CIS 4190/5190: Applied Machine Learning — Fall 2022

Homework 4

Handed Out: October 19 — *Due: November 2, 8:00 p.m.*

- You are encouraged to format your solutions using LaTeX. You'll find some pointers to resources for learning LaTeXamong the Canvas primers. Handwritten solutions are permitted, but remember that you bear the risk that we may not be able to read your work and grade it properly — do not count on providing post hoc explanations for illegible work. You will submit your solution manuscript for the written portion as a single PDF file.
- The homework is **due at 8 PM** on the due date. We will be using Gradescope for collecting the homework assignments. Please submit your solution manuscript as a PDF file via Gradescope. Post on Ed Discussion and contact the TAs if you are having technical difficulties in submitting the assignment.

1 Multiple Choice & Written Questions

Note: You do not need to show work for multiple choice questions. If formatting your answer in LaTeX, use our LaTeX template `hw4_template.tex` (This is a read-only link. You'll need to make a copy before you can edit. Make sure you make only private copies.).

1. [Image Filtering/Convolution] (12 pts) We discussed the use of convolution filters for images briefly in class, mostly in the context of CNNs. However, before CNNs became popular, convolution filters had already been an essential part of signal processing and computational photography. You will probably be surprised by how many features in Photoshop or Lightroom can be easily implemented with the correct choice of convolution filter(s). In this question, we will take a look at a few common types of convolution filters for images, and visualize how they would transform the original image. Let's take the following image for example.

This is a gray-scale image, where each pixel can be represented by a value in $[0, 1]$, where 0 is black and 1 is white. The gray-scale image itself can be represented by a matrix of shape width $\times$ height, and we are going to apply 3×3 convolution filters to the matrix. Assume the bias parameter is set to 0 for all of these convolution filters.

For the following questions, you are given a convolution filter, and a few transformed images. Pick the transformed images that corresponds to applying the given filter to the image above. Here is a ipynb where you can test out these filters yourself: https://www.seas.upenn.edu/~cis5190/fall2022/docs/hw4_conv.ipynb

a. (4 pts) Consider the filter

$$X = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}.$$

If we apply X as a convolution filter to the original image, which of the following transformed images will we see? Briefly justify your answer.

(A)  (B) 

(C)  (D) 

b. (4 pts) Consider the filter

$$X = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

If we apply X as a convolution filter to the original image, which of the following transformed images will we see? Briefly justify your answer.

(A)  (B) 

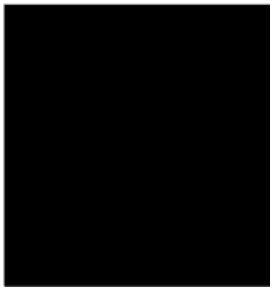
(C)  (D) 

c. (4 pts) Consider the filter

$$X = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}.$$

If we apply X as a convolution filter to the original image, which of the following transformed images will we see? Briefly justify your answer.

(A)  (B) 

(C)  (D) 

2. [Gradient descent] (5 pts) You are performing gradient descent with momentum to train a model with two learnable parameters $w = \begin{bmatrix} w_1 & w_2 \end{bmatrix}^\top$. Suppose that the loss is

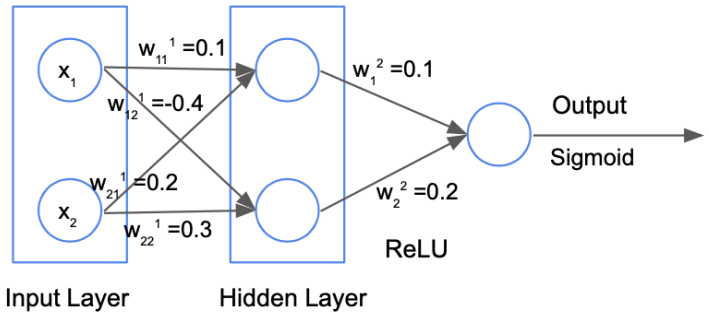
$$L(w) = \frac{1}{2}(w_1 - 1)^2 + \frac{1}{2}(w_2 - w_1)^2.$$

Recall that the update rule is

$$\rho_{t+1} \leftarrow \mu \cdot \rho_t - \alpha \nabla_w L(w_t)$$
$$w_{t+1} \leftarrow w_t + \rho_{t+1}.$$

Suppose we use hyperparameters $\alpha = 0.1$ and $\mu = 0.9$, and that we initialize $\rho_{1,1} = \rho_{1,2} = 0$ and $w_{1,1} = w_{1,2} = 0$. What is the value w_3 (i.e., take two gradient steps)?

