Homework 6: Backprop and Multi-layer Nets **NSC 3270 / 6270**
Due Tue Oct 25 at start of class **Fall 2022**
20 points

This homework assignment builds on Homework 5, both conceptually, and in terms of the code provided. In HW5 you examined error-driven learning in a single layer network. Now you will examine networks with one or more hidden layers.

There's no new starting code for HW6. Rather, you will build upon and adapt the code from HW5: The code I gave you, the code you wrote yourself, and any of the IPYNB files provided. You will also need to refer to class slides for information on how to do some of the parts outlined below.

You will be doing more self-directed coding this time, in terms of overall code organization and structure. But remember, you've been given a lot of code that can be used as a starting point. Get started on the assignment right away and come to Prof Polyn's or Jordan's office hours if you're having trouble getting started (or at any point in the process).

Specific Tasks

Q1 (2 points). Create training patterns and teaching patterns for an XOR classification problem. In Homework 5 you used the gensamples function and judicious concatenation to create training and teaching patterns for a basic classification problem. Now you'll use the same approach to create an "XOR problem" (as illustrated in the BackpropLectureSlides-Part1 class slides).

Create 200 training patterns per category (400 training patterns total) and use the same standard deviations and correlations we used for Homework 5. As in HW5, you'll have to make sure the matrix of training patterns and the matrix of teacher values are configured properly for the Keras functions. Note that you'll have to call the gensamples function 4 times, twice for each class. Then you can use the concatenate function to stitch those training patterns together into one set, and make sure the teaching patterns line up properly with the training patterns.

Create a figure that displays the training patterns. In case you don't remember: This is the kind of figure where there are blue plus signs for one of the classes and red plus signs for the other class. Here you're just creating the training patterns, not training the network yet, so the figure will not have a shaded-in background indicating classifier predictions.

Q2 (5 points). Create a multi-layer neural network that can learn this XOR problem using Keras. We know from lecture that a single-layer network cannot solve the XOR problem. Here, you'll have to make several choices, and also justify those choices. Things you'll have to decide & justify: The number of hidden layers, the number of nodes per layer, which optimizer (and any relevant optimizer settings), the number of training epochs, and batch size.

Regarding justification: One way to justify a choice is to describe how things went wrong when the network was structured differently (e.g., "when I changed X to Y, the

network couldn't learn the classification"). Another way is to describe how your choice is more efficient than some alternative (e.g., "the network reliably learns the classification with N epochs so I didn't need to have N+1000 epochs").

Use "validation_split" to hold out 20% of the training patterns for validation. You will use this to check to see if your network is overfitting. This will be discussed in class as there's a potential pitfall here we don't want you to fall into. The pitfall: Let's say you set your validation split to 0.2. By default, Keras will put aside the last 20% of your training patterns for validation. If your training patterns aren't shuffled, this will mean all of the validation patterns will be from the same class. This will unbalance the number of patterns you have from each class for actual training, and will also mean your validation performance is much less informative. So, after you create your training patterns and teaching patterns, you'll have to shuffle them into a random order, so the class 0 and class 1 patterns are interspersed. Alternatively, you can intersperse the class 0 and class 1 patterns in a more orderly way. As long as the two classes are interspersed, you won't fall into this particular pit.

Note that you may have to try several different network architectures and settings to create a network that will reliably solve your XOR problem (and that you can justify). As in HW5, part of the justification process is making sure you're network isn't dramatically overdoing it by having too many training epochs or hidden units. The plots you create for Q3 can help inform your choices.

As you try different architectures and Keras settings, use a markdown cell to briefly document the different options you investigated and why you chose the particular combinations you arrived at.

Q3 (3 points). Visualize the training performance and classification decisions of the network. Create three plots: A plot of training accuracy as a function of epoch (you did this in Homework 5). A second plot with both the loss `'loss'` and validation loss `'val_loss'` on the same graph as a function of epoch (you didn't do this in HW5). A third plot that displays the training patterns with a shaded background indicating the classifier's predictions for a mesh grid of test patterns (you did this in HW5 using the mesh grid and plottest code provided).

Q4. (3 points). Create your own classification problem for a neural network to learn. We will discuss this in class. You'll use gensamples to make a novel classification problem.

Guidelines: It must be a problem that is not linearly separable. It can't match the examples shown in class (i.e., don't just flip the labels from 0 to 1 and 1 to 0, it has to be appreciably different from the "simple" XOR problem demonstrated in class).

You can stick with two classes of patterns, or you can decide to add more than two classes. The network and its settings need to be chosen appropriately based on any modifications you make (e.g., if you add more classes you'll probably need to add more output units). If your problem has two classes, there must be more than one distribution per class (each time you call gensamples, that's "one distribution").

The distributions that make up your problem should have unequal variances and non-zero correlations. Display your classification problem in a figure (as in Q1).

Q5. (5 points). Create a multi-layer neural network that can learn your custom classification problem using Keras. As with Q2, create a multi-layer neural network that can learn your problem using Keras.

Again, you will need to decide the number of hidden layers, nodes per layer, choose an optimizer, number of epochs, and batch size (and justify these choices as above). As in Q2, hold out 20% of the training patterns for validation (and make sure to intersperse the training patterns as described above).

It will take some exploration of architecture and settings to arrive at choices that work (and that you can justify). Use the plots in Q6 to inform your choices.

We will be looking to see that your network is reasonably small (number of layers and number of nodes per layer) to learn your problem (i.e., that its complexity is justified). A network with several hidden layers and dozens of nodes per layer might learn your problem, but would be undoubtedly far too complex to be justifiable. In a markdown cell, briefly document the different options you investigated and why you chose the particular combinations you arrived at.

Q6 (2 points). Visualize the training performance and classification decisions of the network. As in Q3, create three plots: A plot of training accuracy as a function of epoch. A second plot with both the loss `'loss'` and validation loss `'val_loss'` on the same graph as a function of epoch. A third plot that displays the training patterns with a shaded background indicating the classifier's predictions for a mesh grid of test patterns (you did this in HW5 using the mesh grid and plottest code provided). You may need to adjust the mesh grid of test patterns and edit the `plottest()` function to work with your classification problem (like if your patterns range further on the x- and y-axes of the xy-plane).

Unexcused late assignments will be penalized 10% for every 24 hours late, starting from the time class ends, for a maximum of two days, after which they will earn a 0.