Subhan Ali

Kameron Bray

PART ONE

**TRAINING and TEST data sets**

They are in the attached files.

**DISCUSSION OF RESULTS**

The settings we used for the neural network configuration are as follows:

#define NumOfCols 6

#define NumOfRows 40

#define NumINs 3

#define NumOUTs 2

#define LearningRate 0.05

#define Criteria 2.5

#define TestCriteria 5

#define MaxIterate 10000000

#define ReportIntv 1000001

#define Momentum 0.9

#define TrainCases 50

#define TestCases 10

// network topology by column -----------------------------

#define NumNodes1 4

#define NumNodes2 15

#define NumNodes3 30

#define NumNodes4 40

#define NumNodes5 20

#define NumNodes6 2

**SUMMARY OF RESULTS**

* Our neural network learned the training set. It converged to within the criteria of 2.5. The precision was therefore within 2.5% of the actual average and median values. The percent of training cases that met the criteria was 2.0.
* The training took quite a while. It took 558163 backpropagation iterations.
* The neural network generalized quite well, to within a 5% error margin. The percent of testing cases that met this precision criteria was 2.0.
* I think the neural network "learned" the problem of finding the average and median of a set of three numbers quite well.

PART TWO

**TRAINING and TEST data sets**

They are in the attached files.

**DISCUSSION OF RESULTS**

* Our problem focuses on the mechanical properties of physics. We simulated a rubber band slingshot causing an object to fly a certain amount of distance. We used parameters including the rubber band's constant, the angle of launch, the object's mass, and the distance the rubber band was pulled back. Our dependent variable was the distance the object traveled.
* We developed our training and test data sets in Python using a random number generator. The constant was between 100 and 400 (inclusive), the angle was between 25 and 70 degrees (inclusive), the distance was between 2 and 5 meters (inclusive), and the mass was between 1 and 5 kilograms (inclusive). We used the same methods for creating training data as we did test data, and the random number generation method turned to work out well!
* The settings we used for the neural network configuration are as follows:

// Learning Settings

#define MaxIterate 10000000

#define LearningRate 0.05

#define ReportIntv 100001

#define Momentum 0.9

// Architecture Settings

#define NumOfCols 4

#define NumOfRows 25

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4 Inputs - mass, angle, constant, pulled\_distance (ALL FLOATING POINT NUMBERS)

1 Output - Distance traveled

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define NumNodes2 10

#define NumNodes3 20

#define NumNodes4 25

#define NumNodes5 1

#define NumNodes6 1

#define NumOUTs 1

#define NumINs 4

#define NumNodes1 5

#define TrainCases 50

#define TestCases 10

#define Criteria 7.5

#define TestCriteria 15

In terms of criteria, since we figured that 2.5 and 5 were good training and testing criteria, respectively, for numbers whose output ranged between 1 and 100, we noticed that the outputs for our personal problem was between 25 and 350, or a range of about 300. So, since the range tripled, we decided for our criteria range to be tripled as well, thus the 7.5 and 15.

* The topologies we tested were:
  + 4-3-2-1
  + 4-2-2-1
  + 4-2-1
  + 4-10-10-10-1
  + *4-10-20-25 (this one turned out to be the most successful)*

**SUMMARY OF RESULTS**

* Our neural network learned the training set. It converged to within the criteria of 7.5. The precision was therefore within 7.5% of the actual, physics-calculated distance traveled values. The percent of training cases that met the criteria was 1.0.
* The training took quite a while. It took 815516 backpropagation iterations.
* The neural network generalized quite well, to within a 15% error margin (see above for why a testing criteria of 15 was chosen). The percent of testing cases that met this precision criteria was 1.0.
* I think the neural network "learned" the problem of finding the distance traveled from a slingshot quite well.