3. [Backpropagation] (15 pts) Consider the following two-layer neural network:



You will use the ReLU function as the activation function at the hidden layer, and the sigmoid activation function at the output layer. Thus, the overall neural network is

$$f(x) = \sigma(W_2^\top \text{ReLU}(W_1 x)) \qquad \text{where} \qquad W_1 = \begin{bmatrix} 0.1 & 0.2 \\ -0.4 & 0.3 \end{bmatrix}, \qquad W_2 = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}.$$

The input vector is $x = \begin{bmatrix} 5 & 4 \end{bmatrix}^\top$. All weights are displayed on the image (the superscript denotes the layer). Suppose that the loss function is $\mathcal{L}(\text{output}) = \text{output}$. Compute the gradient of this loss with respect to each of the weights. Be sure to show details. As a reminder, the ReLU function and its derivative are:

$$\text{ReLU}(z) = \begin{cases} z & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$
$$\partial_z \text{ReLU}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

and the sigmoid function and its derivative are:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$
$$\partial_z \sigma(z) = \sigma(z) \cdot (1 - \sigma(z)).$$

4. [CNNs] (8 pts) Consider the following CNN, with layers enumerated from the input:

   (a) **Input**

   (b) **2D Conv:** In channels: 3, Out channels: 6, Kernel: 3×3, Stride 1, Padding 0

   (c) **ReLU**

   (d) **2D Max Pool:** Kernel: 2×2, Stride 2, Padding 0

   (e) **2D Conv:** In channels: 6, Out channels: 12, Kernel: 3×3, Stride 1, Padding 0

   (f) **ReLU**

(g) **2D Max Pool:** Kernel: 2×2, Stride 2, Padding 0

(h) **2D Conv:** In channels: 12, Out channels: 24, Kernel: 3×3, Stride 1, Padding 0

(i) **ReLU**

(j) **2D Max Pool:** Kernel: 2×2, Stride 2, Padding 0

(k) **Output**

If the input image has dimensions 230×230×3, compute the following: (a) output dimensions (3 dimensions), and (b) number of learnable parameters. For (b), include bias terms in your computation.

5. [Ensemble/AdaBoost, Mandatory for CIS 5190, optional for CIS 4190] (8 pts) In this problem, we will try to understand the intuition behind the AdaBoost algorithm by walking through the rationale behind the choice of β_t at each step t. Recall that at each step t, the AdaBoost algorithm trains a base model f_t on the dataset (X, y) with example weights w_t. We assume the weights are normalized, i.e., $\sum_{i=1}^{n} w_{t,i} = 1$. The weighted training error of f_t is

$$\epsilon_t = \sum_{i=1}^{n} \mathbb{1}(y_i \neq f_t(x_i)) \cdot w_{t,i},$$

and we define

$$\beta_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right). \tag{1}$$

Then, on the next step, we update the weights to be

$$w_{t+1,i} = w_{t,i} \cdot \frac{e^{-\beta_t y_i f_t(x_i)}}{Z_t},$$

where $Z_t = \sum_{i=1}^{n} w_{t,i} \cdot e^{-\beta_t y_i f_t(x_i)}$ is a normalization factor that ensures $\sum_{i=1}^{n} w_{t+1,i} = 1$. Also, recall that $y_i \in \{-1, 1\}$.

a. (3 pts) Show that Z_t equals the exponential loss, i.e.,

$$Z_t = e^{\beta_t} \cdot \epsilon_t + e^{-\beta_t} \cdot (1 - \epsilon_t). \tag{2}$$

b. (5 pts) Prove that β_t minimizes Z_t. [Hint: Compute the derivative of Z_t as defined in Eq. (2) with respect to β_t, set this derivative equal to zero, solve for β_t, and show that the solution equals Eq. (1).]

2 Python Programming Questions

Please see the IPython Notebook for the programming portion of this assignment.