Paul Hernandez

CS 360 - Project 1: Part 1

10/1/2019

### Task 1:

1. Given a baseline of 0.01 as the alpha value, we can observe that as the number of training increases, the difference between the expected outputs and the actual outputs decreases. But takes approximately epoch > 2296009 for the outputs became close to the expected output [0.00 0.01 0.01 0.99].

Increasing the alpha value by a factor of 10 (alpha value = 0.10) results in the same decrease in difference between the expected outputs and the actual outputs. However, it achieved actual output at a quicker rate. Specifically once epoch reaches > 1232983, achieved expected output [0.00 0.00 0.00 1.00]. This is approximately 60% increase in the rate of difference change compared to an alpha value of 0.01. From this, we can see that as the learning rate increases, the rate of the difference change increases.

1. The problem appears to be that the outputs remain the same despite increasing epoch numbers, and does not “learn” to output the expected [0 1 1 0] for an XOR function. Specifically the outputs remain at [0.50 0.50 0.50 0.50] with an alpha value of **0.1**. And if we attempt a larger alpha value (increase the learning rate), say **10.0,** the programresults in an output of [0.50 0.03 0.83 0.11] which also reflects no difference change in output as epoch increases.

### Task 2:

Hyper-parameters: {*alpha, hiddenLayerSize, numEpochs*}

Desired output: {0, 1, 1, 0}

Execution time was calculated using time ./task2

|  |  |  |
| --- | --- | --- |
| Hyper-parameters | Output | Execution Time |
| Initial: {0.1, 5, 10,000} | 0.088 0.93 0.92 0.065 | 0.257s |
| **1**: **{0.7, 4, 10,000}** | 0.027 0.98 0.98 0.022 | 0.235s |
| **2: {1.0, 8, 10,000}** | 0.014 0.99 0.98 0.017 | 0.238s |
| **3: {2.0, 6, 10,000}** | 0.011 0.99 0.99 0.013 | 0.225s |

For this neural network, we define an efficient training time to be less than 5 seconds and the reliability to be max(actual output - expected output) < 0.1. From this, we can take a look if the hyper-parameters above satisfy these conditions.

|  |  |  |
| --- | --- | --- |
| Set | Difference in output | Difference in time vs. initial |
| **1**: **{0.7, 4, 10,000}** | {0.027 0.02 0.02 0.022} | - 0.022s |
| **2: {1.0, 8, 10,000}** | {0.014 0.01 0.02 0.017} | - 0.019s |
| **3: {2.0, 6, 10,000}** | {0.011 0.01 0.01 0.013} | - 0.032s |

I choose to keep the number of epochs constant (10000) as increasing them almost always led to an increase in training time with small alpha values < 0.5 and a hiddenLayerSize of 5. However, as you can see, modifying both alpha values and the hiddenLayer values led to significant differences.

Data Set 1 saw an increase in learning rate by 0.6 and decreased the hidden layers by 1. This achieved a 0.022s faster training time than the initial set and led to > 98% accuracy.

Data Set 2 saw an increase in learning rate by a factor of 10 and increased the hidden layers by 3. This achieved a 0.019s faster training time than the initial set and led to > 99% accuracy.

Data Set 3 saw an increase in learning rate by a factor of 20 and increased the hidden layers by 1. This achieved a 0.032s faster training time than the initial set and led to > 99% accuracy. This was the most efficient and reliable set of parameters.

### Task 3:

1.

Questions to ask OH:

1. Is there a reasonable range of learning rates? What it is considered too small? Too big?
2. I get why a single preceptron can’t implement an XOR model, but why would it produce the same outputs as shown above (despite playing around with the learning rate)?
3. Task 2: What counts as ‘small training time’?