Variational Auto-Encoder and generative Zoo

Machine Learning and Data Mining

Maxim Borisyak

National Research University Higher School of Economics (HSE)

November 29, 2017

Generative models

Generative models

- Informally, given samples we wish to learn generative procedure.
- Formally, given samples of a random variable X, we wish to find X', so that:

$$P(X) \approx P(X')$$

In the previous episodes

RBM:

- Maximum Likelihood fit through energy function;
- Gibbs sampling;

GAN:

- transformation from a tractable random variable to a target $X = f(Z)$;
- minimizes Jensen-Shannon distance which estimated via classifier;
- direct sampling.

Variational Auto-Encoder

Latent variables revisited

Before generating a sample, model should first decide what it should generate:

- which digit to generate: 0, 1, ..., 9
- width of stokes;
- 'speed';
- etc.

Such decision can be represented as random variables, often called **latent variables**.

Like most of the generative models, VAE searches random variable as a function of *latent variables* Z:

- easy to sample;
- tractable distribution.

Most common choice $Z \sim \mathcal{N}^n(0, 1)$. Unlike GAN, transformation $Z \to X$ is **not deterministic**:

$$\begin{aligned} P(X) &= \int P(X \mid z)P(z)dz; \\ P(X|z) &= \mathcal{N}(X \mid f(z), \sigma^2 I). \end{aligned}$$

Latent variables revisited

$$P(X) = \int P(X \mid z)P(z)dz = \mathop{\mathbb{E}}_{z} P(X \mid z)$$

- sampling from $P(X)$ might be computationally expensive;
- most of z produce $P(X \mid z) \approx 0$.

Variational bound

$$P(X) = \int P(X \mid z) P(z) dz = \mathop{\mathbb{E}}_{Z} P(X \mid Z)$$

In order to make sampling tractable, P(Z) can be replaced by some $Q(Z \mid X)$.

$$\log P(X) = \mathrm{KL}\left(Q(Z \mid X) \,\|\, P(Z \mid X)\right) + \mathcal{L}(X) \geq \mathcal{L}(X)$$

$$\begin{aligned}\mathcal{L}(X) = & \mathop{\mathbb{E}}_{Z \sim Q}\left[-\log Q(Z \mid X) + \log P(X, Z)\right] = \\ & -\mathrm{KL}\left(Q(Z \mid X) \,\|\, P(Z)\right) + \mathop{\mathbb{E}}_{z \sim Q(Z \mid X)}\left[\log P(X \mid Z)\right]\end{aligned}$$

VAE objective

$$\log P(X) - \mathrm{KL}\left(Q(Z \mid X) \,\|\, P(Z \mid X)\right) = \underset{z \sim Q(Z|X)}{\mathbb{E}} \log P(X \mid Z) - \mathrm{KL}\left(Q(Z \mid X) \,\|\, P(Z)\right)$$

where:

- $\mathrm{KL}\left(Q(Z \mid X) \,\|\, P(Z \mid X)\right)$ - error term:
  - recognition model penalty;
- $\mathbb{E}_{z \sim Q(Z|X)}\left[\log P(X \mid Z)\right]$ - reconstruction error:
  - can be estimated like in an ordinary AE;
- $\mathrm{KL}\left(Q(Z \mid X) \,\|\, P(Z)\right)$ - something similar to regularization;
  - can be computed analytically if $P(Z)$ is well defined.

Reconstruction error

$$\mathrm{RE} = \mathop{\mathbb{E}}_{z \sim Q(Z|X)} [\log P(X \mid Z)]$$

- for Gaussian posterior i.e. $P(X \mid Z) = \mathcal{N}(X \mid f(z), \sigma^2 I)$:

$$\mathrm{RE} = \frac{1}{2} E_{Z \sim Q(Z|X)} \left[f(Z) - X \right]^2 + \mathrm{const}$$

- for Benulli posterior (e.g. for discrete output) $P(X = 1 \mid Z) = f(z)$:

$$\mathrm{RE} = E_{Z \sim Q(Z|X)} \left[X \log f(Z) + (1 - X) \log(1 - f(Z)) \right]$$

The other term

$$\mathrm{KL}\left(Q(Z \mid X) \,\|\, P(Z)\right)$$

Consider:

