

Worksheet: artificial neural networks

Joaquin Rapela

March 10, 2025

This worksheet is a continuation of the [Neuromatch tutorial on decoding neural responses](#) that we covered in the practical session on artificial neural networks.

Here we will:

1. evaluate the performance of the neural network model on test data,
2. use a circular variable loss function,
3. perform model selection (i.e., select optimal number of layers and layer width),
4. visualise the tuning curve of the most relevant neurons,
5. train the model in a GPU.

1 Evaluate the model on test data

Partition the data into a train and a test set. For example, the train set can contain 60% of the total data and the test set the remaining part. Use the train set to estimate the model parameters. For each epoch in the training loop use the mean-squared error (MSE) loss function on the train set and on the test set. Plot these two errors as a function of iteration, as in Figure [1](#).

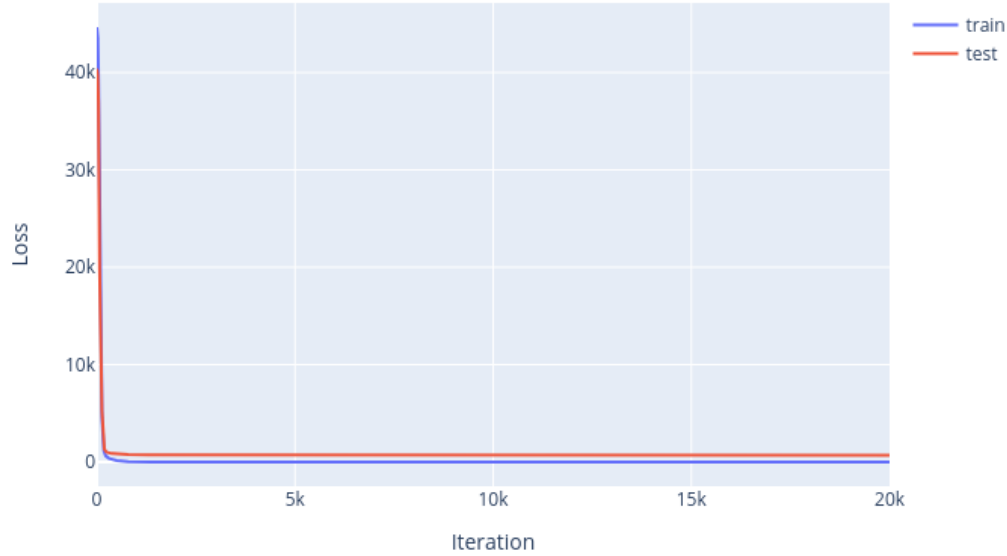


Figure 1: Train MSE and test MSE as a function of iteration. Click on the figure to see its interactive version.

To train the mode you may want to use the script `doTrainNet.py` with the parameters in the script `trainMSETestMSESGD.csh`.

To plot the train and test loss function you may want to use the script `doPlotTrainTestLosses.py` with the parameters in the script `plotLossesTrainMSETestMSE.csh`.

2 Circular variable loss function

Orientation is a circular variable. The MSE is not suited for circular variables. Create a loss function for circular variables. Use it to estimate a model using this new loss function on the train and test data. Build a figure as Fig. 1, but using the new loss function instead of the MSE loss function.

You may want to complete function in function `loss` of class `CircularLoss`. Then you can estimate a model with this loss function using the script `doTrainNet.py` using parameters `train_loss_fn_type=Circular` and `test_loss_fn_type=Circular`. Also, adjust the parameter `learning_rate` to obtain best results.

To test which of the MSE or Circular loss is better, estimate a first model with the MSE loss for the train data and the Circular loss for the test data, and a second model with the Circular loss for the train data and the Circular loss for the test data. The model giving the minimum loss with the test data is better. Use the same random seed for both estimations.

3 Model selection

Estimate models with different number of hidden units, compare their test errors and report the best number of hidden units that you found. Use train and test circular loss functions

Optionally:

1. test if using a nonlinear model yields substantially better results than using a linear one. Tip: replace `net = myNets.DeepNetReLU(n_neurons, n_hidden)` with `net = myNets.DeepNetLinear(n_neurons, n_hidden)` in `doTrainNet.py`.
2. vary the depth of the network.

4 Visualise the tuning curve of the most relevant neurons

Not all neurons are equally informative for the decoding task. Neurons with largest weights into the hidden layer should be more relevant. Plot the tuning function of the first 50 neurons with largest weights to the hidden layer, as in Figure 2. Then plot the 50 next neurons with largest weight. See if you observe a pattern in the most relevant cells that is different from the other cells.

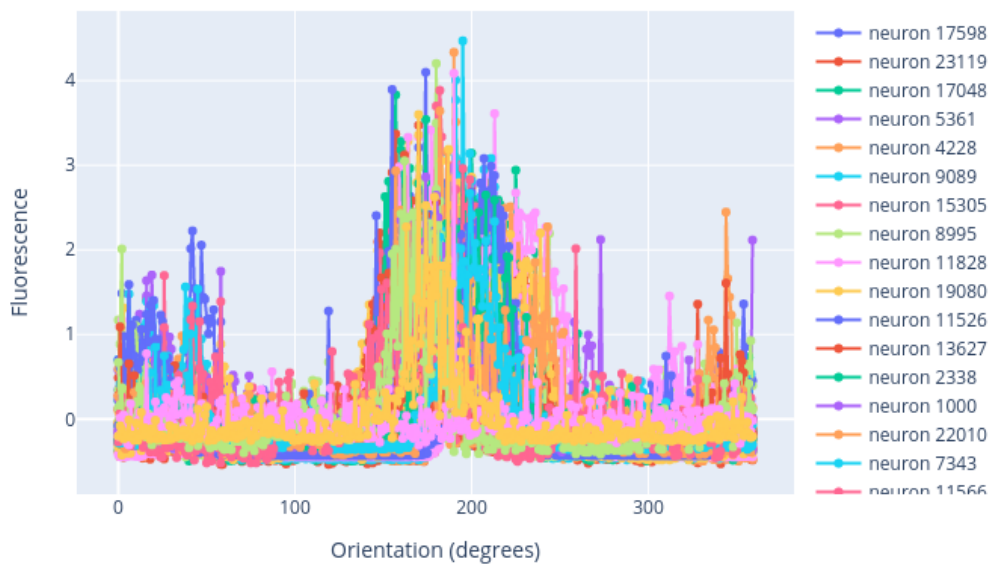


Figure 2: Tuning functions of the 50 neurons with largest weights to the hidden layer. Click on the figure to see its interactive version.

You may want to use the script `doPlotTuningCurves.py` and set the parameter `--rank_1st_neuron`.

5 Train the model in a GPU

If a GPU is available in your system, the script `doTrainNet.py` will run on the GPU.

At the SWC you can run this script in the cluster by:

1. **ssh into the bastion node**

```
ssh ssh.swc.ucl.ac.uk
```

2. **allocating an interactive job in the cluster**

```
srun -p fast -t 2:00:00 --gres=gpu:1 --pty bash -i
```

3. **running the script**

```
./trainCircularTestCircularSGD.csh
```