Knicholas Kennedy
Yaxiaer Atajiang

CMSC 409
Project 3

1.
   The input for this neuron training is time. We simply apply different weights to it to manipulate it into a nonlinear function.
   The output for this problem set is energy consumption. We are attempting to track energy consumption based on the time of day.

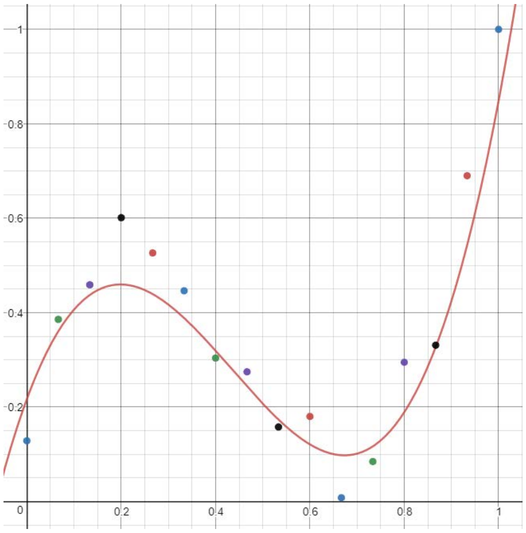
2. The activation function of our decision unit should be a cubic function. Therefore, it should be the third architecture from the problem. I normalized my data, so the activation function ended up being 7.772111x^3-10.246754x^2+3.188755x+0.198393.

3. We did our training algorithm in the code, but I used the perceptron training method because I understood this one the best. I wanted to use the least mean square method but couldn't get it working in time. R-Scores on training data are as followed:
   Cubic Functions:
      Day 1: 0.9243280901909569
      Day 2: 0.9244920863407764
      Day 3: 0.956591677567673
      Day 4: 0.9056239043482227
   Quadratic Functions:
      Day 1: 0.09367511696897057
      Day 2: 0.16284418788729027
      Day 3: 0.029859081850864277
      Day 4: 0.04430524919364398
   Linear Functions: My linear R scores were so trivial as to be zeroes. I mean literally 0.000000004s and such. All of them were terrible.

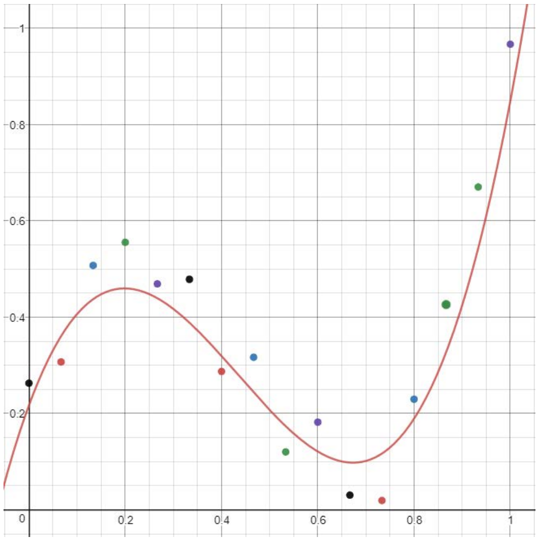
4. I used a learning constant of 0.3 and 10,000 iterations. I also normalized the input data, and normalized the weights using the formula presented in the Kohonen Slides. I selected 0.3 because based on tests, this gave me the highest R Score available. I also tried 0.5, 1.0, 2.0, 0.03, 0.01, and 0.05. None of them worked as well as 0.3 for my test data. I also just believe that this is a well-rounded number for learning constants because it allows the unit to learn fast without heavy jumps. I chose 10,000 iterations simply because it was enough to present predictable data without stressing a PC.
5. Yes, a network of neurons would give us a lower error reported because We would have more accurate measurements on each slope of the line. A cubic function is complex, and if you separate this into several linear layers, you reduce complexity on the algorithm, thus producing cleaner results for slightly more overhead.

