

Rough report on Voice-Controlled bot using own neural nets

Raktim Gautam Goswami and Abhishek Bairagi

November 27, 2018

Abstract

This is a rough report on our current work in making the voice-controlled bot using our own neural networks.

1 Making the bot body

The chassis, motors and wheels are assembled to make the basic body ready. The motors are connected to the motor driver and motor driver is connected to arduino board on the chassis. The arduino board is in turn connected to a bluetooth module (HC-05) for bluetooth communication.

2 Building the neural network

2.1 Theory

We have used linear regression in our model. Here, all the features are tried to be approximated using an n-dimensional straight line (n being the number of features). The equation used for this is

$$\sum_i W_i * x_i + b \quad (1)$$

In matrix form it is

$$out = W.X + B$$

The output(out) is then put as input to the sigmoid function and the output of it is a number scaled between 0 and 1. This is the actual output(Y') we are interested in. The sigmoid function is defined as

$$sigmoid(x) = 1/(1 + exp(-x))$$

The cost function is then calculated using mean squared error as

$$J = 0.5 * (Y - Y')^2$$

Gradient descent algorithm is used to get minimum error using the derivative of the error(J) with respect to weight (W). This process is carried on for a number of times to get the best accuracy.

How is the descent algorithm obtained from the cost function?

We initialized the parameters W1 and b. Now we want Mean Square Error function to be minimum. The way we do this is by taking the derivative (the tangential line to a function) of our cost function with respect to each parameter. Derivative at that point and it will give us a direction to move towards. And then we update the value of all the parameters according to the derivative obtained. And then we iterate the process (number of iterations are decided by us). We make steps down the cost function in the direction with the steepest descent. The size of each step is determined by the parameter α , which is called the learning rate. The gradient descent algorithm is repeated until convergence: $M_j := M_j - (learningrate) * (deltaLoss) * input$

2.2 Python code

<https://github.com/raktimgg/ML-algorithm-for-speech-recognition/blob/master/code.py>

This is the full code that is used for training. The accuracy we are getting is around 98 percent.

2.3 Dataset

We have made our own dataset by recording 25 samples of each word. Each of these samples are recreated by adding empty elements in the front and back in many different combinations to create a dataset of 6250 samples for each word. All the audio files are imported to an array in the code and converted to mfcc format before training. For creating training dataset we recorded 25 audio file of each of the following word -

- 1)Forward
- 2)Left
- 3)Right
- 4)Back
- 5)Stop

The code for generating 6250 samples for each word from 25 samples can be found in the github link attached.

<https://github.com/abhishekbairagi/Making-Dataset-for-ML/blob/master/250files.py>

3 Transferring the weights to Raspberry Pi (Yet to be done)

The weight(W1 and B) are saved in a file at the end of the code. These weights will be transferred to the raspberry pi and a simple program written, will record audio on the raspberry pi, do the calculations using the weights and predict the text output. This output will be sent,using bluetooth, to the toy car, which will move accordingly.