Assignment 2 - CS 4375

Aadi Kothari
Andrew Su

I. Introduction

Our project is in the domain of robotics. The data contains robot movement based on ultrasonic sensor readings. The robot was capable of making one of four moves based on the distances read from the sensors. Since this was a binary classification project, we decided to do four independent binary classifications that would determine which move was made.

II. Error Log

These were the error vectors based on random_state = 3 in the training split. Our algorithm outputs a 4-tuple corresponding to four possible moves: Slight-Left-Turn, Move-Forward, Slight-Right-Turn, Sharp-Right-Turn. An error vector is the sum of the corresponding values over all the instances in the test split.

Error Vector			
Learning Rate	Sigmoid	tanh	ReLu
0.1	[218 83.5 28.5 216]	[364 463 528 332]	[218 83.5 28.5 216]
0.01	[2.76 1.57 28.5 1.66]	[328 462 517 330]	[218 83.5 28.5 216]
0.001	[7.73 4.20 1.60 3.53]	[66.2 176 159 25.5]	[218 83.5 28.5 216]
0.0001	[29.2 13.2 25.0 10.7]	[27.2 20.7 7.22 12.1]	[9.11 83.5 2.95 22.6]

III. A Closer Look at Output Data

When post-processing is applied to the output, vectors of 4 floating point values are converted to directional moves. The actual outputs of the data correspond to one-hot vectors. Here is a sample of the output.

```
Predicted Motion      Actual Motion          Predicted Vectors          Actual Vectors

Sharp-Right-Turn      Sharp-Right-Turn        [0.    0.    0.013 1.   ]    [0. 0. 0. 1.]
Move-Forward          Move-Forward            [1.    0.    0.005 0.   ]    [1. 0. 0. 0.]
Move-Forward          Move-Forward            [0.955 0.    0.002 0.029]    [1. 0. 0. 0.]
Move-Forward          Move-Forward            [1. 0. 0. 0.]                [1. 0. 0. 0.]
```

Sharp-Right-Turn	Sharp-Right-Turn	[0.051 0. 0. 0.971]	[0. 0. 0. 1.]
Sharp-Right-Turn	Sharp-Right-Turn	[0. 0. 0. 1.]	[0. 0. 0. 1.]
Sharp-Right-Turn	Sharp-Right-Turn	[0. 0. 0. 1.]	[0. 0. 0. 1.]
Slight-Left-Turn	Slight-Left-Turn	[0.013 0. 0.979 0.]	[0. 0. 1. 0.]
Sharp-Right-Turn	Sharp-Right-Turn	[0. 0. 0. 0.995]	[0. 0. 0. 1.]
Sharp-Right-Turn	Sharp-Right-Turn	[0. 0. 0. 1.]	[0. 0. 0. 1.]
Slight-Right-Turn	Slight-Right-Turn	[0.001 0.999 0. 0.]	[0. 1. 0. 0.]
Move-Forward	Move-Forward	[1. 0. 0. 0.]	[1. 0. 0. 0.]

This shows that the error calculated by summing floating point values in the vector (i.e. [0.013, 0, 0.979, 0] vs [0, 0, 1, 0]) is negligible since the direction can be unambiguously determined.

IV. Comparing Activation Functions Using Move-Prediction Accuracy

In post-processing, we mapped the 4-tuple output values back into move classes based on which element of the vector was the highest. The following table shows number of incorrect classifications:

Incorrect Move Predictions (out of 910 instances)			
Learning Rate	Sigmoid	tanh	ReLu
0.1	656	674	656
0.01	66	771	656
0.001	15	480	656
0.0001	73	56	182

A learning rate of 0.001 combined with the Sigmoid function produced the lowest error. ReLu performed the worst. We think Sigmoid performs best because it has restrictions with a minimum value at 0 and maximum at 1. Unlike tanh with a -1 minimum. This prevents the weights of certain nodes from dominating others. For the same reason, ReLu would perform poorly since it has no maximum. Early in our algorithm design ReLu would cause some weights to overflow. Six neurons in the hidden layer gave the best results in our testing.

V. Fine Tuning and Overtraining

We decided to adjust the number of iterations to produce the highest number of correct moves. Since we have no regularization in our algorithm, we expected it to over-learn with enough iterations. However, after 100,000 iterations we were still getting lower errors. We could not feasibly train more iterations as it would take too long on the machines we used.

Sigmoid Activation Function					
Iterations	1,000	5,000	10,000	100,000	1,000,000
Incorrect Moves	25	11	7	4	?

VI. Analysis and Conclusion

With such high prediction accuracy, we checked to make sure our training data was not tainted with test data and vice versa. Our training and test data was indeed distinct. We observed that the sensor readings are taken just fractions of a second apart so many instances are extremely similar, which allowed us to get high prediction accuracy even on unseen test data.

We anticipate that with more computing power, we could achieve even lower error, as we had not yet reached the point of overtraining. We were overall satisfied with the results as the neural net could predict the correct movement of the robot in 99.6% (906/910) of instances using readings from just 4 sensors.