CS 4331/5331 – Generative AI
Homework 2

Department of Computer Science
Whitacre College of Engineering
Texas Tech University

Due by Friday, Oct. 17, 11:59pm via Canvas

You are encouraged to work in pairs. Each team member must understand the entire solution, and clearly indicate their individual contributions (who did what) at the top of the first page of the submitted assignment.

The use of generative AI to solve this assignment—especially for writing code—is strictly prohibited. Any violation will result in failing the course. Please feel free to reach out to the course staff with any questions. Gaining hands-on experience is an essential part of the learning process.

1 Gaussian Mixture Models (20 pts)

Construct a 2-dimensional Gaussian Mixture Model (GMM) with three Gaussian components:

$$p(\mathbf{x}) \;=\; \sum_{k=1}^{3} \pi_k \, \mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\,, \quad \text{where } \boldsymbol{\pi} = \begin{bmatrix} 0.2 & 0.5 & 0.3 \end{bmatrix}.$$

Choose *arbitrary but distinct* mean vectors and covariance matrices for components 1, 2 and 3.

1.1 Sampling and Visualization (5 pts)

Sample 100,000 points from your GMM. Produce a 2D scatter plot that colors points by their generating component:

- $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$: red
- $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$: green
- $\mathcal{N}(\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$: blue

Include axis labels and a legend.

1.2 Monte Carlo Estimation of the Entropy of GMM (10 pts)

Extend the class code for MC estimation of the differential entropy of a *single* Gaussian to estimate the entropy of your *Gaussian mixture* $p(\mathbf{x})$.
Hint: For samples $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ drawn from p, an unbiased plug-in estimator of $H(p) = -\mathbb{E}_{\mathbf{x}\sim p}[\log p(\mathbf{x})]$ can be approximated by

$$\widehat{H}(p) \;=\; -\frac{1}{N}\sum_{i=1}^{N} \log p\Big(\mathbf{x}^{(i)}\Big)\,, \quad \text{where } p(\mathbf{x}) = \sum_{k=1}^{3} \pi_k\, \mathcal{N}(\mathbf{x}\,|\,\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\,.$$

Plot the estimated entropy as a function of a number of samples.

1.3 GMM with Nearly-Degenerate Weights (5 pts))

Repeat Task 1.2 using the *same* three Gaussian components but replace the weight vector with

$$\boldsymbol{\pi} = \begin{bmatrix} 1e^{-5} & 1 - 2\cdot 1e^{-5} & 1e^{-5} \end{bmatrix}.$$

Explain your observations and compare your estimate with the entropy of a single Gaussian $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ as in the single Gaussian case discussed in the class (see the code).

Plot the estimated entropy as a function of a number of samples.

Deliverables

- The figure from Task 1.1 with clear coloring and legend.
- Brief description of your chosen $\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3$ and the provided $\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2$.
- Your MC entropy estimates for Tasks 1.2 and 1.3, with the number of samples used and random seed reported.
- A short discussion comparing the Task 1.3 result to the entropy of the single Gaussian component.
- Plots of the entropy estimator as a function of number of samples.
- Well-documented code (with comments).

2 Variational Autoencoders (80 pts)

2.1 VAE.py (20 points total)

You will implement the main model class.

- Encoder / Decoder (10 pts)
  - Build an encoder network that maps input images into a latent distribution.
  - Build a decoder network that reconstructs images from a latent vector.
  - Hint: use simple fully connected layers (MLPs).
- Reparameterization Trick (10 pts)
  - Given mean vector μ and and log-variance vector $\log \sigma^2$, implement:

$$z = \mu + \sigma \odot \epsilon, \epsilon \sim \mathcal{N}(0, I) \tag{1}$$

  - $\mathcal{N}(0, I)$ is the standard multidimensional Gaussian distribution with the mean vector, μ, and the identity covariance matrix, I.
  - This makes the sampling operation differentiable so the VAE can be trained end-to-end.
  - Hint: use torch.randn_like to generate ϵ

2.2 ELBO.py (25 points total)

You will implement the VAE's loss function.

