ECE 685D HW4

Deliverability: Please upload your Jupyter Notebook file (.ipynb file) with all outputs and necessary derivations exported as a PDF or HTML to Sakai. All necessary coding documentation can be found in `https://pytorch.org/docs/stable/index.html`. **We highly encourage you to submit your derivation in LaTeX. If you choose to submit a hand-written derivation, please make sure your derivation is legible. If you have hand-written solutions, please scan/take a photo and insert them into their designated space.** You can follow this tutorial to insert images to .ipynb files: `https://www.geeksforgeeks.org/insert-image-in-a-jupyter-notebook/`

1 Problem 1: Gaussian-Bernoulli Restricted Boltzmann Machines (50 pts)

Let the energy function of Gaussian-Bernoulli RBM take the form:

$$-\left(\sum_i \sum_j W_{ij} h_j \frac{v_i}{\sigma_i} - \sum_i \frac{(v_i - b_i)^2}{2\sigma_i^2} + \sum_j \alpha_j h_j\right).$$

1.1 (a)

Using the definition of $p(\boldsymbol{v}, \boldsymbol{h}) = \exp(-E(\boldsymbol{v}, \boldsymbol{h}))/Z$, derive $p(v_i = x|\boldsymbol{h})$ and $p(h_j = 1|\boldsymbol{v})$. You may leave $p(v_i = x|\boldsymbol{h})$ in an integral form once you cancel the terms involving $\sum_j \alpha_j h_j$.

Hint: First apply the Bayes rule, then simplify.

1.2 (b)

Train a Gaussian-Bernouilli RBM over the Fashion MNIST dataset using contrastive divergence minimization (see the lecture notes). Use the standard training and testing split available in Pytorch. Use learning rate 0.001 and batch size 128. Train the model over 25 epochs for $M = \{10, 50, 100, 250\}$, where M is the dimension of the hidden weights $\mathbf{W}$ and report the mean squared reconstruction error for the test dataset. A template container Bernoulli-Bernoulli has been provided to you. You may use it as a start.

2 Problem 2: Conditional Variational Autoencoders (50 points)

Recall that for a given data point $\mathbf{x} \in \mathbb{R}^D$ the evidence lower bound (ELBO) is given by:

$$ELBO = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] \tag{1}$$

Here $\mathbf{z} \in \mathbb{R}^M$, $M < D$ denotes the latent variable, $p(\mathbf{z})$ is the prior distribution over $\mathbf{z}$, $q_\phi(\mathbf{z}|\mathbf{x})$ is the variational posterior distribution (i.e., encoder), and $p_\theta(\mathbf{x}|\mathbf{z})$ is the likelihood function (i.e., decoder). D_{KL} denotes the Kullback-Liebler divergence and $\mathbb{E}$ is the expectation. In a standard VAE, the variational posterior is assumed to belong to a class of normal distributions, i.e., $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu x, \sigma x)$, whereas the prior is assumed to be standard normal, that is $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. As a further simplification, the covariance matrix of the variational posterior can be assumed to be diagonal.

In conditional VAE, the generative model is conditioned on a class $\mathbf{c}$ in which we want to learn parameters (θ, ϕ) such that $p_\theta(\mathbf{x}|\mathbf{c})$ generated by the generative model approximates $p(\mathbf{x}|\mathbf{c})$ for all c and the inference model, $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{c})$ approximates $p_\theta(\mathbf{z}|\mathbf{x}, \mathbf{c})$ for all x, c

Assume $\mathbf{x}$ belongs to a labeled dataset with multiple classes (such as Fashion MNIST). Recall that to sample from a specific class, we can train a class-conditional VAE; this can be achieved by conditioning the encoder and the decoder on the label of $\mathbf{x}$ (this can be done by concatenating the one-hot encoding vector of the label with the input to the encoder and the decoder).

2.1 (a)

Train a class-conditional VAE over the Fashion MNIST dataset. Use the standard training and testing split available in Pytorch. Build your own architectures for the encoder and decoder; you are free to use any building block and optimization method you have learned so far in this course. Use 20% of the training dataset as a validation set to tune the hyperparameters of your model.

Generate and plot

- The 2D manifold learned by the c-VAE (Use t-SNE to get reduced dimensionality if your latent space size is more than 2)
- 10 images for **Dress, Coat, Sandal,** and **Sneaker** classes each with trained c-VAE.

2.2 (b)

Train a standard VAE with a similar model architecture as the c-VAE you implemented in (a). Generate and plot the 2D manifold learned by the c-VAE (Use t-SNE to get reduced dimensionality if your latent space size is more than 2) Please compare and comment on the differences in feature representation (in terms of 2D manifold learning) between a standard VAE and c-VAE.

3 Problem 3: Sparse Encoding for denoising (bonus: 20pts)

Please consider an auto-encoding like this:

$$\min_{\mathbf{D}} \frac{1}{T} \sum_{t=1}^{T} \min_{\mathbf{h}^{(t)}} \frac{1}{2} ||\mathbf{x}^{(t)} - \mathbf{D}\,\mathbf{h}^{(t)}||_2^2 + \lambda ||\mathbf{h}^{(t)}||_1$$

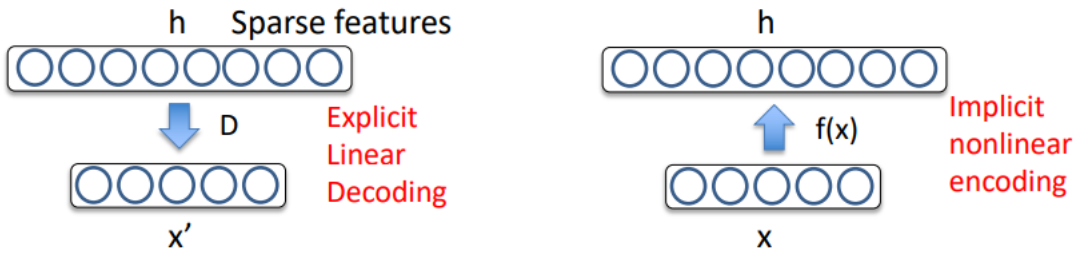

Figure 1: Over-complete auto-encoder with non-linear encoding and linear decoding modules. Lasso (L1) regularization is used to enforce meaningful feature learning

We will use MNIST again. Let the dimension of $\mathbf{h}$ be 1.5 times that of your input and $f(x)$ defined by your own conjecture. Let X be a batch sampled from MNIST, taking $X + Gaussain(0, \sigma I)$ (σ of your choice) as the model input and our goal is to obtain its output $\hat{X} \approx X$. Use the default training split for your auto-encoder. You will plot

1. 5 input-output pairs $(X + Gaussain(0, \sigma I), \hat{X})$ using the data in the test set.

2. The top 5 dictionary vectors in $\mathbf{D}$ whose corresponding intensity $|\mathbf{h_i}|$ are the largest.

Hint: using MLP for this may make the visualization easier.