- $Q(Z \mid X) = \mathcal{N}(Z \mid \mu(X), \Sigma(X))$;
- $P(Z) = \mathcal{N}(0, I)$

$$\begin{aligned}
\mathrm{KL}\left(\mathcal{N}(X \mid f(Z), \Sigma(Z)) \,\|\, \mathcal{N}(X \mid f(Z), \Sigma(Z))\right) = \\
\frac{1}{2}\big(\mathrm{tr}(\Sigma(X)) + \|\mu(X)\|^2 - k - \log\det\Sigma(X)\big) = \\
\frac{1}{2}\left(\|\mu(X)\|^2 + \sum_i \Sigma_{ii}(X) - \log\Sigma_{ii}(X)\right) - \frac{k}{2}
\end{aligned}$$

Training time

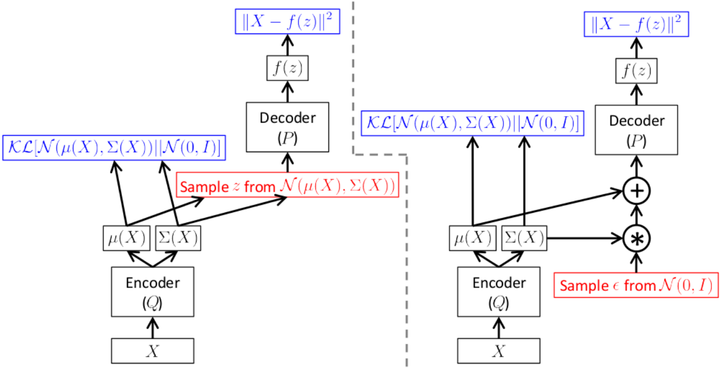


Figure 4: A training-time variational autoencoder implemented as a feedforward neural network, where $P(X|z)$ is Gaussian. Left is without the "reparameterization trick", and right is with it. Red shows sampling operations that are non-differentiable. Blue shows loss layers. The feedforward behavior of these networks is identical, but backpropagation can be applied only to the right network.

Testing time

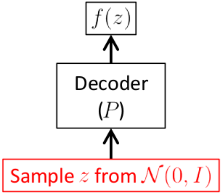


Figure 5: The testing-time variational "autoencoder," which allows us to generate new samples. The "encoder" pathway is simply discarded.

Conditional VAE

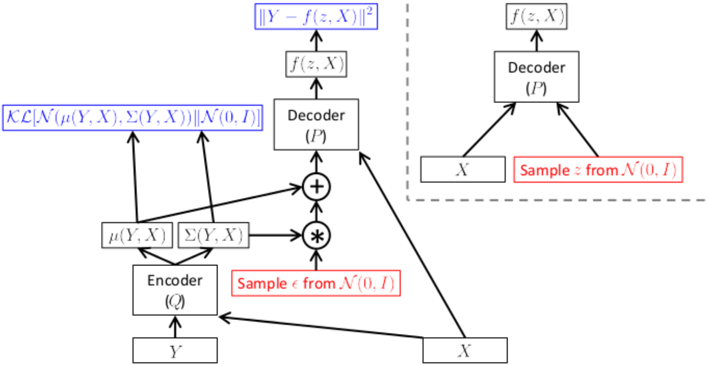


Figure 6: Left: a training-time conditional variational autoencoder implemented as a feedforward neural network, following the same notation as Figure 4. Right: the same model at test time, when we want to sample from $P(Y|X)$.

Discussion

VAE:

- simple to implement;
- stable training;
- fast training;
- form of inference and generative network impose serious restrictions on possible models;
- might be tricky to modify.

GAN:

- might be tricky to implement;
- training requires tricks (or EB/W-GAN);
- training is slow;
- often, produces state-of-the-art models;
- flexible architecture.

BiGAN

The main difference between GAN and VAE is absence of an inference model in GAN ($Q(Z|X)$ in VAE). BiGAN adds this inference by introducing an inference networks:

$$X' = G(Z);$$
$$Z' = E(X);$$

and solving min-max problem for:

$$\mathcal{L}(D, E, G) \quad = \quad \mathop{\mathbb{E}}_{x} \log D(x, E(x)) \;+\; \mathop{\mathbb{E}}_{z} \log(1 \;-\; D(G(z), z))$$

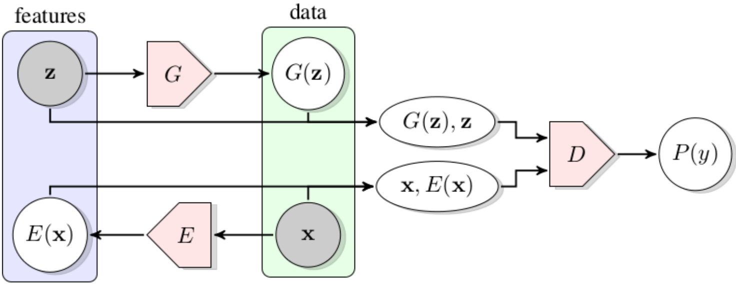


Figure 1: The structure of Bidirectional Generative Adversarial Networks (BiGAN).

