Adversarial Variational Bayes in Edward

Karttikeya Mangalam, 14311
Shubh Gupta, 14670
Ankur Kumar, 14109

Course Project
CS698X - Topics in Probabilistic Modeling and Inference
Prof. Piyush Rai

29 avril 2018

Overview

- Generative Adversarial Networks & Variational Auto-encoders are leading generative models both of which are probabilistic in nature.
- GANs produce sharper image with better visual quality
- VAEs produce a blurry images but have attractive mathematical bounds

FIGURE – Visual output from VAEs [1]

1. Tulyakov et al. ICLR 2017

Adversarial Variational Auto-encoders

- Attempts to combine best of both worlds
- Makzani et al. proposed Adversarial Auto-encoder (AAE)
  - Doesn't lead to MLE assignments in general
  - Don't optimize a lower bound on maximum likelihood
- Mescheder et al. propose an alternate approach : Adversarial VAE
  - Optimize ELBO lower bound to maximum likelihood
  - AAEs can be interpreted as an approximation
  - Uses 2-player game like GANs to circumvent evaluating likelihood

Framework

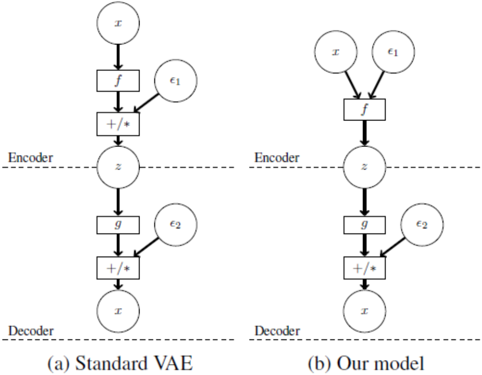


FIGURE – The vanilla VAE model and the modification proposed by Mescheder et al. VAEs are specified by a parametric generative model $p(x_j|z)$ of the visible variables given the latent variables, a prior $p(z)$ over the latent variables and an approximate inference model $q(z_j|x)$ over the latent variables given the visible variables.

ELBO Formulation I

- Standard Evidence Lower bound for Variational Auto-encoders

$$\log p_\theta(x) \geq -\mathrm{KL}(q_\phi(z \mid x), p(z)) + \mathrm{E}_{q_\phi(z|x)} \log p_\theta(x \mid z)$$

- Our Goal is to then solve the following optimization problem

$$\max_\theta \max_\phi \mathrm{E}_{p_{\mathcal{D}}(x)} \mathrm{E}_{q_\phi(z|x)} \big(\log p(z) - \log q_\phi(z \mid x) + \log p_\theta(x \mid z) \big)$$

ELBO Formulation II

- Rewriting ELBO as

$$\max_\theta \max_\phi \mathrm{E}_{p_{\mathcal{D}}(x)} \mathrm{E}_{q_\phi(z|x)} \Big(\boxed{\log p(z)} \\ \boxed{- \log q_\phi(z \mid x)} + \log p_\theta(x \mid z) \Big).$$

- The indicated terms can be re-written as the optimal value of an additional real-valued discriminative network $T(x; z)$ defined as

$$\max_T \mathrm{E}_{p_{\mathcal{D}}(x)} \mathrm{E}_{q_\phi(z|x)} \log \sigma(T(x, z)) \\ + \mathrm{E}_{p_{\mathcal{D}}(x)} \mathrm{E}_{p(z)} \log (1 - \sigma(T(x, z)))$$

Can be thought of as a two player min max game similar to GANs.

ELBO Formulation III

Thus we have the optimal value $T^*(x; z)$ is obtained as

$$T^*(x, z) = \log q_\phi(z \mid x) - \log p(z)$$

$$\max_{\theta,\phi} \mathrm{E}_{p_{\mathcal{D}}(x)} \mathrm{E}_{q_\phi(z|x)} \big(-T^*(x, z) + \log p_\theta(x \mid z) \big)$$

Now what remains is to find the gradients of the above loss with respect to θ and ϕ that can be used in any first order gradient optimizer. Here we employ the result,

$$\mathrm{E}_{q_\phi(z|x)} \left(\nabla_\phi T^*(x, z) \right) = 0.$$

And taking gradient for θ is straightforward.

Algorithm

Now, reparameterization trick is used to rewrite the ELBO loss as

$$\max_{\theta,\phi} \mathrm{E}_{p_{\mathcal{D}}(x)} \mathrm{E}_{\epsilon} \Big(-T^{*}(x, z_{\phi}(x,\epsilon)) + \log p_{\theta}(x \mid z_{\phi}(x,\epsilon)) \Big)$$

Thus, the parameters can be updated as :

$$g_{\theta} \leftarrow \tfrac{1}{m} \textstyle\sum_{k=1}^{m} \nabla_{\theta} \log p_{\theta} \left(x^{(k)} \mid z_{\phi} \left(x^{(k)}, \epsilon^{(k)} \right) \right)$$

$$g_{\phi} \leftarrow \tfrac{1}{m} \textstyle\sum_{k=1}^{m} \nabla_{\phi} \Big[-T_{\psi} \left(x^{(k)}, z_{\phi}(x^{(k)}, \epsilon^{(k)}) \right) + \log p_{\theta} \left(x^{(k)} \mid z_{\phi}(x^{(k)}, \epsilon^{(k)}) \right) \Big]$$

$$g_{\psi} \leftarrow \tfrac{1}{m} \textstyle\sum_{k=1}^{m} \nabla_{\psi} \Big[\log \left(\sigma(T_{\psi}(x^{(k)}, z_{\phi}(x^{(k)}, \epsilon^{(k)}))) \right) + \log \left(1 - \sigma(T_{\psi}(x^{(k)}, z^{(k)})) \right) \Big]$$

Implementation Details

- Prior over latent variable z is Normal Distribution and the generated data x fits a Bernoulli distribution with output of generative network as logits.

Implementation Details

- Prior over latent variable z is Normal Distribution and the generated data x fits a Bernoulli distribution with output of generative network as logits.
- Inference network is modeled by adding Gaussian noise *eps* to observed data.

Implementation Details

- Prior over latent variable z is Normal Distribution and the generated data x fits a Bernoulli distribution with output of generative network as logits.
- Inference network is modeled by adding Gaussian noise *eps* to observed data.
- The parameters of variational distribution of z (a Normal distribution) is found by optimizing the previous ELBO formulation.

Implementation Details

- Prior over latent variable z is Normal Distribution and the generated data x fits a Bernoulli distribution with output of generative network as logits.
- Inference network is modeled by adding Gaussian noise *eps* to observed data.
- The parameters of variational distribution of z (a Normal distribution) is found by optimizing the previous ELBO formulation.
- Optimizing ELBO with respect to ϕ while keeping θ and T fixed makes the encoder network collapse to a deterministic f.

Implementation Details

- Prior over latent variable z is Normal Distribution and the generated data x fits a Bernoulli distribution with output of generative network as logits.
- Inference network is modeled by adding Gaussian noise *eps* to observed data.
- The parameters of variational distribution of z (a Normal distribution) is found by optimizing the previous ELBO formulation.
- Optimizing ELBO with respect to ϕ while keeping θ and T fixed makes the encoder network collapse to a deterministic f.
- Perform several SGD-updates for the adversary for one SGD-update of the generative network.

Implementation Details

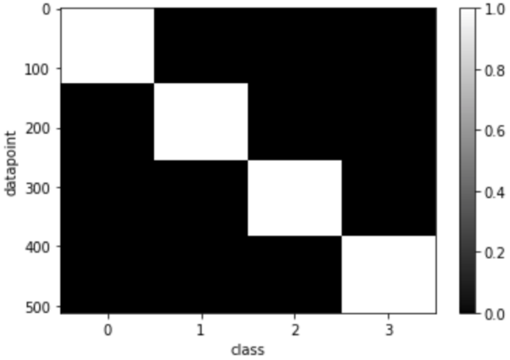
- Prior over latent variable z is Normal Distribution and the generated data x fits a Bernoulli distribution with output of generative network as logits.
- Inference network is modeled by adding Gaussian noise *eps* to observed data.
- The parameters of variational distribution of z (a Normal distribution) is found by optimizing the previous ELBO formulation.
- Optimizing ELBO with respect to ϕ while keeping θ and T fixed makes the encoder network collapse to a deterministic f.
- Perform several SGD-updates for the adversary for one SGD-update of the generative network.
- Inference, Generator and Discriminator network uses fully connected layers.

Implementation Details

- Prior over latent variable z is Normal Distribution and the generated data x fits a Bernoulli distribution with output of generative network as logits.
- Inference network is modeled by adding Gaussian noise *eps* to observed data.
- The parameters of variational distribution of z (a Normal distribution) is found by optimizing the previous ELBO formulation.
- Optimizing ELBO with respect to ϕ while keeping θ and T fixed makes the encoder network collapse to a deterministic f.
- Perform several SGD-updates for the adversary for one SGD-update of the generative network.
- Inference, Generator and Discriminator network uses fully connected layers.
- ADAM optimizer for Discriminative network, SGD for Generative Network.

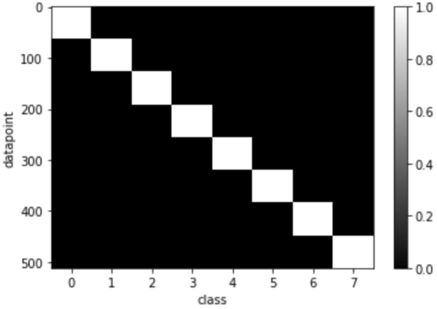
Synthetic Dataset I

- Synthetic data generated as one-hot vectors of feature size 4, to be encoded into 2 dimensional latent variables.



