Part 1: Initial tests for application code

The first task was to download the network definition code and try the check 1 file. I have done that and got the same output as the html file output. I used google collab and checked with that output and it matched with the output I got.

Part 2: Modifications to be made in the network code

For this part, class functions were modified for activation functions and cost functions. ReLU activation function and Loglikelihood cost function were also added to the python file given.

(A)Hyper parameters

Cost Function

Cross Entropy Cost Function- For this class the function was given and the derivative was added to it.

Loglikelihood cost function- In this class there was two functions needed and was written for this cost function with those two functions- ***fn ()*** and ***derivative ().***

Activation function

Tanh – There was a skeleton code provided with this where I modified the ***fn ()*** and ***derivative ()*** functions.

ReLU- In this class function also I modified the ***fn ()*** and ***derivative ()*** functions.

SoftMax- Here the fn () was given and I modified the derivative function for this activation function.

Regularization

There were two functions in the network class that needed the L1 and L2 regularization that is the update_mini_batch () and total_cost () functions and that was added in those specific functions.

Dropout rate

I have tried experimenting in this when the dropout rate is 0.0 there was no drop in the neurons of the network but when I changed it to 1.0 the network drops all the neurons in the layer. So, to overcome this I created a new variable using the formula given below,

dropout = 1 - self.dropoutpercent where dropout percent was a hyper-parameter.

By using the dropout variable, I created a dropout mask that is a list which consists of NumPy array for each hidden layer in the network. The dropout mask was created by checking if the dropout percent is greater than 0.0.

The dropout rate needed a modification in the backprop () function and that has been modified with respect to the dropout rate.

(B) Functions to Modify

Set_parameters () Function

This function has the parameters used in the network. In this function few things were modified as listed below,

- The 'Loglikelihood' cost function needed a modification here because it can only be used while the output activation function is 'SoftMax'. So, I modified the set_parameters () function to automatically change the cost function to 'Cross Entropy' when the output activation function is not 'SoftMax'.
- The 'Tanh' activation function needed a similar change that it only accepts 'Quadratic' cost function. So, I modified the function like whenever the output activation function is 'Tanh' then the cost function automatically changes to 'Quadratic' cost function if it's not.

Feedforward () function

The forward propagation of the network is computed using this function. Few things that are modified in this function are listed below,

- The hidden layers of the network were computed using the activation functions defined for the hidden layers.
- The output layer of the network was computed differently using the activation function defined for the output layers.

Backprop () function

This function is used to compute the backward propagation of the network. In this function, the following things were modified.

- The hidden layers of the network were computed using the activation functions defined for the hidden layers.
- The output layer of the network was computed differently using the activation function defined for the output layer

Update_mini_batch () function

For updating the weights and the biases of the network the update_mini_batch function is used and the changes made here are,

- For the regularization this checks the type of the regularization to classify between L1 regularization and L2 regularization which is provided as a hyper parameter and applies the appropriate regularization. The two types are 'L1' and 'L2 ' regularization.

Total_cost () function

There is a need in changing the cost for the regularization techniques and the following were modified,

- As similar to the update_mini_batch () this function also checks for the type of regularization and changes the appropriate cost for the appropriate regularization.

Part 3: Own Code

Results

Sl.no	Act_hidden	Act_output	cost	regularization	dropout	result
1	Sigmoid	Sigmoid	CrossEntropy	(default)	(default)	Complete
2	Sigmoid	Softmax	CrossEntropy	(default)	(default)	Complete
3	Sigmoid	Softmax	LogLikelihood	(default)	(default)	Complete
4	ReLU	Softmax	LogLikelihood	(default)	(default)	Complete
5	Tanh	Sigmoid	Quadratic	(default)	(default)	Complete
6	Sigmoid	Sigmoid	Quadratic	L2 2.0	(default)	Complete
7	Sigmoid	Sigmoid	Quadratic	L1 2.0	(default)	Partial

The first seven experiments were done with "iris-423" data set and I received matching results except L1 regularization. I have implemented the L1 regularization by updating the weights as mentioned in the book but I'm not able to match the results. There was difference in it. I have also done by updating the costs in the total_cost() function.

Sl.no	**Act_hidden**	**Act_output**	**cost**	**regularization**	**dropout**	**result**
1	Sigmoid	Sigmoid	Quadratic	(default)	0.3	Complete
2	Sigmoid	Softmax	CrossEntropy	(default)	0.3	Complete

In the drop out rate experiment the results did not match with the given output. This is due to the randomness in creating the dropout mask. As mentioned in the question this may vary in the output so in my knowledge I have correctly implemented. A mask was created for each and every layer of the network with matching the size of the neurons and it was then applied to the activations of each layer.

Part 4: Reflections

Overall, this assignment was more challenging than the assignment 3. As for the theoretical part I was able to understand by the help of the book which was really helpful. I faced the difficulty when implementing in the coding part for example I tried tanh in built but the formula one gave me a better output which matched with the given output. In python the NumPy packages was quite hard to get the grasp of some functions.