BiGAN

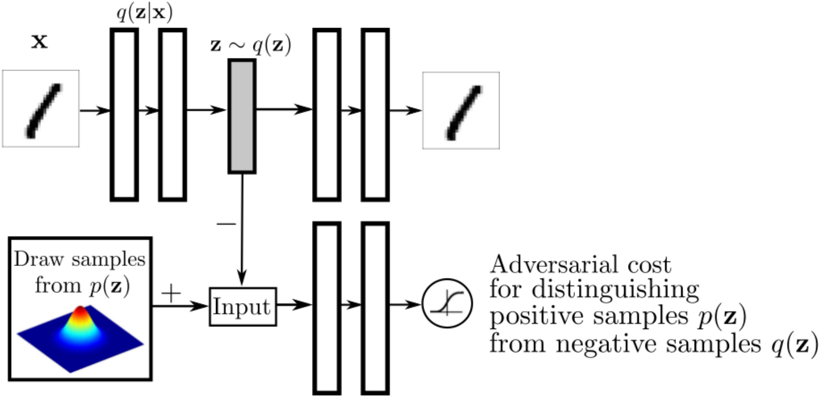
- BiGAN can be used as dimensionality reduction technique;
- unlike AE, importance of a feature produced not by Euclidean distance, but by a discriminator.

Adversarial AutoEncoder

Adversarial AutoEncoder

- replaces VAE regularization tern with GAN objective;
- also makes both inference and generative networks deterministic:
    - removes possible error source;
- essentially, AutoEncoder with regularization term that drives code space to a predefined distribution.

Adversarial AutoEncoder



Adversarial AutoEncoder

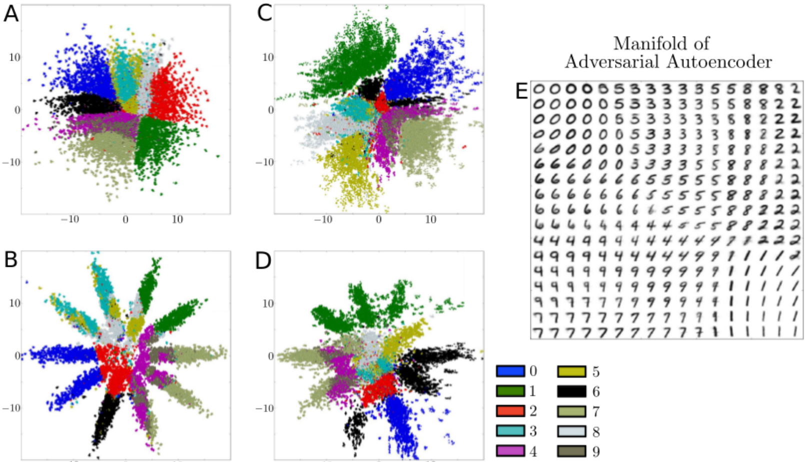


Figure 2: Comparison of adversarial and variational autoencoder on MNIST. The hidden code **z** of the *hold-out* images for an adversarial autoencoder fit to (a) a 2-D Gaussian and (b) a mixture of 10 2-D Gaussians. Each color represents the associated label. Same for variational autoencoder with (c) a 2-D gaussian and (d) a mixture of 10 2-D Gaussians. (e) Images generated by uniformly sampling the Gaussian percentiles along each hidden code dimension **z** in the 2-D Gaussian adversarial autoencoder.

Codes can be associated with known high level features, e.g. face expression.

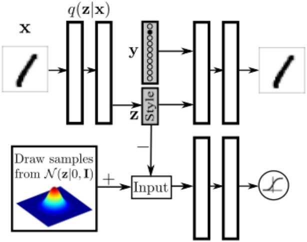


Figure 6: Disentangling the label information from the hidden code by providing the one-hot vector to the generative model. The hidden code in this case learns to represent the style of the image.

