How to train your “Adaline”!

Setting up

Once you have assembled your Widrow-Hoff ‘Adeline’ demonstrator device, you will need to upload the Arduino code from the Github repository here using the Arduino IDE. Connect your computer to the Arduino USB port with the appropriate cable and compile and upload the code to the correct COM port. Once, the upload process is complete, all the keys on the keypad should illuminate completely red and the LCD screen should read, “`Sig = -0.00`” and have a red background.

Training

Now select a training pattern, for example a ‘J’ shaped letter perhaps (see Figure 1, this is the example that Bernard Widrow used himself to demonstrate his device back in the 1960s), by pressing the appropriate keys to *toggle* them to blue as shown below. [One could, of course, choose any pattern (or indeed make a ‘J’ from red buttons on a blue background for example)].

Figure 1: Enter a ‘J’ pattern using blue buttons on red background and then train by pressing “–1”.

Now that the ‘J’ pattern has been entered, press the red pushbutton (negative class) for a second or so until the `Sig` value starts to approach `-0.7` or lower. It is worth noting that the output value passes through a *sigmoid function* (hyperbolic tangent, see Figure 2) which is often used in machine learning to *normalise* the output range from +1 to –1 regardless of its input range. This is helpful if further layers of ‘Adalines’ are used (e.g., in a multi-layer neural network), it also standardises the output range, so we know what to expect. Any absolute value above 0.6 is sufficiently high to show a strong *classification effect*.

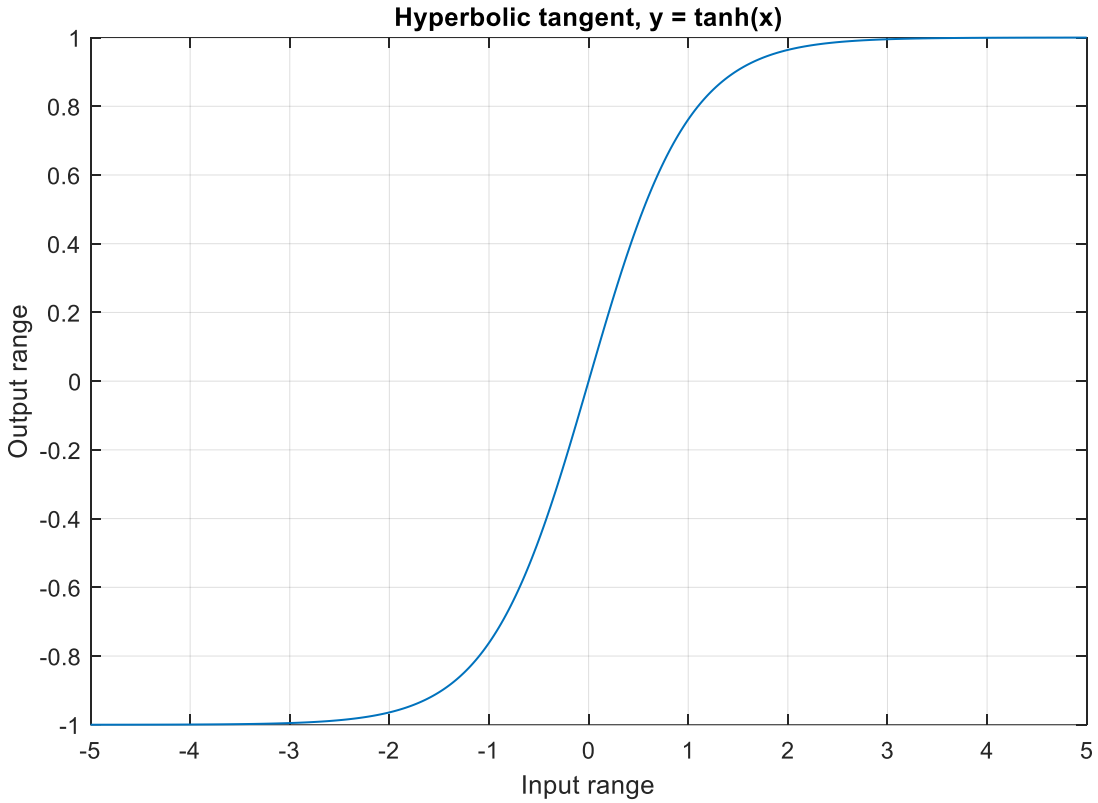

Figure 2: Hyperbolic tangent [y = tanh(x)] function compresses wide input ranges to ±1 outputs.

Now enter a 'T' pattern into the keypad by toggling the relevant buttons to show a 'T' like letter as shown below in Figure 3. Then press the blue pushbutton to train this pattern towards the positive class – again a value of `Sig = 0.7` or higher is good. Holding the button for a second or two should be enough to achieve this.

Figure 3: Enter a 'T' pattern using blue buttons on red background and then train by pressing "+1".

Since pressing either button invokes the LMS algorithm and adapts the internal *weight vector*, it is likely that the previous 'J' pattern output may have been affected (as they share several common 'input' buttons in their respective patterns). So, change the pattern back to the original 'J' shape by toggling the buttons once more to check. If the `Sig` output is not sufficiently negative, then press the red pushbutton once again to re-train the machine to achieve a negative output approaching –0.8 or thereabouts. It may also be a good idea to recheck the 'T' pattern – after a couple of checks back forth – the system should have correctly classified the two letter patterns correctly, i.e., the 'J' belonging to the negative (red) class and the 'T' to the positive (blue) class.

Translated Patterns

One can extend the patterns that the machine can recognise perhaps by *translating* the original patterns by one 'pixel' (again Widrow used this idea himself in his original Adaline machine). By this we mean to input the same 'J' and 'T' patterns but shifted horizontally by one column of keys to the right or left.

Figure 4: Enter a translated 'J' pattern and then train by pressing "–1".

Figure 5: Enter a translated 'T' pattern and then train by pressing "+1".

So, we could train the device to classify the translated ‘J’ also as the negative (red) class and the translated ‘T’ as the positive (blue) class. In doing so, we should check the original patterns once again and, through an *iterative* process of checking and retraining (if needed), it should be easily possible to match both ‘J’s to the -ve class (red) and both ‘T’s to the positive class (blue).

One could extend this experiment further – it should be possible to introduce *rotated* letters on the keypad and group them in the correct classes. So long as the patterns differ by at least one ‘pixel’ or button, then the machine can learn to distinguish them through sufficient training.

The authors would encourage users to experiment as much as they can with the device to get maximum understanding of the Adaline and LMS algorithms. Indeed, it would be interesting to receive any suggestions to improve or ameliorate the device in the future. Please feel free to contribute your own designs to the project.

V1.0