CS 6375
ASSIGNMENT 2

Names of students in your group:
Terisha Kolencherry Prax

Number of free late days used: 1
Note: You are allowed a **total** of 4 free late days for the **entire semester**. You can use at most 2 for each assignment. After that, there will be a penalty of 10% for each late day.

Please list clearly all the sources/references that you have used in this assignment.
Fixing the Relu Derivative Array Issue
https://stackoverflow.com/questions/46411180/implement-relu-derivative-in-python-numpy

Fixing the Max vs Maximum Issue for Relu
https://stackoverflow.com/questions/44957704/passing-relu-function-to-all-element-of-a-numpy-array

Nitty Gritty Logic for the Neural Network - First Attempt and Background
https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/

Simplified Way to Code Neural Network
https://medium.com/@qempsil0914/implement-neural-network-without-using-deep-learning-libraries-step-by-step-tutorial-python3-e2aa4e5766d1

What Loss Function for Binary Classification
https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/#:~:text=In%20this%20article%2C%20we%20will,used%20for%20binary%20classification%20problems

Decision Boundaries for Different Classifiers
https://medium.com/analytics-vidhya/activation-functions-in-neural-network-55d1afb5397a

Tanh
https://datascience.stackexchange.com/questions/109547/why-does-using-tanh-worsen-accuracy-so-much

Small Learning Rate for Tanh and Relu
https://stats.stackexchange.com/questions/324896/training-loss-increases-with-time

Kaggle Dataset
https://www.kaggle.com/datasets/mojtaba142/hotel-booking/

Assignment 2 Write Up

Executive Summary

This project looked at hotel cancellations in relation to the type of hotel, the number of babies a guest was bringing, whether a guest was a repeat customer, if the guest had previously cancelled reservations, how many special requests they had, and what type of deposit a guest had made when booking the room.

I compared neural networks that had varying learning rates, epochs, sample sizes, and activation functions. I noted that learning rates had to be much smaller for the ReLu and TanH activation functions compared to the sigmoid function. For this data, the sigmoid function seemed to work best, but I take that conclusion with a grain of salt since there is quite a bit I think I could improve on the neural network:

- It's possible that this data isn't great. The correlation table for quantitative predictors isn't very high and of the qualitative predictors, there weren't many that seemed like they would be worth the complexity of additional dummified columns. However, the most thorough methodology would be to start with a full model and then start pruning predictors.
- Better decision boundaries – I ended up using the sigmoid function for the output layer (see below) so I just used the standard decision boundary at 0.5 for all three activation function options, but I think that boundary could probably be better represented.
- More hidden layers – I think having a few more layers might help boost the ReLu/TanH accuracy. Although, we want to be careful not to having vanishing or exploding gradients.

PARAMETER TESTING SUMMARY

N	Epoch	Learning Rate	Activation	Train Error	Test Error	Result/Notes
300	50	0.5	Sigmoid	0.200	0.183	Too big for training data - results in all zeroes
300	20	0.5	Sigmoid	0.246	0.183	Too big for test data - results in all zeroes
300	10	0.5	Sigmoid	0.696	0.683	Good size - mix of zeroes and ones
ALL	20	0.5	Sigmoid	0.452	0.459	Loss is going back and forth - check lr
ALL	50	0.5	Sigmoid	0.399	0.407	The loss keeps going up
ALL	80	0.5	Sigmoid	0.389	0.397	The loss keeps going up
ALL	30	0.5	Sigmoid	0.433	0.440	Loss is going back and forth - lr
ALL	30	1	Sigmoid	0.438	0.440	Loss is going down but very slowly
ALL	30	5	Sigmoid	0.382	0.388	
ALL	30	2	Sigmoid	0.384	0.390	
ALL	30	0.3	Sigmoid	0.475	0.480	Loss is going down
ALL	30	3	Sigmoid	0.342	0.388	
1000	100	3	Sigmoid	0.357	0.343	
3000	30	1	Sigmoid	0.356	0.345	
3000	10	1	Sigmoid	0.406	0.387	
3000	30	1	Relu	0.357	0.343	
3000	30	5	Relu	0.357	0.343	Need a MUCH smaller learning rate
300	10	0.00001	Relu	0.679	0.679	
300	30	0.001	Tanh	0.317	0.317	
3000	10	0.00001	Tanh	0.604	0.604	
3000	10	0.01	Tanh	0.604	0.615	
3000	10	0.001	Tanh	0.447	0.444	Removed 'babies' + 'special_requests' + Sigmoid output
3000	30	3	Sigmoid	0.432	0.425	Removed 'babies' and 'special_requests'
3000	10	0.000001	Relu	0.446	0.445	Removed 'babies' and 'special_requests' + Sigmoid output

Part Zero: Pre-Processing

I used the Hotel Cancellations dataset from Kaggle to build a neural network model that could correctly predict whether a reservation would be cancelled. The response variable here is a binary one – either a room is cancelled (1) or not (0). We have learned in class that simpler is often better and we also talked about vanishing/exploding gradients, so I decided to initially scale down on the number of predictors from the initial 35 predictors to 9.

To save myself some time, I only looked at variables that had values for all rows, which surprisingly did not significantly narrow down the number of predictors. I went through and looked at the descriptions of each and selected a few that I thought might be pertinent: hotel type, number of babies, if they were a repeated guest or not, if they had previous cancellations or not, the total number of special requests the guest had, and the deposit type.

Before I started using my data, I had to dummify some of the columns. Since hotel type only had two options – resort or city hotel I just encoded them as 1 and 0, respectively. But for deposit type, I created three separate columns for each deposit type: No Deposit, Non Refundable, and Other.

Part Two: Building the Network – Lessons Learned

Loss Function

I started building the neural network using the sigmoid activation function that we had studied in class. The first error I made was forgetting that the loss function I had used for last time (MSE) wouldn't work because this is a classification question and not a regression question. So I spent some time on the internet looking up loss functions for binary classification. I ended up using the log loss function because I thought it might fit better for a model that was outputting specific probabilities instead of a distribution of probabilities and I figured I could reuse the math for logistic regression. If I had more time, it would be interesting to see if the hinge-loss would be more or less informative than what I used.

Decision Boundaries

Once I had a working neural network, I started coding in the additional activation functions and I slowly realized I was assuming that they all had the same decision boundary. Since ReLu and TanH are not linear activation functions, it makes sense that their decision boundaries might not be linear and even if they were, their thresholds might not be the same since their ranges vary from the sigmoid function.

Hidden Layer Activation vs Output Activation

Aside from the decision boundaries, one of the hardest issues I dealt with was trying to figure out why my error rate was so high whenever I toggled to ReLu or TanH. Initially I was using those activation functions at both my hidden layer and output layer and I kept getting errors or a large proportion of zeroes. For reference, the prior probability for the "No Cancellation" class was about 63%, but I was getting around 90% zeroes because of the way the ReLu and TanH functions and derivatives are structured. I did tweak the model to use a sigmoid function on the output and that helped drive down the error rate a bit. I think since my model only had one hidden layer it didn't really get to flex the power of ReLu or TanH so in the future it would be interesting to see how those error rates change given multiple hidden layers utilizing those activation functions.

Removing Predictors

Once I had a few iterations under my belt, I looked back at the data and saw that two predictors weren't really providing that much additional information so I took out 'babies' and 'special_requests' and that helped the error rate go down.