COMP135

Empirical/Programming Assignment 4

Behnam Heydarshahi

Neural Networks

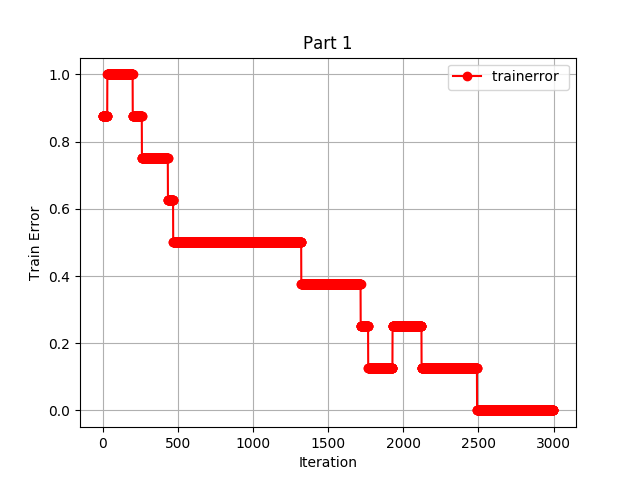
**Part 1: 838 dataset**

Below figure contains the trend of neural network's training error rate with increase of iteration.

General trend is, the more iterations, the more accurate the predictions. Additionally I made the following three observations:

(1) Error increase from .825 to 1 in when the iteration is smaller than 20. That is perhaps addressable to the fact that the network learns anything, it is guessing the same number, rather randomly, and chances of guessing one of the 8 output nodes randomly is 1/8. As the network tries to learn after 20 iterations or so, it looses the benefit of randomness, but soon after 200 iterations, the benefits of making a learned guess start to show.

(2) The error rate changes in steps. That is because we have a limited number (8) of input data. Therefore, when a new value is learned, the error rate drops by 1/8.

(3) There is a surge in error near 1750 iterations. One theory is that the network starts to learn a better compression rule at that point, where it drops the current rule, which hurts the accuracy at first, but gains it later on. At 2500+ iterations the predictions are always correct.

**Hidden Units Representation:**

Below is the representation my code prints.

[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0] --> 0.183218139209 0.256381555915 0.975508515699 --> 1

(0x1)

[0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0] --> 0.977969925039 0.424915809918 0.0691604695012 --> 2 (1x0)

[0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0] --> 0.529877611108 0.975555069763 0.0505748912021 --> 3

(x10)

[0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0] --> 0.378697489384 0.361981334259 0.336551316341 --> 4

(xxx)

[0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0] --> 0.031148699745 0.969650075395 0.83674947487 --> 5

(011)

[0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0] --> 0.329056011223 0.356633608434 0.410168341984 --> 6

(xxx)

[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0] --> 0.957579040378 0.0268982899954 0.92707061013 --> 7

(101)

[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0] --> 0.392479560595 0.332274864132 0.359044383485 --> 8

(xxx)

Even though my code reaches 100% accuracy, and while some encodings are clear, such as 101 for 7 and 011 for 5, my code does not reflect the binary representation the way the example from class does. I marked anything between 0.25 and 0.75 as unknown (x), below 0.25 as 0, and above .075 as 1. That said, if we want to force ourselves to find a correlation with binary representation, one way would be to normalize the values to a different base and roof other than 0 and 1.