MIT 6.C01/6.C51 Modeling with Machine Learning
Spring 2024
Problem Set 3

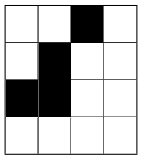
Release date. **Monday, March 11, 2024, 4:00 PM ET**
Due date. **Thursday, March 21, 2024, 11:59 PM ET**

Instructions

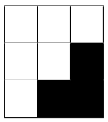
- This problem set contains three questions, each containing sub-questions.
- Submissions should be made via Gradescope. You are required to prepare one PDF containing all your written answers and code snippets. Please answer each question on a separate page, since Gradescope will require you to select pages pertaining to each sub-question.
- You have been provided `6_c01_pset_3.py`, a Python file containing starter code for Problem 1. Use it to write and test your implementations of the different functions we ask you to provide code for. When submitting your solutions, paste code that the question asks you into the final PDF you will submit wherever asked for.
- Some sub-questions are marked as **Graduate version.** These sub-questions are mandatory for those who have registered for the graduate version of this class (`6.C51`). It is optional for others.
- Please adhere to the collaboration policy described on our course page on Canvas. Clearly mention the names of your collaborators on the top of your first page in your submission.

Problem 1: CNN [55 points] [55 points]

Consider the following 4X4 image:



Pixel (i,j) in the image is the pixel on row i and column j, the upper left pixel is (1,1). You learned in 6.C01 that you can use filters to identify patterns in images and you want to locate the following pattern:



Assume black pixels have value of 1 and white pixels value of -1.

1. (5 points) If we apply the filter without padding, will we be able to identify the pattern? Explain.
2. (5 points) Now we pad the input image with a layer of zeros, which you can think of as white cells. Then we apply the filter on the image with stride 1. What will be the size of the output image?
3. (10 points) What are the indices of the pixel with the highest value in the output image and what is the value?
4. (5 points) How many times was pixel (1,1) involved in the convolution without the padding? How many times it was involved in the convolution with the padding?
5. For the last part of this problem, we are going to work hands-on with a CNN to classify images. You and your friends have found an interest in chess over IAP, and want to build a helper that can classify a position by looking at the picture of a board (see Figure 1). For this problem, you will have two classes: "insufficient material[1]" and "sufficient material".

[1]You do not need to know what this means to solve the problem, but for the curious: The pieces in chess are king, queen, rook, knight, bishop, and pawn. For our purposes, all boards have the two kings plus one extra non-pawn piece. If the piece is a knight or a bishop, then the game is a draw due to insufficient material, and otherwise it is not.

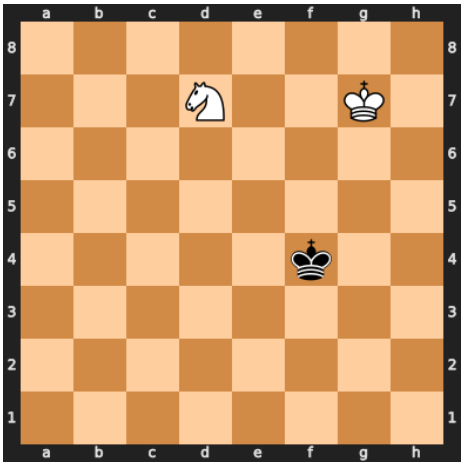


Figure 1: A chess board which is a draw due to insufficient material.

Instead of building your CNN from scratch, you are going to work with a pre-trained model called ResNet18. These models are trained on a large corpora of images and are very good at image classification, and you can fine-tune them to suit many image classification tasks.

(a) (5 points) Just like what you've seen in class, ResNet has a lot of hidden layers connected to a linear layer with 1000 output nodes to classify an image among 1000 possible labels. However, you only have two labels. Complete the code in `6_c01_pset_3.py` to modify the last layer of the model to reflect our problem. Answer by copying your code snippet (it should not be longer than two lines!).

(b) (15 points) Next, follow the instructions in the code to fill in the train and validate loops. Answer by copying your code snippets (it should not be longer than nine lines total!).

(c) (10 points) Before training your model, run the validation loop once. Finally, train and validate the model for one epoch. Answer by 1) stating your validation error and accuracy before training, 2) your average epoch loss at the end of the epoch, and 3) your validation error and accuracy after training.

