MATH/COMP 562 ASSIGNMENT 4
DUE: APRIL 9TH (SUNDAY)
VERSION: April 9, 2023

ADAM M. OBERMAN

Instructions. Submit your solutions on MyCourses course page. You can get help from other students, but you should do the write up yourself. Code solutions can be a PDF of a workbook, you can use any programming language, but NumPy is recommended.

Generative Models. Refer to `https://udlbook.github.io/udlbook/`

Exercise 4.1 (GANs). Given a dataset of genuine images, $S^m = \{x_1 \ldots, x_m\}$. The goal is to train a generator, $x_i = g(z_i, \theta)$, to generate images $x_i \in S^m$ from noise z_i. This is achieved by also training a discriminator, $f(x, \phi)$, which classifies an image as genuine or generated, using the minimax training loss

$$\max_{\theta} \min_{\phi} \left(\widehat{L}_{S_\theta}(f(x, \phi), y) \right)$$

applied to the combined dataset, $S_\theta = S^N \cup S^G_\theta$,

- $S^N = \{(x_1, 1), \ldots, (x_m, 1)\}$, the (fixed) genuine images, with label $y = 1$,
- $S^G_\theta = \{(x^\theta_1, 0), \ldots, (x^\theta_m, 0)$, a changing set of generated images $x^\theta_i = g(z_i, \theta)$, with label $y = 0$.

(a) Write down and explain the loss for the discriminator, if the generator is fixed.
(b) Write down and explain the loss (or gain, since it is a maximization) for the generator, if the discriminator is fixed.
(c) Characterize the saddle points of the loss. If the generator outputs the full set of genuine images, is this optimal? What if the generator outputs just a subset of the genuine images?

A saddle point, (θ, ϕ), is a point where (i) if we fix ϕ and change θ, we cannot further increase the loss, and (ii) if we fix θ and change ϕ, we cannot further decrease the loss.

Exercise 4.2 (Diffusion generative models). Given the dataset $S^m = \{x_1, \ldots, x_m\}$, the goal is to generate images from the dataset. The inputs are

- a finite set of numbers $\alpha_t, t \in T$
- a finite set of noise vectors $\epsilon_j, j \in J$.

The outputs are a set of models, $g_t : \mathcal{X} \to \mathcal{X}$, $g_t(x) = g(x,t), t \in T$.
For each $t \in T$, let $S_t = \{x_{t,i,j}\}, i = 1, \ldots, m, j \in J$ be the dataset of t-noisy images,

$$x_{t,i,j} = \sqrt{\alpha_t} x_i + \sqrt{1-\alpha_t}\epsilon_j$$

The training loss for g_t is given by

$$\widehat{L}_{S_t}(g_t) = \frac{1}{m}\sum_{i=1}^{m}\sum_{j\in J} \|g_t(x_{t,i,j}) - \epsilon_j\|^2$$

Define the closest point map

$$g^{CP}(x) = g^{CP}(x, S^m) = \arg\min_{x_i \in S^m} \|x - x_i\|^2$$

(a) Assume that, for each $x_{t,i,j} \in S_t$, $g^{CP}(x_{t,i,j}) = x_i$. Find the loss minimizer g_t^*, and show that the loss is zero.
(b) In the case above, are there any other loss minimizers?
(c) How do the loss minimizers for this generative model compare to the loss minimizers for GANs? Which is better in terms of coverage of the dataset?

Exercise 4.3 (Normalizing Flows: transformation of densities). Problems 16.1 and 16.2 from "Understanding Deep Learning"

Exercise 4.4 (Normalizing Flows: Fixed point theorem). Problem 16.11 from Chapter 16 of "Understanding Deep Learning"

Exercise 4.5 (Diffusion generative models). Problems 18.1 and 18.2 from Chapter 18 of "Understanding Deep Learning"

Coding exercises.

Exercise 4.6 (GAN training). Your task is to train a generator that will generate samples from a 1D Gaussian with the goal of approximating the true Gaussian distribution centered around a true mean. Use binary cross-entropy to train both the generator and discriminator. Propagate the losses to the correct parameters. Track the losses of both generator and discriminator and plot them. Finally, plot the discriminator probability function for various epochs during training. Comment on any interesting behavior you notice. Change the learning rates for both generator and discriminator, what do you notice?

Bonus: Modify the training loop for stochastic gradient descent (one example at a time). What can you say about the loss graphs of the generator/discriminator compared to the ones trained with gradient descent?

Code at: `https://colab.research.google.com/drive/1P5kcP_a-AIYcz3JJPhyJJ5yJmO8sgOH` `usp=sharing`. *Note, the algorithm in the code is different from what is described in the book/hw. Here the idea is to train the generator to convince the discriminator that*

the generated images are natural. This is done by flipping the labels (so that we can minimize instead of maximize. There is no point changing the labels on the genuine images, since the discriminator is not changing.