| | |
|---|---|
| **COMPSCI 570** | **Instructor:** Ronald Parr |
| **Homework 4** | **Due:** 11/20/25 |

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 Linear Separability (20 points)

Consider the family of functions defined over $n$ binary inputs which outputs true (1) when exactly $k$ inputs are true, and false (0) otherwise. Observe that XOR corresponds to n=2, k=1. Prove that for any $n \geq 2$ and any $1 \leq k \leq n-1$, the function is not linearly separably. Note that your proof must work for all $n$ and $k$. Providing an example that works for one choice of $n$ and $k$ will get 0 points.

## 2 Understanding Backpropagation (20 points)

The central algorithm in deep learning is gradient backpropagation, which computes gradients for each parameter of the network. The general derivation is somewhat tedious, but understanding what's happening in a simpler case can help build your intuitions. While backpropagation is presented in terms of a sigmoid activation function, it can work with any differentiable activation function.

Suppose a training point is represented as $(x_1, x_2, t)$, where $x_1$ and $x_2$ are features, and $t$ is the target. We first pass $x$ through a node with the activation function as $e^{-x}$ (chosen to make the calculus simple), namely, $z = e^{-(w_1 x_1 + w_2 x_2)}$. (We'll skip a bias term for this layer.) We then pass $z$ through a linear node to get the final output as $y = w_3 z + b$. Suppose we are doing gradient descent using squared loss, $f = (\hat{y} - y)^2$. Calculate the gradient of $f$ with respect to the parameters of the model, $w_1, w_2, , w_3$, and $b$.

# 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.*

Submit a single PDF to gradescope with your answers to questions 1 and 2, and your completed python notebook.