// CS570 Project 3a

// Heyan Huang Apr 3, 2013

Artificial neural network for a moonlander

Abstract

In this project, an artificial neural network nonlinear regression model was created, trained and tested for a monlander program. All programs were implemented in C++. The training data (>270,000 groups of input and output data) were collected using a tricky generation method to train the model. A feed-forward neural network with 3 layers(6+4+2) was created. It was trained by using back-propagation algorithm with the training data. A logistic function was used as activation function. The neural network was successfully trained (MSE < 0.000,01). A new test data was used to test the neural network and the successful rate was significantly improved.

Algorithm

1) Training data collection

Since the training data (6 inputs except the acceleration variable and 2 outputs) was not provided, I figured out a tricky way to collect a bunch of data.

At all the beginning of the landing (when height = 100), I kept 4 exploratory variables, xPosition, height, Xvelocity and fuel, the same, while generated the other two independent variables, Yvelocity and wind, randomly in ranges. And instead of using burn and thrust as output, still, I randomly generated these two “dependent”variables used as input, followed with a bunch of status updates and landing tests, all the 8 variables are used as inputs to test if the moonlander has landed the specific planet successfully. Since the Neural Network lacks the ability to handle categorical variables, I collected only successfully landed observations including processing status data to be used as training data for the ANN model to goal at successful landing

To better cover all possibilities, the initial inputs are created as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| xPosition | height | Xvelocity | Yvelocity | wind | fuel |
| 0  0  0  0  .  .  .  0 | 100  100  100  100  .  .  .  100 | 0  0  0  0  .  .  .  0 | 0  0.01  0.02  0.03  .  .  .  10 | -0.1  -0.099  -0.098  -0.097  .  .  .  0.1 | 100  100  100  100  .  .  .  100 |

Table 1 Initial input data for training

Two outputs (burn and thrust) were then produced under the principle that keeps Yvelocity low and keeps xPosition around 0. After burn and thrust were obtained, inputs in next update were calculated by using function update(). Note that wind was randomly changed with every update. Only the data that lead to a successful landing were chosen for later training. No less than 270,000 groups of data (6 inputs and 2 outputs) were collected.

2) Artificial neural network (6+2+2)

A feed-forward network with 3 layers was created. The first layer is input layer which simply holds for the input data vector. It has 6 units. The second layer is a hidden layer which has four units. The third layer is output layer with two units. See figure1 below for the structure of the neural network.

Figure 1. Structure of neural network

The units in hidden layer and output layer are sigmoid perceptron, which use logistic function as activation function.

I chose one hidden layer because by adjusting the number of units in the hidden layer I can already achieve the “bump”figure shown in textbook (P732 Figure 18.23 (b) ), which is the kind of shape in my mind for the moonland successful landing pathes. And I was targetted to adjust the number of hidden layer units in range [4, 8] to mimick the highest successful landing rate. And I have applied bias unit in each non-output layer to tuning my model slightly better.

For the most especially important part, the initial weight matrix for ANN could be a deterministic factor determine whether the ANN can eventually converge or not. Since we have two outputs, burn and thrust, and each output needs one complete ANN model, and the final weight matrix could be completely different, I would better build two model objects one for each, which share the same CBackProp class data structure but with different initial weights to benefit separate models.

3) Training algorithm

Back-propagate algorithm, which is the same as the one in the textbook ([1], page 734), was used for training and tuning weights. A CBackProp class coded using C++ was refered and downloaded from website, with more details in reference and attached codes. This data structure was modified to create and train the ANN model for the moonlander project.

The learning rate was set as 0.3 and the momentum parameter was 0.1.

The training data I collected at very beginning was used and scanned to tune the weights, which was initialized referring to the correlation between exploratory and dependent variables. The minumum of mean square error (MSE) was used as the criteria to tune the weights. The threshold for MSE was set to be 0.000,001 to stop the training. And the 270,000 observations in training data were scanned more than once trying to achieve ANN model convergence.

4) Neural network testing

To validate the trained models performance, the models were tested together on a newly generated test dtaset with a reasonable observation count. I could have used the same method to generate test and validation dataset as I did with training dataset. But considering our final goal was to land the planet successfully, the “output” varaible values for burn and thrust were not necessary, neither was MSE criteria, and it was the successfully landing rate which matters. So I simply generated the 6 input variables completely randomly with bounded variable value range.

Based on the test observation input data vector, burn and thrust were predicted by the previous trained ANN models. These status variables then would be updated and tested if it landed successfully. If the height was greater than 0 (having not landed yet), control (newly predicted burn and thrust based on updated 6 input variables using ANN models) would be applied, and repeated the same process until either landed or crashed.

The number of successful landing (including process status updates, the same as did on training dataset) was counted and used to calculate the successful landing rate.

Results

1) Training data collection

270,387 groups of training data were collected and stored in a dynamic 2-dimensional array (figure 2). All these data were from successful landing.

Figure 2 The training and testing of the artificial neural network

2) Network training

Since there are two outputs in the neural network, the training data were used twice for different outputs. For the output of burn, 16809 groups of training data were used while 21748 groups of training data were used for thrust to achieve the convergence. Both training are successful (figure 2).

3) Network testing

12,550 groups of data were used for testing. They were then updated while the moonlander is landing. Output of burn and thrust were produced by network. The number of success landing were counted and used to calculate the success rate. From figure 2, the success landing rate of the ANN is 100%.

Based on the results, the training is very successful. However, I spent lots of time to figure out how to collect the training data. It is critical to consider most possibilities of initial state of the moonlander. Since the output of last update will be used as input of next update, it might be good to use recurrent instead of feed-forward network for this project.

Conclusion

A feed-forward artificial neural network with 3 layers (6+2+2) was created. More than 40000 groups of training data were collected. Using back-propagation algorithm and the training data, this neural network was successfully trained for the control of a moonlander. 2100 groups of test data were used to test the network and the success rate is 100%.

1. Russell, S.J. and P. Norvig, Artificial intelligence: a modern approach. 2010: Prentice Hall.