CSCI 1051 Problem Set 3

January 23, 2025

Submission Instructions

Please upload your solutions by **5pm Friday January 24, 2025.**

- You are encouraged to discuss ideas and work with your classmates. However, you **must acknowledge** your collaborators at the top of each solution on which you collaborated with others and you **must write** your solutions independently.
- Your solutions to theory questions must be written legibly, or typeset in LaTeX or markdown. If you would like to use LaTeX, you can import the source of this document here to Overleaf.
- I recommend that you write your solutions to coding questions in a Jupyter notebook using Google Colab.
- You should submit your solutions as a **single PDF** via the assignment on Gradescope.

Problem 1: Image Embeddings

In this problem, we will embed images using an autoencoder that you train from scratch. I recommend that use one of the datasets from the MNIST family.

Part A: Autoencoder Training

Train a small (three or so layers with activations) autoencoder to produce embeddings in two dimensions via reconstruction loss. I suggest writing one class for the encoder and one for the decoder.

Part B: Clustering Plot

Take (a subset of) images in your training data, encode them with the encoder portion of the autoencoder and plot them on a scatter plot.

Part C: Reconstruction Plot

Take a grid of points (about 10 by 10) in the range of points from the prior plot and pass them through your decoder. Now plot the resulting images on a grid based on their latent dimension.

Extra Credit (2 points): Variational Autoencoder

Train a variational autoencoder with reconstruction loss and variational loss simultaneously. Remember that the encoder will reproduce two vectors that you will interpret as the mean $\boldsymbol{\mu} \in \mathbb{R}^2$ and standard deviation $\boldsymbol{\sigma} \in \mathbb{R}^2_+$ of the latent vector. You can find the latent vector by computing $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.

The KL divergence (aka cross entropy) loss simplifies to something like

$$\sum_{i=1}^{2} -\log(\sigma_i) + \sigma_i^2 + \mu_i^2. \tag{1}$$

Create the clustering and reconstruction plots for the variational autoencoder you just trained.

Problem 2: Diffusion

In this problem, we will train a diffusion model from scratch.

Part A: Point Distribution

Write code to sample points from a recognizable but simple distribution in two dimensions.

Part B: Tuning T and α

Recall $\mathbf{x}_t = \sqrt{\alpha^t}\mathbf{x}_0 + \sqrt{1-\alpha^t}\mathbf{z}$ for $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Write code to plot $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T$. Choose the number of steps T and the multiplicative weight α so that the images slowly turn to noise.

Part C: Diffusion Training

Initialize a several layer neural network (both your input and output should be in $\mathbb{R}^2$). Train your diffusion model to predict the corresponding $\mathbf{z}$ for a given $\mathbf{x}_t$. Carefully tune your learning rate so the loss consistently decreases.

Part D: Diffusion Evaluation

In your training loop (every say 10 iterations), sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and repeatedly diffuse to $\mathbf{x}_0$ as described in class. Plot the resulting process and ensure your model learns the distribution!

Problem 3: Schrödinger Bridges

In this problem, we will train a Schroedinger Bridge from scratch.

Part A: Two Distributions

Create one distribution p_{data} that is a recognizable but simple distribution like two balls of points. Create another distribution p'_{data} that is the first distribution but shifted.

Part B: Diffusion Training

Set $T \approx 20$ and $\gamma = 2/T$. Using the simplified diffusion approach described in class, train a 'forward' model f to go from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ to p_{data} and a 'backward' model b to go from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ to p'_{data}.

Part C: Transport Plot

During training:

- Sample points $x_0 \sim p_{\text{data}}$ and apply the backward model b for T steps. Plot the progression of the points.
- Sample points $y_T \sim p'_{\text{data}}$ and apply the forward model f for T steps. Plot the progression of the points.

Extra Credit (2 points): Schrödinger Bridge

Iteratively train the models together via the Schrödinger bridge training as described in class.

During training, produce the same plots as described above. What do you notice?