Computer Vision I : Project 5 (10 points)

Experiments with Generative Model and Descriptive Model

Due on Dec 22th (Thursday) 11:59pm

Objective. The objective of this project is to experience the integration of the descriptive model as we studied in project 3 and the generative model as we worked in project 4. The prototypes of the descriptive model is the exponential family distribution, such as the FRAME model. The prototype of the top-down generative model is the factor analysis, which has been generalized to the multi-layer generator network, whose expressive power comes from the top-down network that maps a simple prior to be close to the data distribution. In project 4, we assume the low-dimensional prior distribution of the latent factors follow a given simple prior, such as the isotropic Gaussian distribution or the uniform distribution. In this project, following the empirical Bayes philosophy, we learn a descriptive model from empirical observations. we assume the latent vector follows a descriptive model serving as the prior model that stands on the top-down network of the generator model. Both the latent space descriptive model and the top-down network can be learned jointly by the maximum likelihood, which involves short-run MCMC sampling from both the prior and posterior distributions of the latent vector.

1 Introduction to the Generative Model with the Flexible Prior

Notations. Let I be a D-dimensional observed example, such as an image. Let z be the d-dimensional vector of continuous latent factors, $z = (z_k, k = 1, ..., d)$. The joint distribution of (I, z) is

$$p_\theta(I, z) = p_\alpha(z) p_\beta(I|z), \tag{1}$$

where $p_\alpha(z)$ is the prior model with parameters α, $p_\beta(I|z)$ is the top-down generation model with parameters β, and $\theta = (\alpha, \beta)$.

The prior model $p_\alpha(z)$ is formulated as a descriptive model or an energy-based model,

$$p_\alpha(z) = \frac{1}{Z(\alpha)} \exp(f_\alpha(z)) p_0(z). \tag{2}$$

where $p_0(z)$ is a reference distribution, assumed to be isotropic Gaussian white noise distribution or the uniform distribution. $f_\alpha(z)$ is the negative energy and is parameterized by a small multi-layer perceptron with parameters α. $Z(\alpha) = \int \exp(f_\alpha(z)) p_0(z) dz = \mathrm{E}_{p_0}[\exp(f_\alpha(z))]$ is the normalizing constant or partition function.

The generation model is the same as the top-down network in the generator model. For image modeling,

$$I = g_\beta(z) + \epsilon, \tag{3}$$

where $\epsilon \sim \mathrm{N}(0, \sigma^2 I_D)$, so that $p_\beta(I|z) \sim \mathrm{N}(g_\beta(z), \sigma^2 I_D)$. Usually σ^2 takes an assumed value. The marginal distribution is $p_\theta(I) = \int p_\theta(I, z)dz = \int p_\alpha(z)p_\beta(I|z)dz$.

For the training data $\{I_i, i = 1, ..., n\}$, the generator model can be trained by maximizing the log-likelihood

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p(I_i; \theta) \doteq \mathrm{E}_{p_{data}}[\log p_\theta(x)]. \tag{4}$$

The learning gradient can be calculated according to

$$\nabla_\theta \log p_\theta(I) = \mathrm{E}_{p_\theta(z|I)}\left[\nabla_\theta \log p_\theta(I, z)\right] = \mathrm{E}_{p_\theta(z|I)}\left[\nabla_\theta(\log p_\alpha(z) + \log p_\beta(I|z))\right]. \tag{5}$$

For the prior model, $\nabla_\alpha \log p_\alpha(z) = \nabla_\alpha f_\alpha(z) - \mathrm{E}_{p_\alpha(z)}[\nabla_\alpha f_\alpha(z)]$. Thus the learning gradient for an example I is

$$\delta_\alpha(I) = \nabla_\alpha \log p_\theta(I) = \mathrm{E}_{p_\theta(z|I)}[\nabla_\alpha f_\alpha(z)]^+ - \mathrm{E}_{p_\alpha(z)}[\nabla_\alpha f_\alpha(z)]^-. \tag{6}$$

For the generation model,

$$\delta_\beta(I) = \nabla_\beta \log p_\theta(I) = \mathrm{E}_{p_\theta(z|I)}[\nabla_\beta \log p_\beta(I|z)], \tag{7}$$

where $\log p_\beta(I|z) = -\|I - g_\beta(z)\|^2/(2\sigma^2) + \text{constant}$, which is about the reconstruction error. Writing $\delta_\theta(I) = (\delta_\alpha(I), \delta_\beta(I))$, we have $L'(\theta) = \mathrm{E}_{p_{data}}[\delta_\theta(I)]$, and the learning algorithm is $\theta_{t+1} = \theta_t + \eta_t \mathrm{E}_{p_{data}}[\delta_{\theta_t}(I)]$.

Expectations in (6) and (7) require MCMC sampling of the prior model $p_\alpha(z)$ and the posterior distribution $p_\theta(z|I)$. We can use Langevin dynamics. For a target distribution $\pi(z)$, the dynamics iterates

$$z_{k+1} = z_k + s\nabla_z \log \pi(z_k) + \sqrt{2s}e_k, \tag{8}$$

where k indexes the time step of the Langevin dynamics, s is a small step size, and $e_k \sim \mathrm{N}(0, I_d)$ is the Gaussian white noise. $\pi(z)$ can be either $p_\alpha(z)$ or $p_\theta(z|x)$. In either case, $\nabla_z \log \pi(z)$ can be efficiently computed by back-propagation.