Adversarial AutoEncoder

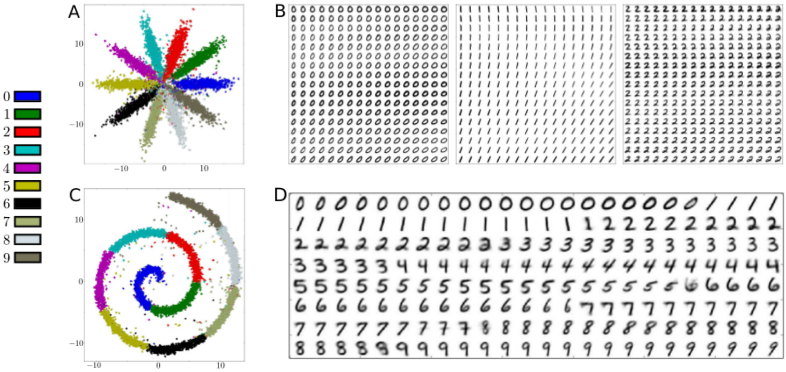


Figure 4: Leveraging label information to better regularize the hidden code. **Top Row:** Training the coding space to match a mixture of 10 2-D Gaussians: (a) Coding space **z** of the *hold-out* images. (b) The manifold of the first 3 mixture components: each panel includes images generated by uniformly sampling the Gaussian percentiles along the axes of the corresponding mixture component. **Bottom Row:** Same but for a swiss roll distribution (see text). Note that labels are mapped in a numeric order (i.e., the first 10% of swiss roll is assigned to digit 0 and so on): (c) Coding space **z** of the *hold-out* images. (d) Samples generated by walking along the main swiss roll axis.

Adversarial Variational Bayes

Adversarial Variational Bayes

Adversarial Variational Bayes replaces inference model $Q(Z \mid X)$ by a black box:

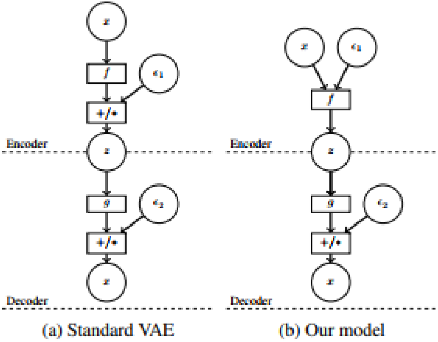


Figure 2. Schematic comparison of a standard VAE and a VAE with black-box inference model, where ϵ_1 and ϵ_2 denote samples from some noise distribution. While more complicated inference models for Variational Autoencoders are possible, they are usually not as flexible as our black-box approach.

$$\mathcal{L}(\theta, \phi) =$$

$$\mathop{\mathbb{E}}_{x}\left[-\mathrm{KL}(q_\phi(z \mid x) \,\|\, p(z)) + \mathop{\mathbb{E}}_{q_\phi(z|x)} \log p_\theta(x \mid z)\right] =$$

$$\mathop{\mathbb{E}}_{x} \mathop{\mathbb{E}}_{q_\phi(z|x)} \left[\log p(z) - \log q_\phi(z \mid x) + \log p_\theta(x \mid z)\right]$$

Solution for the optimization problem:

$$\max_T \mathop{\mathbb{E}}_{x} \mathop{\mathbb{E}}_{q_\phi(z|x)} \log \sigma(T(x,z)) \quad + \quad \mathop{\mathbb{E}}_{x} \mathop{\mathbb{E}}_{p(z)} \log(1 \quad - \quad \sigma(T(x,z)))$$

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

$$\boxed{\mathcal{L}(\theta, \phi) = \mathop{\mathbb{E}}_{x}\left[-T^*(x,z) + \log p_\theta(x \mid z)\right]}$$

Adversarial Variational Bayes

$$\text{AVB} = \text{VAE objective} + \\ \text{black box inference} + \\ \text{adversarial estimation of KL};$$

Summary

Summary

Varitional AutoEncoder:

- AutoEncoder that drives code space to match predefined distribution;
- often inferior to GAN;
- stable and relatively fast training;
- various adversarial modifications.

References I

- Doersch, C., 2016. Tutorial on variational autoencoders. arXiv preprint arXiv:1606.05908.
- Kingma, D.P. and Welling, M., 2013. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.
- Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O. and Courville, A., 2016. Adversarially learned inference. arXiv preprint arXiv:1606.00704.

References II

- Donahue, J., Krähenbühl, P. and Darrell, T., 2016. Adversarial feature learning. arXiv preprint arXiv:1605.09782.
- Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. and Frey, B., 2015. Adversarial autoencoders. arXiv preprint arXiv:1511.05644.
- Mescheder, L., Nowozin, S. and Geiger, A., 2017. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. arXiv preprint arXiv:1701.04722.