Deep Learning / Spring 2024

Homework 3

Please upload your assignments on or before May 6, 2024.

- You are encouraged to discuss ideas with each other. But you **must acknowledge** your collaborator, and you **must compose your own** writeup and code independently.
- We **require** answers to theory questions to be written in LaTeX. (Figures can be hand-drawn, but any text or equations must be typeset.) Handwritten homework submissions will not be graded.
- We **require** answers to coding questions in the form of a Jupyter notebook. It is **important** to include brief, coherent explanations of both your code and your results to show us your understanding. Use the text block feature of Jupyter notebooks to include explanations.
- Upload both your theory and coding answers in the form of a **single PDF** on Gradescope.

---

1. **(5 points)** *Understanding policy gradients*. In class we derived a general form of policy gradients. Let us consider a special case here which does not involve any neural networks. Suppose the step size is η. We consider the so-called *bandit* setting where past actions and states do not matter, and different actions a_i give rise to different rewards R_i.

   a. Define the mapping π such that $\pi(a_i) = \text{softmax}(\theta_i)$ for $i = 1, \ldots, k$, where k is the total number of actions and θ_i is a scalar parameter encoding the value of each action. Show that if action a_i is sampled, then the change in the parameters is given by:

   $$\Delta\theta_i = \eta R_i(1 - \pi(a_i)).$$

   b. If constant step sizes are used, intuitively explain why the above update rule might lead to unstable training. How would you fix this issue to ensure convergence of the parameters?

2. **(5 points)** *Minimax optimization*. In this problem we will see how training GANs is somewhat fundamentally different from regular training. Consider a simple problem where we are trying to minimax a function of two scalars:

   $$\min_x \max_y f(x, y) = 4x^2 - 4y^2.$$

   You can try graphing this function in Python if you like (no need to include it in your answer.

a. Determine the saddle point of this function. A saddle point is a point (x, y) for which f attains a local minimum along one direction and a local maximum in an orthogonal direction.

b. Write down the gradient descent/ascent equations for solving this problem starting at some arbitrary initialization (x_0, y_0).

c. Determine the range of allowable step sizes to ensure that gradient descent/ascent converges.

d. (2 points). What if you just did regular gradient descent over both variables instead? Comment on the dynamics of the updates and whether there are special cases where one might converge to the saddle point anyway.

3. **(5 points)** *Generative models*. In this problem, the goal is to train and visualize the outputs of a simple Deep Convolutional GAN (DCGAN) to generate realistic-looking (but synthetic) images of clothing items.

   a. Use the FashionMNIST training dataset (which we used in previous assignments) to train the DCGAN. Images are grayscale and size 28×28.

   b. Use the following discriminator architecture (kernel size = 5×5 with stride = 2 in both directions):

      - 2D convolutions ($1 \times 28 \times 28 \rightarrow 64 \times 14 \times 14 \rightarrow 128 \times 7 \times 7$)
      - each convolutional layer is equipped with a Leaky ReLU with slope 0.3, followed by Dropout with parameter 0.3.
      - a dense layer that takes the flattened output of the last convolution and maps it to a scalar.

   Here is a link that discusses how to appropriately choose padding and stride values in order to desired sizes.

   c. Use the following generator architecture (which is essentially the reverse of a standard discriminative architecture). You can use the same kernel size. Construct:

      - a dense layer that takes a unit Gaussian noise vector of length 100 and maps it to a vector of size $7 * 7 * 256$. No bias terms.
      - several transpose 2D convolutions ($256 \times 7 \times 7 \rightarrow 128 \times 7 \times 7 \rightarrow 64 \times 14 \times 14 \rightarrow 1 \times 28 \times 28$). No bias terms.
      - each convolutional layer (except the last one) is equipped with Batch Normalization (batch norm), followed by Leaky ReLU with slope 0.3. The last (output) layer is equipped with tanh activation (no batch norm).

   d. Use the binary cross-entropy loss for training both the generator and the discriminator. Use the Adam optimizer with learning rate 10^{-4}.

   e. Train it for 50 epochs. You can use minibatch sizes of 16, 32, or 64. Training may take several minutes (or even up to an hour), so be patient! Display intermediate images generated after $T = 10$, $T = 30$, and $T = 50$ epochs.

If the random seeds are fixed throughout then you should get results of the following quality: