Homework 8 **NSC 3270 / 6270**
Due Thu Dec 8 at start of class **Fall 2022**
30 points

This assignment combines two homework assignments into one, to give you more time to work on both parts. Both parts are due the last day of class (Thu Dec 8). We will discuss Part One first, and we will discuss Part Two in more detail after Thanksgiving break.

PART ONE (20 points)

Overview. You will use Keras to carry out a set of pattern classification analyses on one sample participant from the Haxby et al. (2001) functional MRI study.

Note that this assignment will require more programming (especially in terms of organizing code into multiple functions) than other assignments.

In previous assignments, you used the Keras toolbox to create, train, and test error-driven classification networks. Now you will use the same Keras framework to analyze functional neuroimaging data. This gives you hands-on experience with multivariate pattern analysis (MVPA), a common tool for analyzing neuroimaging data (with the Haxby et al. paper one of the first examples of the technique).

Participants in the Haxby experiment viewed images from 8 different categories (faces, houses, cats, shoes, scissors, bottles, scrambled images, and chairs) while the fMRI scans were recorded (see class slides for details). You will create classification networks using Keras where the training patterns are the fMRI voxels recorded while participants viewed the images, and the target output values indicate which category is being viewed (using one hot coding). In other words, your networks will attempt to decode the participant's perceptions from the pattern of blood-flow changes across the ventral temporal lobe.

Lecture slides provide details regarding how to carry out certain parts of this assignment.

You will find the following data files and Jupyter notebook on Brightspace:

`haxby_vt_patterns.npy`	contains the pre-processed functional imaging data
`labels.txt`	contains the category labels and run numbers
`Homework8.ipynb`	beginning code for the assignment

Q1a (15 points). **Run an MVPA analysis using leave-one-out cross validation on the supplied Haxby et al. data.** The class slides will explain cross-validation and provide a template for selecting training and testing patterns for the cross-validation procedure.

A cross-validation analysis involves selecting a portion of the data for training a classifier and another portion for testing that classifier. Each time you do this, it is called a "fold". You will be carrying out cross-validation at the level of the individual runs. Because there are 12 runs, this means there will be 12 folds of the cross validation. For each fold, one run's

worth of data become the test patterns, and the rest of the data become the training patterns.

You will use a basic classification network without a hidden layer for Q1a and Q1b. The class slides note that you should not use validation items (we will discuss why) and you should use a regularizer (to help prevent overfitting). You will need to select the number of training epochs, as discussed in class. We recommend using a softmax activation function for the output layer, and categorical cross entropy for the loss function.

Iterate over the full leave-one-out cross-validation 5 times (in a real application, you might run it 100 times). We do this because the classifier itself is "noisy" (relying on stochastic gradient descent over a network initialized randomly). This allows you to evaluate how reliable the classification performance is.

Report average classification accuracy overall (over the 12 folds, 8 categories, and 5 iterations). Because there are 8 categories, chance performance is 12.5%. You should be able to get classification performance somewhere in the 30-50% range (neural data is noisy, and this can limit the maximum performance of the classifier).

Then you'll report classification performance in more detail. Report the average classification accuracy for each of the 5 iterations of the analysis. How much does this value vary across the iterations? Report the average classification performance for each of the 8 categories. Make note of which categories have better and worse overall classification performance. Report the variability of the category-level performance across folds and iterations.

Finally examine and report average classification performance for each of the 12 folds. Remember that each fold uses images from one "run" of the experiment as testing patterns. Are there any folds that seem to have poor classification performance? This could mean that the data from that run is of lower quality (e.g., if the participant moved in the scanner during that run).

Q1b (5 points). **Determine whether your observed classification accuracy is greater than chance.** Because there are 8 categories, we know that chance performance should on average be 1/8 or 12.5%. However, this doesn't tell us how much variability around that value to expect. If our classifier gets 15% of the test patterns correct, is that better than chance? This task will help you answer that question.

To determine whether the performance of your classifier is reliably better than chance, you will write code to carry out a permutation test. A full permutation test can take a long time to run (possibly overnight depending on how powerful your laptop is) so the assignment allows you to run a partial version of the permutation test. If your laptop is fast enough, you may want to run the full version for your own edification, but it is not required for the assignment.

The class slides will provide details of how to carry out the permutation test.

The essence of a permutation test is to scramble the labels associated with the fMRI data (in a particular way, as noted on the class slides) and re-run the analysis. Scrambling the labels breaks the true correspondence between the fMRI signal and the actual category of the observed picture. As such, the level of performance by a classifier for the permuted (scrambled) analysis provides a measure of the level of classification performance you would expect by chance. And by looking at the different values you get each time you re-scramble, you get a sense of the distribution of values in the "chance distribution".

`Homework8.ipynb` provides code for randomly scrambling the labels, and the procedure will be further explained in the class slides. We will talk about strategies for saving you a lot of computation time. For example, you can write your code so it saves the results of the permutation analysis to disk. That way, if you end up changing some aspect of the code that's not part of the permutation analysis, you don't necessarily have to re-run the whole thing.

In a full permutation test, you would scramble the data 1000 or even 5000 times. For each permutation, you would record the level of accuracy from a classifier trained on this scrambled data. The distribution of 1000 or 5000 levels of accuracy provides a distribution of classification performance you would expect by chance. From this large number of classifications of scrambled data, you can find the critical level of accuracy (upper 5^{th} percentile) whereby levels of classification accuracy (on the original data) would be considered "significantly greater than chance". If everything works out properly, you should expect your true accuracy to be greater than all the values in your chance-level-accuracy-distribution (probably by a pretty wide margin).

PART TWO (10 point)

For Part Two of this final assignment, you will generate some representational dissimilarity matrices (RDMs) and multi-dimensional scaling (MDS) solutions for the images and network representations from the CNN model you created for Homework 7. (We will talk about RDMs and MDS the week after Thanksgiving, in class.)

You will get a substantial amount of code to work with, in the Jupyter notebook file `Homework8PartTwoMNIST.ipynb` on Brightspace. This code illustrates the approach to use for this part of the assignment with the original MNIST dataset (and an example CNN model). You will adapt this code to work with the Fashion-MNIST dataset and the CNN model you submitted for Homework 7. Recall that much of the coding for MNIST and Fashion-MNIST are nearly identical to one another, so the code I gave you provides a lot of what you need. You just need to understand the code and adapt it appropriately.

Q2a (4 points). **Examine the similarity structure of Fashion-MNIST patterns.** You will create RDMs and MDS solutions using five examples of each of the 10 clothing types in the

Fashion-MNIST dataset. Much of this task can be accomplished by adapting the code examples provided for the handwritten digits MNIST dataset.

Calculate and display the RDM based on the raw pixel values of the images (the resulting RDM will be 50x50) and display a 2D MDS solution from the RDM. Make sure the RDM and MDS plots are labeled with the names of the clothing types.

Q2b (4 points). **Examine the similarity structure of the intermediate representations in your CNN.** and display the RDM based on an early or intermediate layer of your trained CNN model from Homework 7 (based on the discussion from class and using the approach outlined in the ipynb file provided) and display a 2D MDS solution from this RDM. Make sure the RDM and MDS are labeled with the names of the clothing types.

Q2c (2 points). **Examine the similarity structure of the deeper representations in your CNN.** Do the same as Q2b but for the penultimate layer (the layer before the classification layer) of your CNN model from Homework 7.

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.