Due Date: April 29th 23:59, 2020

Instructions

- *Read all instructions and questions carefully before you begin.*
- *For all questions, show your work!*
- *The repository for this homework is* https://github.com/CW-Huang/IFT6135H20_assignment

Problem 1

Variational Autoencoders (VAEs) are probabilistic generative models to model data distribution $p(\boldsymbol{x})$. In this question, you will be asked to train a VAE on the *Binarised MNIST* dataset, using the negative ELBO loss as shown in class. Note that each pixel in this image dataset is binary: The pixel is either black or white, which means each datapoint (image) a collection of binary values. You have to model the likelihood $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, i.e. the decoder, as a product of bernoulli distributions.[1]

1. **(`unittest`, 4 pts)** Implement the function '`log_likelihood_bernoulli`' in '`q1_solution.py`' to compute the log-likelihood $\log p(\boldsymbol{x})$ for a given binary sample $\boldsymbol{x}$ and Bernoulli distribution $p(\boldsymbol{x})$. $p(\boldsymbol{x})$ will be parameterized by the mean of the distribution $p(\boldsymbol{x}=1)$, and this will be given as input for the function.
2. **(`unittest`, 4 pts)** Implement the function '`log_likelihood_normal`' in '`q1_solution.py`' to compute the log-likelihood $\log p(\boldsymbol{x})$ for a given float vector $\boldsymbol{x}$ and isotropic Normal distribution $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$. Note that $\boldsymbol{\mu}$ and $\log(\boldsymbol{\sigma}^2)$ will be given for Normal distributions.
3. **(`unittest`, 4 pts)** Implement the function '`log_mean_exp`' in '`q1_solution.py`' to compute the following equation[2] for each $\boldsymbol{y}_i$ in a given $Y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_i, \ldots \boldsymbol{y}_M\}$;
$$\log \frac{1}{K} \sum_{k=1}^{K} \exp\left(y_i^{(k)} - a_i\right) + a_i,$$
where $a_i = \max_k y_i^{(k)}$. Note that $\boldsymbol{y}_i = [y_i^{(1)}, y_i^{(2)}, \ldots, y_i^{(k)}, \ldots, y_i^{(K)}]$s.
4. **(`unittest`, 4 pts)** Implement the function '`kl_gaussian_gaussian_analytic`' in '`q1_solution.py`' to compute KL divergence $D_{\text{KL}}\left(q(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z})\right)$ via analytic solution for given p and q. Note that p and q are multivariate normal distributions with diagonal covariance.
5. **(`unittest`, 4 pts)** Implement the function '`kl_gaussian_gaussian_mc`' in '`q1_solution.py`' to compute KL diveregence $D_{\text{KL}}\left(q(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z})\right)$ by using Monte Carlo estimate for given p and q. Note that p and q are multivariate normal distributions with diagonal covariance.

[1]The binarized MNIST is not interchangeable with the MNIST dataset available on `torchvision`. So the data loader as well as dataset will be provided.

[2]This is a type of log-sum-exp trick to deal with numerical underflow issues: the generation of a number that is too small to be represented in the device meant to store it. For example, probabilities of pixels of image can get really small. For more details of numerical underflow in computing log-probability, see `http://blog.smola.org/post/987977550/log-probabilities-semirings-and-floating-point`.

6. **(report, 15 pts)** Consider a latent variable model $p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})dz$. The prior is define as $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_L)$ and $\boldsymbol{z} \in \mathbb{R}^L$. Train a VAE with a latent variable of 100-dimensions ($L = 100$). Use the provided network architecture and hyperparameters described in '`vae.ipynb`'[3]. Use ADAM with a learning rate of 3×10^{-4}, and train for 20 epochs. Evaluate the model on the validation set using the **ELBO**. Marks will neither be deducted nor awarded if you do not use the given architecture. Note that for this question you have to:

   (a) Train a model to achieve an average per-instance ELBO of ≥ -102 on the validation set, and report the ELBO of your model. The ELBO on validation is written as:

$$\frac{1}{|\mathcal{D}_{\text{valid}}|} \sum_{\boldsymbol{x}_i \in \mathcal{D}_{\text{valid}}} \mathcal{L}_{\text{ELBO}}(\boldsymbol{x}_i) \geq -102$$

   Feel free to modify the above hyperparameters (except the latent variable size) to ensure it works.

7. **(report, 15 pts)** Evaluate *log-likelihood* of the trained VAE models by using importance sampling, which was covered during the lecture. Use the codes described in '`vae.ipynb`'. The formula is reproduced here with additional details:

$$\log p(\boldsymbol{x} = \boldsymbol{x}_i) \approx \log \frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(\boldsymbol{x} = \boldsymbol{x}_i | \boldsymbol{z}_i^{(k)})\ p(\boldsymbol{z} = \boldsymbol{z}_i^{(k)})}{q_\phi(\boldsymbol{z} = \boldsymbol{z}_i^{(k)} | \boldsymbol{x}_i)}; \quad \text{for all } k\text{: } \boldsymbol{z}_i^{(k)} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x}_i)$$

   and $\boldsymbol{x}_i \in \mathcal{D}$.