Problem 2: RNNs [10 points] [20 points]

Consider a recurrent neural network (RNN) of the following form, where x_t is the input, y_t is the corresponding label used in training (both dimension $n \times 1$), h_t is the hidden layer (dimension $m \times 1$), $\hat{y}_t$ is the RNN output (dimension $n \times 1$), and f_1, f_2 are the activation functions:

$$h_t = f_1(Wx_t + Vh_{t-1})$$
$$\hat{y}_t = f_2(Uh_t)$$

For simplicity, assume that all the offsets in this problem are 0.

1. (10 points) Consider an RNN where $m = 2$ and $n = 1$, f_1, f_2 are identity functions, and the weights are as follows:

$$W = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad V = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \quad U = \begin{bmatrix} -2 & 3 \end{bmatrix}$$

   If the hidden states are initialized to zeroes and the first two elements in the input sequence are $x_1 = -1, x_2 = 1$, what is the RNN output sequence? (*Hint: There should be two elements in the RNN output sequence.*)

2. **Graduate version.** (10 points) Assume that we finished training the given RNN with some overall loss $J = \sum_t \text{Loss}(y_t, \hat{y}_t)$, which is the sum of the individual losses given our training set (x_t, y_t). We are now interested in measuring the sensitivity of our loss J with respect to a particular input x_t. Assuming that we know $\frac{\partial \text{Loss}}{\partial \hat{y}_t}$ (dimension 1×1), find $\frac{\partial J}{\partial x_t}$ (dimension 1×1) in terms of weights W, V, U and input x_t. (*Hints: Not all matrices will be used. Check that the matrix dimensions work out.*)

Problem 3: Backpropagation [14 points] [14 points]

In this question, we will explore some issues that arise during backpropagation, specifically vanishing and exploding gradients. Let us illustrate the vanishing gradient problem with the sigmoid activation function $f(x) = \frac{1}{1-e^{-x}}$, which squishes our input space to the range [0, 1], as seen below:

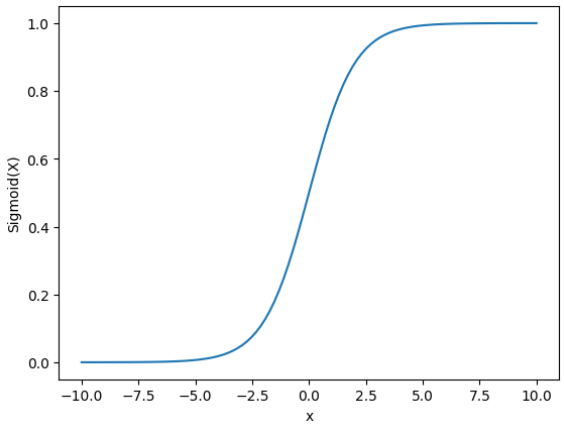


Figure 2: Sigmoid Function

1. As we have seen in previous examples, each update in backpropagation consists of many gradients multiplied together. When you multiply many small gradients together, the resulting product $\frac{\delta Loss}{\delta \theta_i}$ will tend to 0 as the depth of the neural network increases. This leads to the vanishing gradient problem.

   (a) (3 points) This problem can arise at the "saturating regions" of a graph, where the derivatives approach zero. Where does this occur for the sigmoid function?

   (b) (3 points) Recall the gradient descent algorithm $\theta_{i+1} \leftarrow \theta_i - \eta \frac{\delta Loss}{\delta \theta_i}$, where η is the learning rate. If $\frac{\delta Loss}{\delta \theta_i} \rightarrow 0$ due to the vanishing gradient problem, why might this be problematic? (1-2 sentences)

   (c) (5 points) One way to address the vanishing gradient problem is choosing a different activation function. Recall the activation function $\text{ReLU}(x) = \max(0, x)$. Find the derivative of ReLU with respect to x and explain how this can address the vanishing gradient problem. (1-2 sentences)

2. (3 points) Conversely, the exploding gradient problem occurs when the gradient gets larger as our backpropagation progresses. Referring back to our gradient descent algorithm, how might exploding gradients be problematic? (1-2 sentences)