Neural Network for Classification of Bipolar Electrode Recordings of the Cortical Region of the Brain

Data Pre-processing

My classifier starts by filtering the signal using a band-pass Butterworth filter with cut-off frequencies 5 and 5 000 Hz, to reduce the high-frequency noise and low-frequency drift of the recording. While the training dataset does not suffer from drift, it is not representative of the submission dataset, so the filtering is necessary.

Then, peaks are detected in the filtered waveform. By inspection of the training data, I found that the threshold height should be 2 mV, and peaks should rise at least 1 mV from their surroundings. In the training data, the indices of the spikes were offset by a few samples before the peak itself. The mode offset was 8, so I hard-coded this value in my pre-processor.

For each peak, a 70-sample-wide window centred on the peak is stored as the data to be classified. I found that this window width was sufficient to store the widest peaks without including too much irrelevant noise. I found that the samples in the spikes ranged from -1.5 to 12.5, so I re-mapped the samples from the range $[-1.5, 12.5]$ to $[0, 1]$ to normalise my inputs.

Classifier Model

I used an artificial neural network to classify each recording. I chose a neural network over k nearest neighbour because it was a 70-dimensional problem, which is too high-dimensional to be suitable for KNN.

I used a 128-neuron hidden layer as this gave very good classification accuracy (>95%) when tested against my testing dataset, and it trained very quickly. To achieve fast training times, I used the Relu activation function because its derivative is 1, while the logistic sigmoid function has a derivative of 1/4. This allows errors to backpropagate more quickly[1].

During the training phase, my model randomly sets the output of 20% of neurons in the hidden layer to 0. I did this to prevent overfitting of my model.

I used the Adam optimiser function to perform the gradient descent during the training phase. This function achieves greater computational efficiency than stochastic gradient descent by varying its learning rate per parameter[2].

I applied the softmax function to the output layer so that the value of each output represents a confidence level, summing to 1. This wasn't necessary for this application as I simply take the most likely predicted classification as my output, but it allows my model to be used in a future application where the confidence level would be considered.

Prediction

Once the model is trained, it can be used to predict the indices and classifications of the spikes in an unlabelled dataset. The waveform is pre-processed to find the locations of spikes and split the waveform into 70-sample-wide windows centred on each spike. The location of the peak minus the offset (8) is saved as the spike's index. The window of samples is classified using the neural network, and the classification with the highest confidence level is saved as the spike's class.

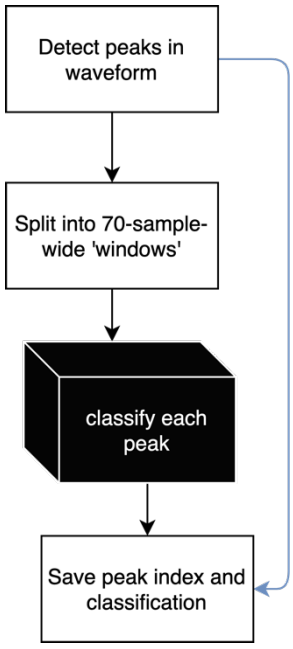

Figure 1: flowchart of classifier operation

[1] https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions

[2] https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/