Cubic Functions

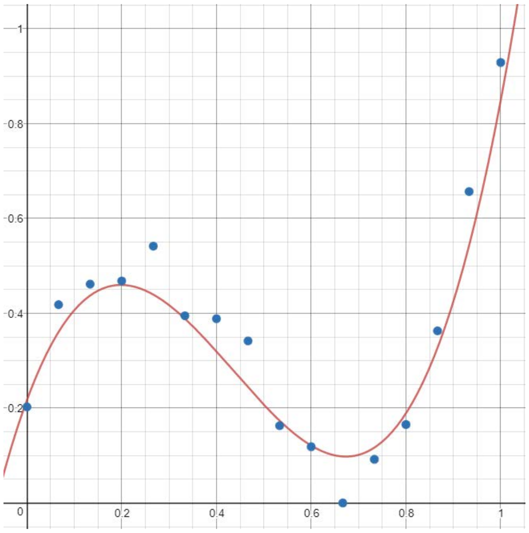
Day 1



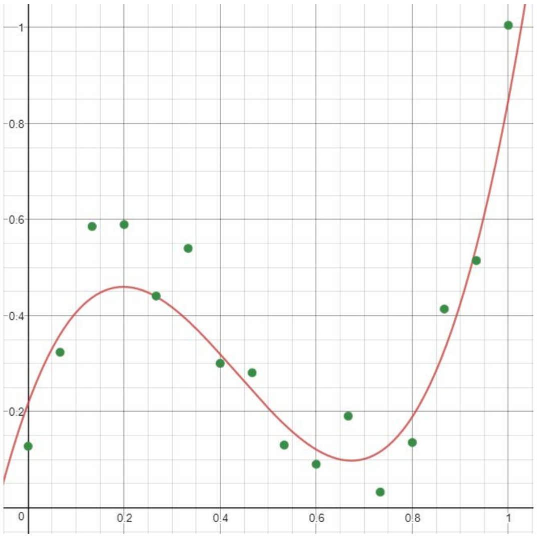
Day 2



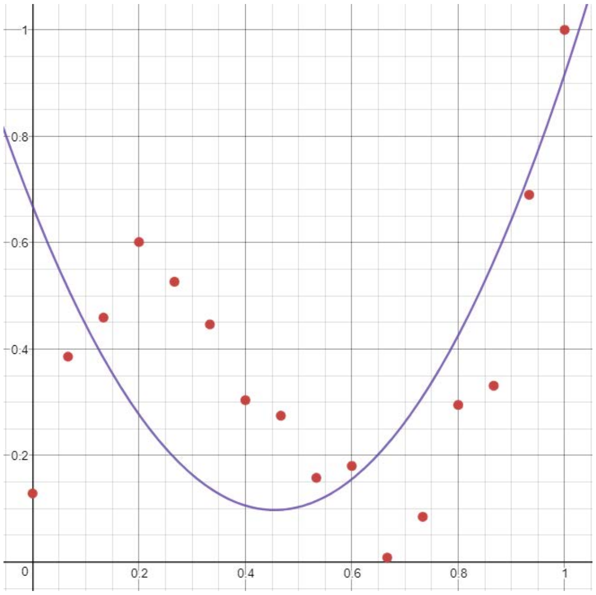
Day 3



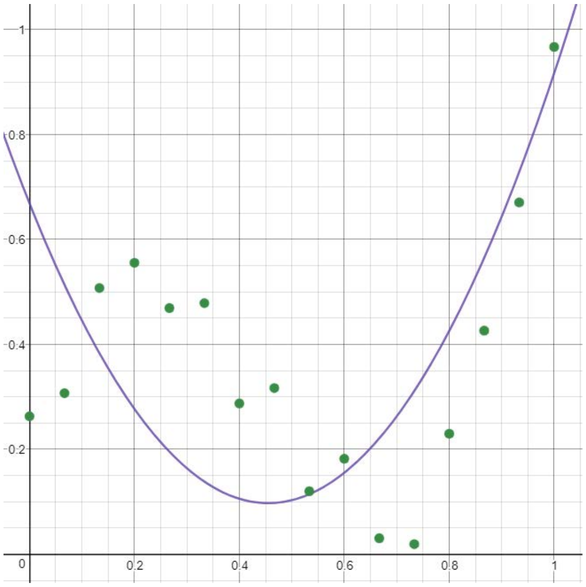
Day 4



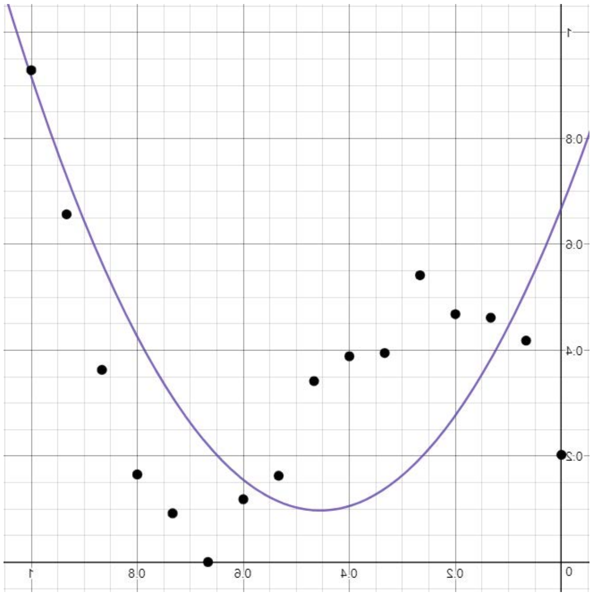