- Reconstruction Loss (10 pts)
  - Implement negative_log_likelihood(x, recon_x).py.
  - For this assignment, assume the output distribution is Gaussian.
- KL Divergence (10 pts)
  - Implement kl_diag_normal(mu, logvar).py for a diagonal Gaussian posterior vs. a standard normal prior.
  - Formula:

$$KL = -\frac{1}{2}\sum_{k=1}^{K}(1 + \log \sigma^2 - \mu^2 - \sigma^2) \tag{2}$$

- ELBO Loss (5 pts)
  - Formula:

$$L = ReconLoss + KL \tag{3}$$

  - Return the per-batch loss for training.

2.3 Derivation of Linear-Gaussian Model (5 pts)

For KL Divergence the general case in the general case is defined by:

$$KL(q||p) = \mathbb{E}_{x\sim q}[\log q(x) - \log p(x)] \tag{4}$$

For multivariate Gaussians it is defined by:

$$KL_{(q||p)} = \frac{1}{2}[-\log|\Sigma_q| + \log|\Sigma_p| - (x-\mu_q)^T\Sigma_q^{-1}(x-\mu_q) + (x-\mu_p)^T\Sigma_p^{-1}(x-\mu_p)] \tag{5}$$

For the specific case with diagonal covariacne matrices, it is reduced to:

$$KL = -\frac{1}{2}\sum_{k=1}^{K}(1 + \log\sigma^2 - \mu^2 - \sigma^2) \tag{6}$$

Prove it.

2.4 Visualization (20 points total)

You will write small scripts for visualizing the learned latent space.

- Latent Traversal (5 pts) — latent_traversal.py
  1. Vary each latent dimension over a range (e.g., -3 to +3) while keeping others fixed.
  2. Decode and visualize the results.
- Prior Sampling (5 pts) — sample_from_prior.py
  1. Sample random latent vectors from the prior $\mathcal{N}(0, I)$.
  2. Decode and visualize the results.
- Latent Space Visualization with t-SNE (5 pts) — tsne.py
  1. Encode a subset of the MNIST dataset into the latent space.
  2. Apply t-SNE to project the latent vectors into 2D and plot them, coloring each point according to its digit label.
- Plot the learning curve for ELBO (5 pts)
  1. Plot the ELBO learning curve per iteration.
  2. A learning curve represents values of ELBO as a function of iteration number.
  3. You would need to create your own function this time.
  4. Save the figure as "ELBO_Curve.png".
- Repeat the visualizations for different latent space dimensionalities. Generate plots at the beginning, middle, and end of training, and briefly explain your observations.

2.5 Experiments with Different Architecture (5 pts)

Remove the conv layers and replace it with fully connected layers only. What do you notice?

2.6 Experiments with Different Dimensionality of Latent Space (5 pts)

Increase and decrease the dimensionality of the latent space. What behaviors do you notice?

Deliverables

- Proof for Problem 2.3.
- Visualizations from Problem 2.4:
  - Latent traversal, prior sampling, latent space visualization with t-SNE, and ELBO learning curve.
  - For each latent space dimensionality, provide visualizations at the beginning, middle, and end of training.
  - Use at least two different latent space dimensionalities.
  - In total, this amounts to $4 \times 3 \times 2 = 24$ plots.
- Written answers for Problem 2.5.
- Written answers for Problem 2.6.
- Well-documented code (with comments).

Files

1. `VAE.py` — Contains the main VAE model.
2. `train.py` — Contains the function to train the VAE using ELBO.
3. `ELBO.py` — Contains the functions used to calculate ELBO.
4. `latent_traversal.py` — Contains the function for visualizing latent traversal.
5. `sample_from_prior.py` — Contains the function for visualizing sampling from learned prior.
6. `tsne.py` — Contains the function for visualizing the encoded MNIST latent space.
7. `main.py` — Contains the main code to run everything. Everything should run only using `python main.py`.