Variational AutoEncoder

Jacob Goldberger

Generative model (decoder):

$$y \sim N(0, I)$$
$$x|y \sim N(\mu_\theta(y), \Sigma_\theta(y))$$

$\mu_\theta(y), \Sigma_\theta(y)$ are computed by a NN with input y and parameters θ.

Approximate posterior inference (encoder):

- $\mu_\lambda(x), \Sigma_\lambda(x)$ are computed by a NN with parameters λ.

$$q(y|x;\lambda) = N(y; \mu_\lambda(x), \Sigma_\lambda(x))$$

- hard decision: $\hat{y} = \arg\max_y q(y|x;\lambda) = \mu_\lambda(x)$.
- Estimating the probability of an object x:

$$\log p(x) \geq \frac{1}{k}\sum_{i=1}^{k} \log p(x|y_i;\theta) - KL(N(\mu_\lambda(x), \Sigma_\lambda(x))||N(0,I))$$

where $y_i|x$ is sampled according to $N(\mu_\lambda(x), \Sigma_\lambda(x))$.

Learning the model parameters:

$$\log p(x;\theta) = \log \sum_y p(y,x;\theta) = ELBO(\theta,\lambda) + KL(q(y|x;\lambda)||p(y|x;\theta))$$

$ELBO(\theta,\lambda)$ is the Evidence Lower BOund (or negative free energy):

$$\begin{aligned} ELBO(\theta,\lambda) &= \sum_y q(y|x;\lambda)\log p(x,y;\theta) - \sum_y q(y|x;\lambda)\log q(y|x;\lambda) \\ &= \sum_y q(y|x;\lambda)\log p(x|y;\theta) - KL(q(y|x;\lambda)||p(y;\theta)) \end{aligned}$$

Variational EM:

$$\hat{\theta} = \arg\max_\theta \log p(x;\theta) = \arg\max_\theta \max_\lambda ELBO(\theta,\lambda) \geq \arg\max_\theta \max_{\lambda\in\Lambda} ELBO(\theta,\lambda)$$

E-step: Find $\lambda \in \Lambda$ that maximizes $ELBO(\theta,\lambda)$.
M-step: Find θ that maximizes $\sum_y q(y|x;\lambda)\log p(x,y;\theta)$.

Deep Learning for Variational EM: we directly optimize the ELBO:

$$\begin{aligned} ELBO(\theta,\lambda) &= E_{q(y|x;\lambda)} \log p(x|y;\theta) - KL(q(y|x;\lambda)||p(y;\theta)) \\ &= E_{q(y|x;\lambda)} \log N(x;\mu_\theta(y),\Sigma_\theta(y)) - KL(N(\mu_\lambda(x),\Sigma_\lambda(x))||N(0,I)) \end{aligned}$$

The second term has a closed-form solution:

$$KL(N(\mu, \sigma^2)||N(0,1)) = \frac{1}{2}(\mu^2 + \sigma^2 - \log \sigma^2).$$

We use sampling to compute the first term:

$$E_{q(y|x;\lambda)} \log N(x; \mu_\theta(y), \Sigma_\theta(y)) \approx \log N(x; \mu_\theta(y), \Sigma_\theta(y))$$

such that $y|x$ is sampled according to $N(\mu_\lambda(x), \Sigma_\lambda(x))$.

Training Score:

- $x \to \mu_\lambda(x), \Sigma_\lambda(x) \to y \sim N(\mu_\lambda(x), \Sigma_\lambda(x))$

  $$\to \mu_\theta(y), \Sigma_\theta(y) \to p(x|y;\theta) = N(x; \mu_\theta(y), \Sigma_\theta(y))$$

- $S = \log N(x; \mu_\theta(y), \Sigma_\theta(y)) - KL(N(\mu_\lambda(x), \Sigma_\lambda(x))||N(0, I))$

Computing the gradient using the score function estimation:

$$\nabla_\lambda E_{p(y;\lambda)}[f(x,y)] = \int \nabla_\lambda p(y;\lambda) f(x,y) dy \int p(y;\lambda) \nabla_\lambda (\log p(y;\lambda)) f(xy) dy$$

$$= E_{p(y;\lambda)}[f(x,y) \nabla_\lambda (\log p(y;\lambda))] \approx f(x,y) \nabla_\lambda (\log p(y;\lambda))$$

such that $y|x$ is sampled according to $N(\mu_\lambda(x), \Sigma_\lambda(x))$. This estimator has a very high variance. The key contribution of the VAE paper is to propose an alternative estimator that is much better behaved based on the so-called reparametrization trick.

Reparameterization trick of Variational Autoencoder: We can write $q_\lambda(y|x)$ as following:

$$y = g_\lambda(\epsilon, x) = \mu_\lambda(x)\epsilon + \Sigma_\lambda(x)$$

s.t. $\epsilon \sim p(\epsilon) = N(0,1)$

$$\nabla_\lambda E_{y \sim p(y;\lambda)}[f(x,y)] = \nabla_\lambda E_{\epsilon \sim p(e)}[f(x, g(\epsilon, x)] = E_{\epsilon \sim p(e)} \nabla_\lambda [f(x, g(\epsilon, x))]$$

$$\approx \nabla_\lambda [f(x, g(\epsilon_0, x))]$$

such that ϵ_0 ia sampled from $N(0, I)$ and $y = \mu_\lambda(x)\epsilon_0 + \Sigma_\lambda(x)$. After sampling ϵ_0 the function is deterministic and we can backpropagate through it:

$$x \to \mu_\lambda(x), \Sigma_\lambda(x) \to y = \mu_\lambda(x)\epsilon_0 + \Sigma_\lambda(x) \to \mu_\theta(y), \Sigma_\theta(y) \to p(x|y;\theta)$$