COMPSCI 570	**Instructor:** Ronald Parr
Homework 4	**Due:** 11/22/24

Note: You may **not** search the internet, use ChatGPT/ChatGPT-like systems, or use any other (generative)AI/foundation/large-language models to help answer these questions. You may use the internet to look up basic definitions and methods relating to probability, expectation, and Bayesian probability.

You may have high level discussions with your classmates about understanding the underlying concepts and about how to approach these problems in general, but may not share anything resembling a complete solution. What you write up and turn in should be your own.

1 Neural Weight Design (20 points)

Consider the XOR problem illustrated in the graph below, where points $(0, 1)$ and $(1, 0)$ are treated as negative and points $(0, 0)$ and $(1, 1)$ are treated as positive.

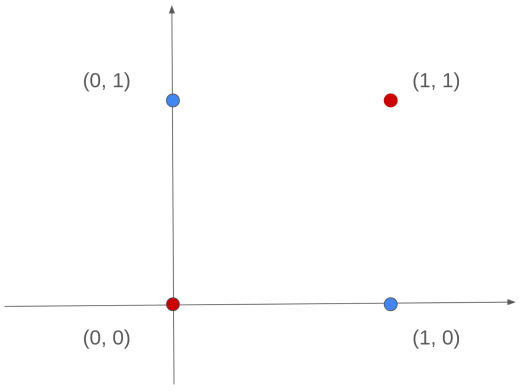

Figure 1: Bayes net for Question 1

1. We gave an intuitive argument in class that that no linear classifier that could classify the points perfectly. Prove this rigorously. (10 points)
2. Construct a two-layer perceptron (step activation function) that can classify the points perfectly. You should specify the weights and biases of each node and show that every data point after computation will yield desired results. (10 points)

2 Understanding Backpropagation (20 points)

The central algorithm in deep learning is gradient back propagation that computes gradients for each parameter of the network. The general derivation is somewhat tedious, but understanding what's happneing in one dimension can help build your intuitions.

Suppose a training point is represented as (x, y). We first pass x through a node with the activation function as *cos* (chosen to make the calculus simple), namely, $x_1 = cos(w_1 x + b_1)$. We then pass x_1 through a linear node to get the final output as $\hat{y} = w_2 x_1 + b_2$. Suppose we are doing gradient descent using squared loss, $f = (\hat{y} - y)^2$. Please calculate the partial gradients of f with respect to the parameters of the model, w_1, b_1, w_2, b_2.

3 Exploring Deep Learning (60 points)

In this question, you will work with and implement a couple of neural networks to classify the CIFAR-10 images. An IPython notebook file that contains implementation instructions and our starter code is provided for you. The notebook file also contains some inline questions to be filled out.

The networks are be implemented in Python 3. We expect you to know a bit of Python. If you don't have any experience with Python, this quick tutorial might help you. You can easily install the latest version of Python via Anaconda although Anaconda is not specifically required. You also need to have SciPy packages, PyTorch, torchvision and Jupyter installed **to the latest version**. Once the environment is set up, download `CompSci570DeepLearning.ipynb`, copy it to your working directory and launch `JupyterLab` (or notebook) under that directory. Alternatively, you could do all this through colab.

`PyTorch` is an open source machine learning framework where you can quickly prototype and train neural networks. For this homework, you will work with and modify CNNs (please read the tutorial) and also use Autograd (please read the tutorial). This question is fairly lightweight, requiring primarily that you run our code, make some observations, and made some modest changes to try a few new ideas of your own. *There is no excuse for copying somebody else's code.*

We will follow up with instructions on Ed for how to submit your notebook on GradeScope.