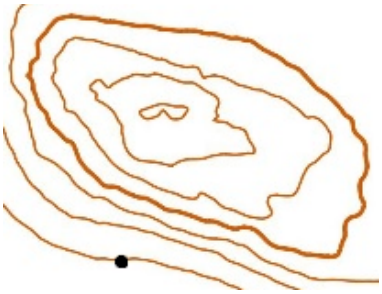
ECE 411 Homework 4 (Fall 2024)
Instructor: Dr. Chau-Wai Wong
Material Covered: Modern ML applications

All work must be submitted in the PDF format to Gradescope. Do not submit Colab or Jupyter Notebook files or links.

Problem 1 (20 points) [Level Curves and Gradient Descent]

a) A set of level curves is shown as follows. Use the dot as the starting point, and draw a trajectory of gradient descent steps. Annotate each descent step using a line segment with an arrow at the end. Explicitly draw a tangent line at each step, which can assist you in determining the negative gradient direction. You may vary the descent step size.



b) A cost function is given as $J(w_1, w_2) = -\exp(-w_1^2 - 10w_2^2)$. Use the command you learned from Problem 1 of HW2 to draw a contour plot/set of level curves. Illustrate by drawing two initial points and their gradient descent trajectories using a fixed step size. One initial point must lead to fast convergence and the other must lead to slow convergence. (You may draw the trajectories on a printout, or you may take a screenshot, paste it into PowerPoint, draw the arrows and tangent lines, and save it as a PDF file.)

Problem 2 (20 points) [Simple Neural Network for Data with Nonlinear Decision Boundaries] You are going to play with the code of a simple neural network that does binary classification. Open *the Colab notebook file* using Google Drive. Quickly scan through the whole document to get a high-level idea, and then sequentially run the code blocks by clicking the play button on the top left corner of each code block.

a) Draw three convergence curves in one plot for `learning_rate` $= 10^{-3}, 10^{-4},$ and 10^{-5}. The horizontal axis should be the iteration number and the vertical axis should be the loss/training error. For smaller learning rates/step sizes, you may want to increase `num_of_iter` to allow the curve to flatten out.

b) *This Colab notebook file* will walk you through implementing a simple linear regression model $y = 3x + 1$ using a PyTorch neural network. Fill in the missing lines of code

around `criterion` and `outputs` by learning the syntax from part a) and reading the PyTorch Documentation, if needed. After that, compare the regression lines under different loss functions, e.g., `nn.L1Loss()`, `nn.MSELoss()`, and under different noise variances within the range $[1.5, 5]$. Submit the plots and key lines of your source code.

Problem 3 (20 points) [Neural Network for Classifying Digits] Open *the Colab notebook file* using Google Drive. Quickly scan through the whole document to get a high-level idea, and then sequentially run the code blocks by clicking the play button on the top left corner of each code block. Examine how the following factors affect the convergence rate and test accuracy:

a) Learning rate

b) Number of epochs

c) Batch size

d) Number of hidden units

In your opinion, what is the best combination of the parameters that leads to a reasonable trade-off between accuracy and convergence time?

Problem 4 (20 points) [Character-level LSTM] (Work on this problem after Monday, 9/16's lecture) In *this Colab notebook file*, you will use a recurrent neural network with long short-term memory (LSTM) units to predict the next character based on a Shakespeare writing. The trained model auto-generates texts, which can imitate the writing style of Shakespeare. To start text generation, you should pass a starting char(acter), from which you then generate one char at a time. Examine the following items:

a) Test the trained model by passing different starting chars. Also, train with your own dataset, e.g., writing from another author, and show the results.

b) Use one sentence each to explain what the following functions do: `random_chunk()` and `random_training_set()`.

c) Draw an unrolled block diagram for the following code segment:

```
for c in range(chunk_len - 1):
    out_target = target[c].unsqueeze(0).type(torch.LongTensor)
    out, hidden, cell = model(inp[c], hidden, cell)
    loss += criterion(out, out_target)
```