Perceptrons, Part 2 – Training

Eckel, TJHSST AI2, Spring 2024

Background & Explanation

Last week, we laid the groundwork – now it's time to train our perceptrons and see what they can, and can't, model.

A video explaining the perceptron training algorithm is linked on the course website.

The formulas mentioned in the video are:

To calculate current output: $f^* = A(\vec{w} \cdot \vec{x} + b)$

To update the weight vector: $\vec{w} = \vec{w} + (f(\vec{x}) - f^*) \cdot \lambda \cdot \vec{x}$

To update the bias: $b = b + (f(\vec{x}) - f^*) \cdot \lambda$

Remember to keep looping until either your target number of generations has been reached OR two consecutive epochs have the same identical outcome.

If that isn't clear, please re-watch the video!

A few important notes about this algorithm:

- I wasn't clear about this in the video, but it's possible for two consecutive epochs to have identical outcomes while the accuracy still isn't 100%. It just means that any changes during that epoch get reversed by later inputs. Bottom line: **AFTER THIS PROCESS COMPLETES, YOU MUST CHECK ACCURACY SEPARATELY. You can't just assume it's 100% correct.**
- In case you're curious – this will certainly converge to a 100% correct answer *if such an answer can be found*, but it *isn't* guaranteed to converge to the *best* answer if 100% isn't available. So, we can definitively answer "which Boolean functions are 100% reproducible", but not "what is the closest we can get to a Boolean function that isn't reproducible."
- Finally, don't forget that I mention 100 epochs being sufficient in the video. Don't go higher than that; it'll inflate your runtimes unnecessarily!

Required Task

Your goal is to write code that will determine how many *n*-bit Boolean functions can be perfectly modeled by a perceptron, for (in theory) any value of *n* (though *n* > 4 is impractical). So, your code should:

1. Take a number of bits as an argument.
2. Generate every truth table possible with that number of bits (ie, loop over each possible value for canonical integer representation for that number of bits, returning the truth table for each one.)
3. For each truth table:
    a. Start a perceptron model with a zero vector for weight and a zero value for bias and run the perceptron training algorithm until it completes, as shown in the video.
    b. **TEST THE ACCURACY** of the completed perceptron by looping over the truth table one more time and calculating the percentage of input vectors that are categorized correctly by your final perceptron.
4. Output the total number of possible functions, and how many of those truth tables could be 100% correctly modeled. For example, for *n* = 2, the answer should be "16 possible functions; 14 can be correctly modeled."

Find the answers for 3 bits and 4 bits. Four bits will take several minutes to run; don't be impatient! When you're done, **send your answers to me on Mattermost OR check them with another student who already knows the right answers and then tell me you've done so.** (So if you're working when I'm not available, you can still check with someone else.)

Get Your Code & Answers Ready to Submit

Make your code receive *two command-line inputs*. The first one will represent a number of bits and the second one will specify the canonical integer representation of a specific Boolean function. For example, 4 60800 would specify 4-bit Boolean function #60800. Your code should train a perceptron using the process described on the first page and, after 100 epochs or stability is reached, output:

1. The final weight vector
2. The final bias value
3. The accuracy of the perceptron as a decimal or percent

Specification

Submit **a single python script** to the link on the course website.

This assignment is **complete** if:

- You follow the instructions on the submission form to format your submission properly.
- Your code matches the specification above.