Deep Learning Fall 2023

Prof. Gustavo Sandoval

- You are encouraged to discuss ideas with each other. But you **must acknowldge** any collaborators, including students, ChatGPT, books, online sources, etc. and you **must** write up your own solutions individually.
- We **require** answers to theory questions to be writen in LaTeX. Figures can be drawn by hand, but **must** be scanned and inserted into your document. We will not accept hand-written answers.
- We **require** code for programming questions to be submitted as a Jupyter notebook. It is important to include sufficient documentation so that the grader can understand your code. Use the text cells in Jupyter notebook to provide explanations.
- Upload your .pdf, .ipynb and any other files to Gradescope in a single PDF.

Problem 1: Slow rate of descent (20 pts)

Consider a simple function having two weight variables:

$$L(w_1, w_2) = 0.5(aw_1^2 + bw_2^2).$$

(a) Write down the gradient $\nabla L(w)$, and derive the weights w^* that achieve the minimum value of L.

(b) Instead of simply writing down the optimal weights, let's now try to optimize L using gradient descent. Starting from some arbitrary initialization point $w_1(0), w_2(0)$, write down the gradient descent updates. Show that the updates have the form:

$$w_1(t+1) = \rho_1 w_1(t), \quad w_2(t+1) = \rho_2 w_2(t)$$

where $w_i(t)$ represent the weights at the t^{th} iteration. Derive the expressions for ρ_1 and ρ_2 in terms of a, b, and the learning rate.

(c) Under what values of the learning rate does gradient descent converge?

(d) Provide a scenario under which the convergence rate of gradient descent is very slow. (*Hint: consider the case where a/b is a very large ratio.*)

Problem 2: Designing Convolution Filters by hand (20 pts)

Consider an input 2D input image and a 3x3 filter (say w) applied to this image. The goal is to guess good filters which implement each of the following image processing operations. Write both the weights of the filters and an explanation on how/why they work.

(a) **Feature Extraction:** The filter should output a high value when the input image has an edge, and a low value otherwise.

(b) **Blurring Filter:** The filter should blur the image.

(c) **Sharpening Filter Horizontal:** The filter should sharpen the image Horizontally.

(d) **Noise Reduction:** The filter should reduce noise in the image.

Problem 3: The IOU Metric(10 pts)

In class we discussed the IOU metric (or Jaccard similarity) for evaluating object detection.

(a) Using elementary properties of sets, prove that the IOU metric between any two pair of bouding boxes is always between 0 and 1.

(b) If we represent each bounding box as a function of the top-left and bottom-right coordinates, then argue that the IOU metric is non-differentiable and hence cannot be used as a loss function for training a neural network.

Problem 4: Training AlexNet (25 pts)

Open the (incomplete) Jupyter notebook provided as an attachment to this homework in Google Colab (or other Python IDE of your choice) and complete the missing items.

Problem 5: Object Detection (25 pts)

We will be finetuning a pretrained Torchvision object detection model in this problem. It will be helpful to go through the excellent tutorial with sample code provided by PyTorch here. You can reuse whatever code you like from the tutorial in your solution with proper attribution.

(a) The tutorial shows how to finetune a pretrained model (what they call option 1). Now follow their approach to add a different backbone (option 2). You can use the same dataset (and code) as the tutorial.

(b) How does the performance of the two models compare on the training data after 10 epochs.

(c) Test both models on this image of the Beatles. Be careful to make sure that the dimensions align and the scaling is similar. How do the two models compare in terms of performance on this image?