CSE 599i (Generative Models) Homework 2

Analysis

Problem 1. (KL-Divergence between Gaussians) Let $\mu : \mathcal{X} \to \mathbb{R}^k$, $\sigma : \mathcal{X} \to \mathbb{R}^k$ and let $q(z|x) = \mathcal{N}\left(z; \mu(x), \text{diag}(\sigma^2(x))\right)$. Suppose that $p(z) = \mathcal{N}(z; 0, I)$. Show that

$$D(q(z|x) \parallel p(z)) = \frac{1}{2}\sum_{i=1}^{k}\left(\sigma_i^2(x) + \mu(x)_i^2 - 1 - \log \sigma_i^2(x)\right).$$

Problem 2. (Marginal Likelihood Estimation) Consider a latent variable model over pairs (x, z) where x is observed and z is latent, with a marginal likelihood given by

$$p_\theta(x) = \int_{\mathcal{Z}} p_\theta(x|z) r(z)\, dz.$$

Given a proposal distribution $q(z|x)$, consider the importance sampling estimator

$$\widehat{\mathcal{L}}(x) \equiv \log \frac{1}{M}\sum_{i=1}^{M}\frac{p_\theta(x|z_i) r(z_i)}{q(z_i|x)}, \text{ where } z_i \sim q(\cdot|x).$$

Prove that $\widehat{\mathcal{L}}(x)$ is a biased estimator of $\log p_\theta(x)$, but is asymptotically unbiased as $M \to \infty$:

$$\mathop{\mathbb{E}}_{z_i \sim q(\cdot|x)}\left[\widehat{\mathcal{L}}(x)\right] \leq \log p_\theta(x), \text{ but } \lim_{M\to\infty} \widehat{\mathcal{L}}(x) = \log p_\theta(x).$$

Problem 3. (Normalizing Flows) Let q_0 be a probability distribution on $\mathcal{Z}$, and define $\mathbf{z}_s = g_s(\mathbf{z}_{t-1})$ where $g_s : \mathcal{Z} \to \mathcal{Z}$ are invertible functions. Prove that the pushforward distribution on $\mathbf{z}_t = g_t \circ \cdots \circ g_1(\mathbf{z}_0)$ is given by q_t, where

$$\log q_t(\mathbf{z}_t) = \log q_0(\mathbf{z}_0) - \sum_{s=1}^{t} \text{logdet}\left(\frac{\partial g_s(\mathbf{z}_{s-1})}{\partial \mathbf{z}_{s-1}}\right).$$

Hint: consider using the inverse function theorem.

Problem 4. (A Monte-Carlo Estimator for Entropy Gradients) Let $q_\varphi(z|x)$ be a family of conditional distributions with parameters φ. Show that

$$\nabla_\varphi H(q_\varphi(z|x)) = -\mathop{\mathbb{E}}_{z\sim q_\varphi(\cdot|x)}\left[\frac{(1 + \log q_\varphi(z|x))}{q_\varphi(z|x)}\nabla_\varphi q_\varphi(z|x)\right].$$

Hint: apply the policy gradient theorem (or mimic its proof).

Implementation

See the github repository for framework code. Turn in your versions of `models.py` and `losses.py`. Consider using smaller models to debug your code before the final run with default hyper-parameters.

Problem 5. (Gaussian Variational Autoencoders) In this problem you will train a Gaussian VAE for the MNIST dataset, using the training framework found in `gaussian_vae.ipynb`. This dataset consists of 28×28 grayscale images.

Part a. Implement the `ConvnetBlock` found in `models.py`.

Part b. Train a Gaussian VAE to minimize the negative evidence lower-bound:

$$\mathcal{L}(x; \theta, \varphi) = \mathop{\mathbb{E}}_{z \sim q_\varphi(\cdot|x)} \left[\frac{1}{2\sigma^2} \|x - g_\theta(z)\|^2 \right] + D(q_\varphi(z|x) \parallel r(z)).$$

Implement this lower bound as `gaussian_elbo` in `losses.py` using the closed-form expression for the KL divergence derived in Problem 1. Report visual samples from your optimized model.

Part c. Train a Gaussian VAE with using a monte carlo estimate of the KL divergence:

$$D(q_\varphi(z|x) \parallel r(z)) = \mathop{\mathbb{E}}_{z \sim q_\varphi(\cdot|x)} \left[\log \frac{q_\varphi(z|x)}{r(z)} \right].$$

Implement this lower bound as `mc_gaussian_elbo` in `losses.py`. Report the test-set value of the evidence lower bound. Compare your results to the results of Part b.

Problem 6. (VAE's with Discrete Outputs) In this problem you will train a Gaussian VAE with discrete outputs for the *binarized* MNIST dataset, using the training framework found in `binaryvae.ipynb`. This dataset consists of 28×28 black-and-white images derived from MNIST by sampling a value in $\{0, 1\}$ for each pixel value. In place of the MSE reconstruction term, use a binary cross-entropy loss in the evidence lower-bound:

$$\mathcal{L}(x; \theta, \varphi) = \mathop{\mathbb{E}}_{z \sim q_\varphi(\cdot|x)} \big[-\log p_\theta(x|z) \big] + D(q_\varphi(z|x) \parallel r(z)).$$

Part a. Implement `MaskedConvnetBlock` in `models.py` (analogous to `ConvnetBlock`, but using masked convolutions). Train a PixelCNN using `pixelcnn.ipynb`. Report test-set log-likelihood.

Part b. Implement the discrete-output ELBO as `discrete_output_elbo` in `losses.py`. Train a VAE using `binaryvae.ipynb` and report the test-set marginal log-likelihood estimated using importance sampling with $M = 1,000$ samples z_i for each data point x.

Part c. Set `autoregress = True` in `binaryvae.ipynb` to train a VAE Decoder that conditions on previously-generated pixels using the PixelCNN you built in Part a and the VAE Encoder from Part b. Compute the test set value of the ELBO after 20 epochs and compare to the marginal log-likelihood computed with importance sampling ($M = 1,000$). Plot the divergence in the ELBO as a function of training iterations: you should visibly observe posterior collapse during training.

Part d. Modify the VAE Encoder in `models.py` to incorporate 8 steps of Inverse Autoregressive Flow. Report the test-set marginal log-likelihood using importance sampling ($M = 1,000$).

Part e. (open ended) Train a PixelVAE by combining a PixelCNN Decoder (Part c) with an IAF Encoder (Part d). Can you prevent posterior collapse?