Quadratic Day 1



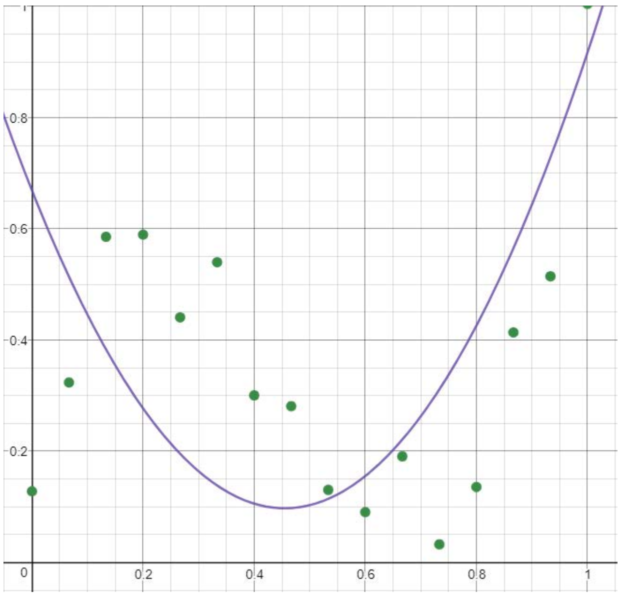
Day 2



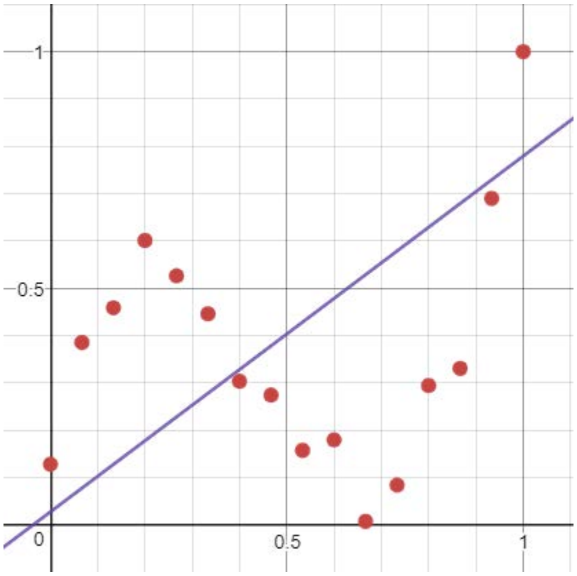
Day 3



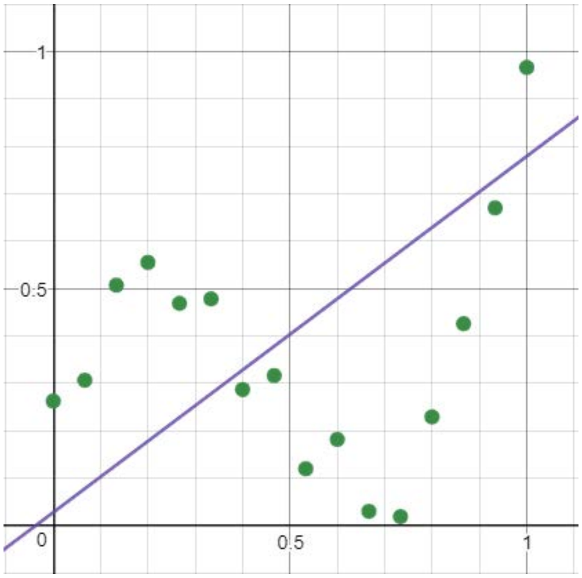
Day 4



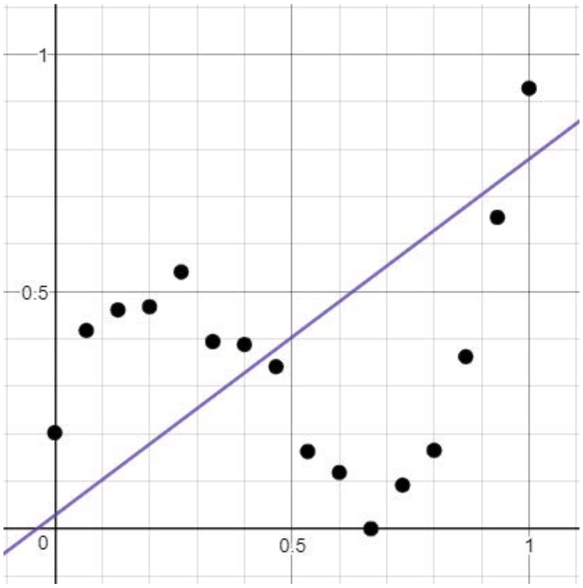
Linear Day 1



Day 2



Day 3



Day 4