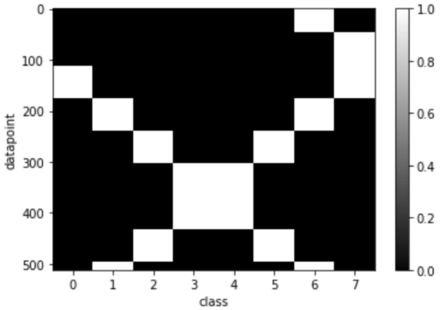
Synthetic Dataset II

- Synthetic data generated as one-hot vectors of feature size 8 with 2 dimensional latent space visualization



Synthetic Dataset III

- Synthetic data generated as
  binarized features of feature size 8 with 2 dimensional latent space visualization



MNIST Experiment I

- Visualization of 2 dimensional embeddings over binarized MNIST dataset.

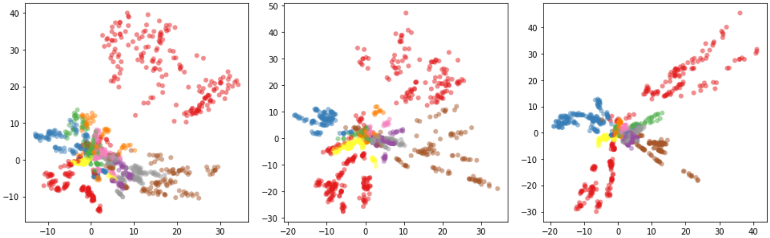
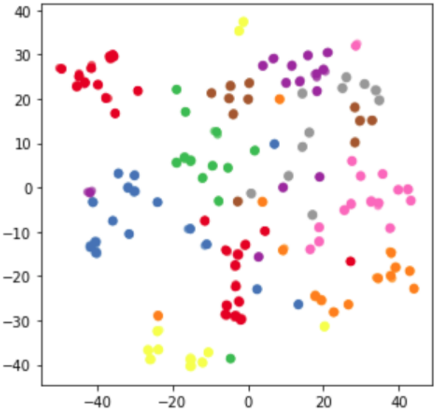


FIGURE – a) 500 iterations b) 2000 iterations c) 5000 iterations

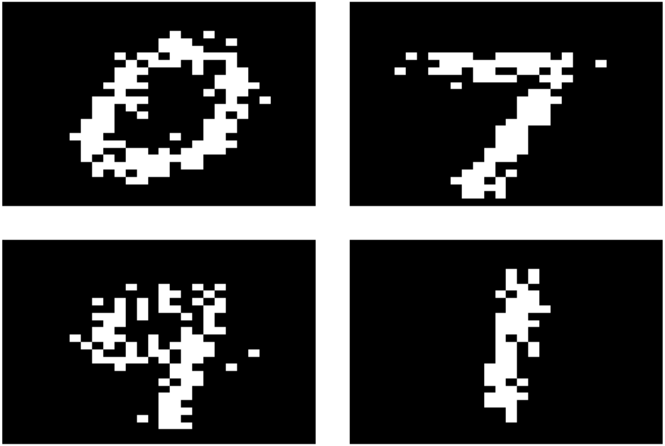
MNIST Experiment II

- T-SNE visualization over 10 dimensional latent space embeddings of binarized MNIST



MNIST Experiment III

- Sampled digits from our implemented model on MNIST



CelebA Dataset

- Samples from results of original paper on CelebA dataset.

Log likelihoods

	$\log p(x) \geq$	$\log p(x) \approx$
AVB (8-dim)	$(\approx -83.6 \pm 0.4)$	-91.2 ± 0.6
AVB + AC (8-dim)	$\approx -96.3 \pm 0.4$	-89.6 ± 0.6
EDWARD AVB	$\approx -96.7 \pm 0.5$	-
VAE (8-dim)	-98.1 ± 0.5	-90.9 ± 0.6

TABLE – Log likelihood values for different approaches on binarized MNIST dataset.

- Our implementation is competitive although the slight loss in performance is due to using few fully connected layers instead of deep convolutional layers for discriminator network.

Conclusion

- Programming Probabilistic models is easier in Edward as compared to other non-probabilistic frameworks with lesser lines of code.
- Training time taken for Edward is more than other frameworks, mostly because of high level abstraction and hence less customization.
- GPU utilization for Edward programs is more than that for other probabilistic frameworks, since it uses Tensorflow's computation capabilities.
- Edward is still in development phase, and hence many models, such as LDA, are hard to train, even if modeling is easier.