Documenting the development of a multilayer perceptron for the prediction of signal environments

David Simmons

October 2, 2017

Abstract

This document highlights the development that took place when the author was tasked with applying machine learning techniques to make predictions about which type of environment a signal had propagated through. Concretely, the author received 36 twenty second time series signal envelope channel measurements. The signals were classified into two types: onbody and body to body. An onbody link is one in which the source and receiver of the signal are located on a single user. A body to body link is one in which the source and receiver of the signal are located on distinct users. With basic features (signal variance, peak-to-trough distance, ith percentile, max, min, and skew) the author was able to attain a test accuracy of $\sim 95\%$ using a two hidden layer fully connected perceptron without regularization. In this case, effectively no overfitting could be induced. By increasing the feature space through frequency domain samples, the author was able improve test accuracy to $> 99\%$ without regularization. When frequency domain samples were considered, overfitting was observed, which could be dealt with using neuron dropout.

1 The system model that is being classified

The system model is depicted in Fig. 1 below. Three devices (nodes) were located on two users: two on one user, one on the other. The devices acted as transceivers operating at 2.48 GHz, successively measuring the received signal strength between each other for 20 second periods. Time domain samples were taken for the 6 links ($1 \leftrightarrow 2$, $1 \leftrightarrow 3$, $2 \leftrightarrow 3$) that were present in total. The goal of this study was to classify the links into either onbody ($1 \leftrightarrow 2$) or body to body ($1 \leftrightarrow 3$, $2 \leftrightarrow 3$).

2 The code

To run the code, fork the repository (which includes test data) and run the python file "MLP.py" using "python MLP.py" from the folder "/MLP". The script

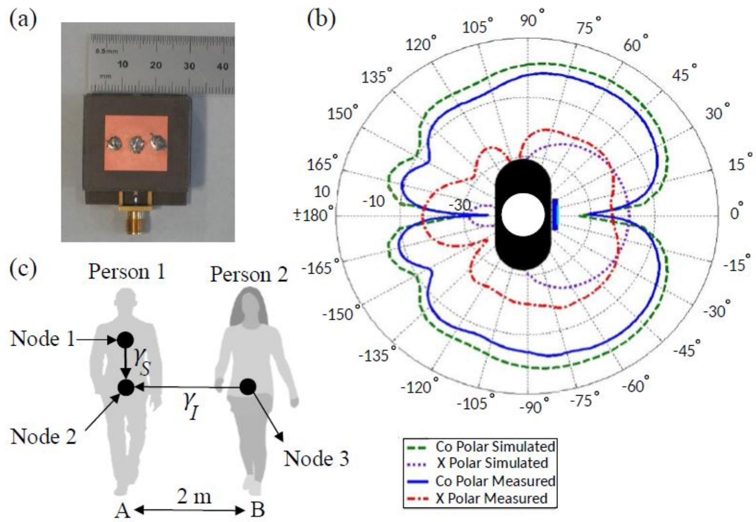

Figure 1: (a) The antenna under test, (b) the antenna's radiation pattern, (c) the two person setup.

will request feature related inputs and then run.

2.1 The data scraper

The scraper() function in get_data.py scrapes data from the folder /Original_data/Data_separated_S1_S2/dBm_in/. Each of the data blocks (36 in total) contains ~ 20 seconds of time domain samples (135000 samples). Blocks of overlapping 20000 sample windows were then extracted from each of the 20 second windows. The window shift was 500 samples, so that 230 blocks of data could be extracted from each of the 20 second windows. In total, $230 \times 36 = 8280$ windows were extracted from the data. Finally, each block was normalized by its euclidean norm.

2.2 Feature extraction

When the code is run, the user will be asked if they want to consider statistical features. If they enter y, the "features" function will extract

1. variance,
2. 10th, 20th, ... , 90th percentiles
3. max
4. min
5. max - min

Figure 2: Figures showing cross entropy and accuracy as functions of the number of batches when a dropout probability of 0. Each batch contained 100 signal windows.

6. skewness.

The code will then ask the user if they want to consider frequency domain samples as features. If they enter y, the "features" function will perform an FFT of order specified by the user. Finally, the code will take the requested feature types, concatenate them into a single feature vector, and normalize this vector by its norm.

Results

Selecting both feature types, with an order of 100 for the FFT and a neuron dropout probability of 0, gives the accuracy and cross entropy results shown in Fig. 2. As can be seen from this figure, overfitting begins to emerge. This is tackled by considering a neuron dropout probability of 0.02 in Fig.3.

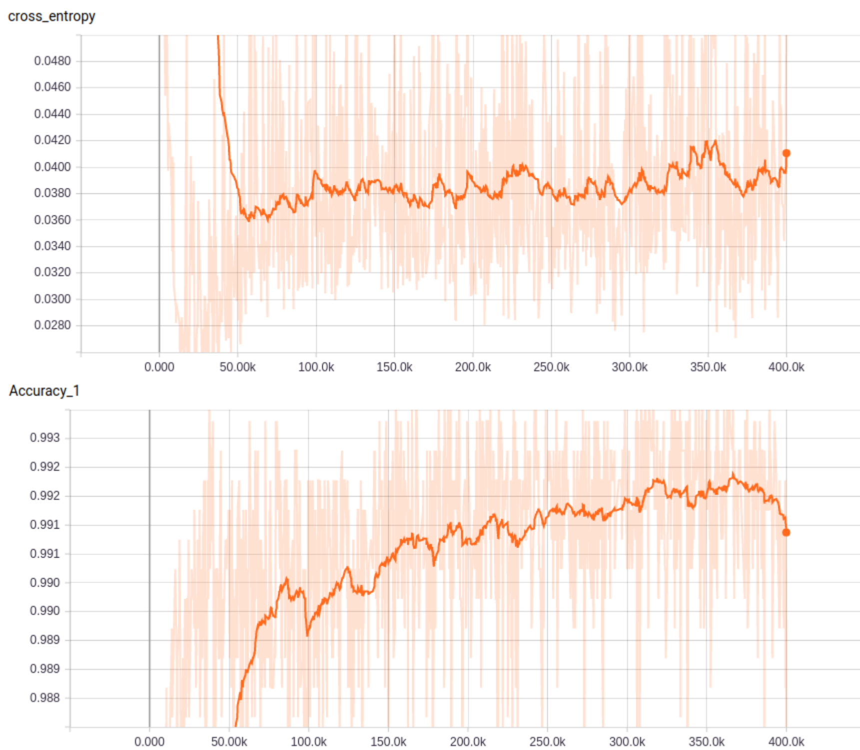

Figure 3: Figures showing cross entropy and accuracy as functions of the number of batches when a dropout probability of 0.02 is employed. Each batch contained 100 signal windows.