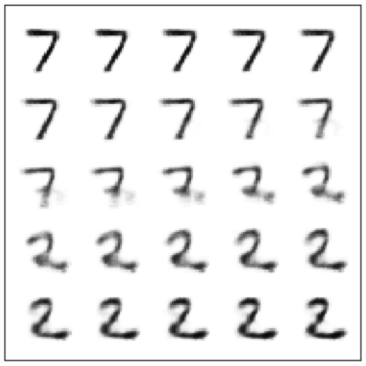
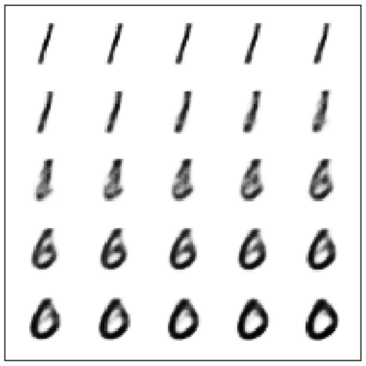
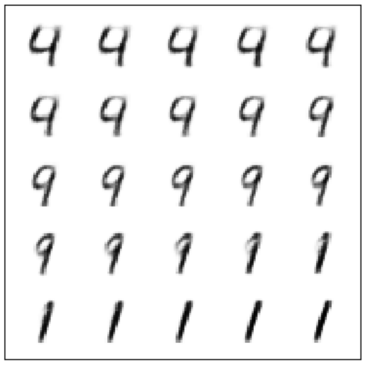

Fig. 5. Reconstructed images from interpolated latent representations. Test pairs 1 to 3.

IV. Conclusions

A non-linear, generative model was successfully optimized by employing variational inference to obtain a set of optimal parameters to generate images from a random prior. The inference process maximizes the lower bound of the marginal distribution, increasing the likelihood of the distribution and the train data, proving successful to train the auto-encoder.

When assessing the performance of the generator, the best results were obtained when using the encoder and extracting

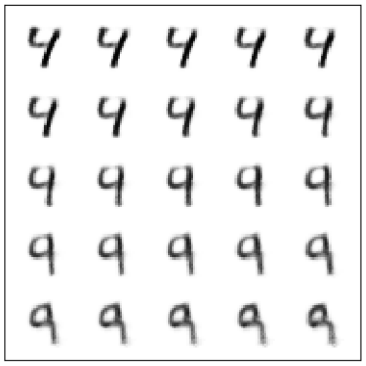
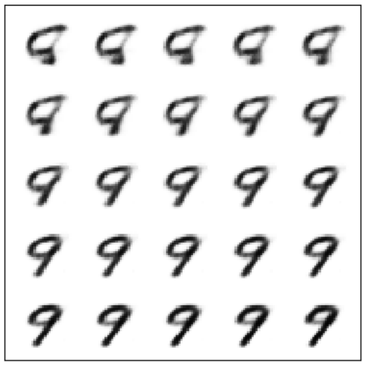

Fig. 6. Reconstructed images from interpolated latent representations. Test pairs 4 and 5.

the mean of the factors, while the worst results were obtained when generating ambiguous latent representations in the interpolation test.

In this sense, it seems that a random number image of some degree of quality can be directly obtained from the prior distribution; this approach can be employed for data generation purposes. Nevertheless, there exist optimal latent representations, generated by the encoder, from which the decoder network shows a high level of quality and certainty on the output map. Hence, the auto-encoder configuration can be employed for data compression.

References

[1] Autograd: Efficiently computes derivatives of numpy code. Available at: `https://github.com/HIPS/autograd`

[2] Kingma, D. P., & Ba, J. Adam: A method for stochastic optimization, International Conference on Learning Representations, 2014.