Algorithm 1: Learning the generator with latent-space descriptive model prior.

```
input  : Learning iterations $T$, learning rate for prior model $\eta_0$, learning rate for generation
         model $\eta_1$, initial parameters $\theta_0 = (\alpha_0, \beta_0)$, observed examples $\{I_i\}_{i=1}^n$, batch
         size $m$, number of prior and posterior sampling steps $\{K_0, K_1\}$, and prior and
         posterior sampling step sizes $\{s_0, s_1\}$.
output: $\theta_T = (\alpha_T, \beta_T)$.
for $t = 0 : T - 1$ do
    1. Mini-batch: Sample observed examples $\{I_i\}_{i=1}^m$.
    2. Prior sampling: For each $I_i$, sample $z_i^- \sim \tilde{p}_{\alpha_t}(z)$ using equation (8), where the
       target distribution $\pi(z) = p_{\alpha_t}(z)$, and $s = s_0$, $K = K_0$.
    3. Posterior sampling: For each $x_i$, sample $z_i^+ \sim \tilde{p}_{\theta_t}(z|I_i)$ using equation (8), where
       the target distribution $\pi(z) = p_{\theta_t}(z|x_i)$, and $s = s_1$, $K = K_1$.
    4. Learning prior model: $\alpha_{t+1} = \alpha_t + \eta_0 \frac{1}{m} \sum_{i=1}^m [\nabla_\alpha f_{\alpha_t}(z_i^+) - \nabla_\alpha f_{\alpha_t}(z_i^-)]$.
    5. Learning generation model: $\beta_{t+1} = \beta_t + \eta_1 \frac{1}{m} \sum_{i=1}^m \nabla_\beta \log p_{\beta_t}(I_i|z_i^+)$.
```

2 Experiment

For the lion-tiger images, in Proj-4 you have learnt a generative model with 2-*dim* latent vector. In this project, you can learn a high-dimensional latent vector, e.g., 32-*dim*, whose distribution follows the descriptive model in Eq.(2). To reduce the computational complexity, all images can be resized into the size of 128×128 pixels.

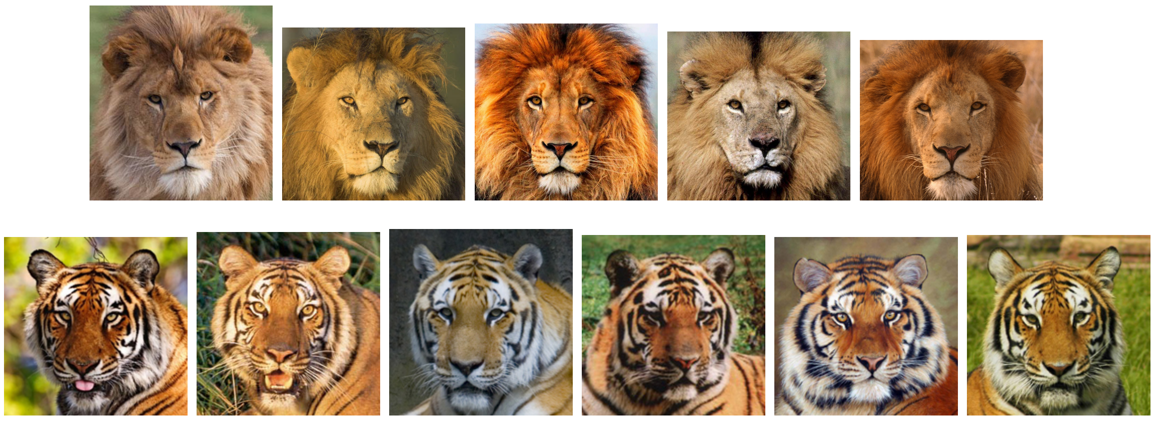

Figure 1: lion-tiger images.

Fill the blank part of ./code/generator_prior.py. Design your own structure of both the generator and the descriptor. To generate vivid results, you should adjust the learning rate, descretization step size of the Langevin dynamics, the number of langevin steps, and the annealing or tempering parameter σ^2 in the generator. Show:

(1) Reconstructed images of training images, using the inferred z from training images. For the posterior sampling, you can try both the warm-start scheme (running a finite number of steps of Langevin dynamics starting from the current value of z_i) and cold-start scheme (running a finite number of steps of Langevin dynamics starting from the fixed prior of the latent factors.)

(2) Randomly generated images, using randomly sampled z from the latent-space descriptive model.

(3) Comparing the quality of the generated results from the prior of the latent-space descriptor with the generated results from the Gaussian prior as in Proj-4. For each comparison, you can try the latent vector with dimensions 2, 8, and 32, to study the power of the descriptive model to correct the independence or conditional independence assumptions of the simple prior such as isotropic Gaussian distribution.

(4) Generated images with linearly interpolating two dimensions from the latent factors with (-d_1, d_2). For example, you inperlolate 10 points from (-d_1, d_2) for each dimension of z. Then you will get a 10×10 panel of images. Detemine the reasonable range (-d_1, d_2), and you should be able to see that tigers slight change to lion. With the high-dimensional latent vector, could you find novel semantic factors against that from 2-dim latent vector?

(5) Plot of loss and both the negative and positive energy of over iteration.

Write a report to show your results. And zip the report with your code.