   (a) Report your evaluations of the trained model on the test set using the log-likelihood estimate ($\frac{1}{N}\sum_{i=1}^{N} \log p(\boldsymbol{x}_i)$), where N is the size of the test dataset. Use $K = 200$ as the number of importance samples, D as the dimension of the input ($D = 784$ in the case of MNIST), and $L = 100$ as the dimension of the latent variable.

Problem 2

Generative Adversarial Network (GAN) enables the estimation of distributional measure between arbitrary empirical distributions. In this question, you will first implement a function to estimate the Squared Hellinger as well as one to estimate the Earth mover distance. This will allow you to look at and contrast some properties of the f-divergence [4] and the Earth-Mover distance [5].

We provide samplers [6] to generate the different distributions that you will need for this question. In the same repository, we also provide the architecture of a neural network function Critic : $\mathcal{X} \rightarrow \mathbb{R}$ s.t. $\mathcal{X} \subset \mathbb{R}^2$ in model.py. For training, you may use SGD with a learning rate of $1e-3$ and a mini batch size of 512.

[3]This file is executable in Google Colab. You can also convert `vae.ipynb` to `vae.py` using the Colab.
[4]Relevant reference on f-divergence: https://arxiv.org/abs/1606.00709
[5]Relevant reference on Wasserstein GAN: https://arxiv.org/abs/1701.07875
[6]See the assignment repository https://github.com/CW-Huang/IFT6135H20_assignment

1. **(`report`, 4 pts)** Provide the objective function of the Squared Hellinger in your report (See Nowozin et al. – Footnote 4).

2. **(`unittest`, 4 pts)** Implement the function '`vf_squared_hellinger`' in '`q2_solution.py`' to compute the objective function for estimating the Squared Hellinger distance. Please, consider the definition given in Nowozin. We give more instruction in the code template.

3. **(`report`, 4 pts)** In your report, provide the objective function of the Wasserstein distance and the objective function of the "*Lipschitz Penalty*" [7]

4. **(`unittest`, 4 pts)** Implement the functions '`vf_wasserstein_distance`' and '`lp_reg`' in '`q2_solution.py`' to compute the objective function of the Wasserstein distance and compute the "*Lipschitz Penalty*". Consider that the norm used in the regularizer is the $L2$-norm.

5. **(`report`, 10 pts)** Let $u \sim U[0,1]$ be the uniform random variable with support in the interval $[0,1]$. We define p the distribution of $(0,u) \in \mathbb{R}^2$ and q the distribution of $(\theta,u) \in \mathbb{R}^2$ (We provide a function generating the p and q in the file q2_samplers.py). Plot the estimated Squared Hellinger distance between p and q and the estimated Earth-Mover distance between p and q for $\theta \in [0,2]$ with interval of 0.1 (i.e. 21 points). The x-axis should be the value of θ and the y-axis should be your estimate. In your report, provide two plots: a plot of the estimate of the Squared Hellinger with respect to θ and a plot of the estimate of the Wasserstein distance with respect to θ. Also, provide a one or two lines explanation of the behaviour you observe.

Problem 3

Recent years have shown an explosion of research into using deep learning and computer vision algorithms to generate images.

In this final question, you will use the GAN framework train a generator to generate a distribution of high dimensional images $\mathcal{X} \subset \mathbb{R}^{32\times32\times3}$, namely the **Street View House Numbers** dataset (SVHN) [8]. We will consider the prior distribution $p(z) = \mathcal{N}(\mathbf{0}, I)$ the isotropic gaussian distribution. We provide a function for sampling from the SVHN datasets in 'q3_samplers.py'.

Hyperparameters & Training Pointers We provide code for the GANs network as well as the hyperparameters you should be using. We ask you to code the training procedure to train the GANs as well as the qualitative exploration that you will include in your report. You can re-use the WGAN-lp objective you wrote in the the previous question.

[7] See Section 5 of https://arxiv.org/pdf/1709.08894.pdf
[8] The SVHN dataset can be downloaded at http://ufldl.stanford.edu/housenumbers/

Qualitative Evaluation In your report,

1. **(`report`, 8 pts) Provide visual samples.** Comment the quality of the samples from each model (e.g. blurriness, diversity).

2. **(`report`, 8 pts) We want to see if the model has learned a disentangled representation in the latent space.** Sample a random z from your prior distribution. Make small perturbations to your sample z for *each dimension* (e.g. for a dimension i, $z'_i = z_i + \epsilon$). ϵ has to be large enough to see some visual difference. For each dimension, observe if the changes result in visual variations (that means variations in $g(z)$). You do not have to show all dimensions, just a couple that result in interesting changes.

3. **(`report`, 8 pts) Compare between interpolating in the data space and in the latent space.** Pick two random points z_0 and z_1 in the latent space sampled from the prior.

   (a) For $\alpha = 0, 0.1, 0.2 \ldots 1$ compute $z'_\alpha = \alpha z_0 + (1-\alpha) z_1$ and plot the resulting samples $x'_\alpha = g(z'_\alpha)$.

   (b) Using the data samples $x_0 = g(z_0)$ and $x_1 = g(z_1)$ and for $\alpha = 0, 0.1, 0.2 \ldots 1$ plot the samples $\hat{x}_\alpha = \alpha x_0 + (1-\alpha) x_1$.

   Explain the difference with the two schemes to interpolate between images.