Voice Recognition through Machine Learing

Raktim Gautam Goswami[1], Abhishek Bairagi[2] & G V V Sharma[3]

Contents

***Abstract*—This manual shows how to develop a voice recognition algorithm and use it to control a toycar.**

1 Dataset

1.1 Record 'forward' 80 times using you phone and save as 'forwardi.wav' for $i = 1, \dots, 80$.

1.2 Repeat by recording 'left', 'right', 'back' and 'stop'. Make sure that the audio files for each command are in separate directories. Download the following directory for reference

```
svn checkout https://github.com/gadepall/
    EE1390/trunk/AI−ML/audio_dataset
```

1.3 Use the following script to generate a dataset for 'back' command. Explain through a block diagram.

```
https://raw.githubusercontent.com/gadepall/
    EE1390/master/AI−ML/codes/250files.py
```

Solution: to generate the dataset needed for training. The following diagram explains how this is done for the back command.

back (80 files) $\xrightarrow{250files.py}$ $25000 files.$

1.4 Suitably modify the above script to generate similar datasets for 'left', 'right', 'stop' and 'forward'.

1.5 Store the complete dataset in a directory and run **code.py** from within the directory. Note that this should be done on a powerful work-station. This will generate two files **W1.out** and **b.out**.

2 Implementation

2.1 Execute **record.py** and issue any of the commands 'forward', 'left', 'right', 'back' and 'stop'. The output will be as per Table **??**.

2.2 Install Google API "Arduino Bluetooth Controller" using google play-store

2.3 Open the app and connect to HC-05.

2.4 Open voice control section in the app and tap to give following commands.
Left, Right, Forward, Back & Stop

3 Building the neural network

3.1 Problem Statement

3.1.1 Consider $\mathbf{x}$ be 4043×1 to be human voice issuing either 'forward', 'left', 'right', 'back' and 'stop'. Let $\mathbf{W}$ be 4043×5 and $\mathbf{b}$ be 5×1. $\mathbf{W}$ and $\mathbf{b}$ are the machine parameters. Then the machine makes a decision based on

$$\hat{\mathbf{y}} = \mathbf{x}^T \mathbf{W} + \mathbf{b} \tag{3.1}$$

3.1.2 The problem is to estimate $\mathbf{W}$ and $\mathbf{b}$. This is done by considering

$$\min_{\mathbf{W},\mathbf{b}} J = \frac{1}{2} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 \tag{3.2}$$

The authors are with the Department of Electrical Engineering, Indian Institute of Technology, Hyderabad 502285 India . e-mail: 1. ee17btech11004@iith.ac.in, 2. ee17btech11051@iith.ac.in, 3. gadepall@iith.ac.in

3.2 *Solution: Gradient Descent*

3.2.1

3.3 *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

$$\sum_i Wi * xi + b \tag{3.3}$$

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.

a) 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 itterations 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: $Mj := Mj - (learningrate) * (deltaLoss) * input$

3.4 *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.

3.5 *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 cobinations 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/bl

4 Transfering 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.