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 in your laptop and save as 'forwardi.wav' for $i = 1, \ldots, 80$.

1.2 Record files by entering this command in terminal - "arecord filename.wav" and press Ctrl+C to stop recording.(make sure your recorded audios are less than 3 sec of duration.)

1.3 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

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

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

```
https://github.com/raktimgg/Voice−recognition
    −bot/blob/master/AI−ML/codes/25files.py
```

Solution: to generate the dataset needed for training. The following diagram explains how this is done for the back command.
back (80 files) $\xrightarrow{25files.py} 2000 files.$

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

1.5 After creating 2000 back files rename all the files in the format 'backi.wav' for $i = 1, \ldots, 2000$. Use the given link for renaming.

```
https://github.com/raktimgg/Voice−recognition
    −bot/blob/master/AI−ML/codes/
    rename_back.py
```

1.6 Suitably modify the above script to generate similar datasets for 'left', 'right', 'stop' and 'forward'.So now we have 10000 audio files in total.

1.7 Store the complete dataset in a directory and run **code.py** from within the directory. Note that this should be done on a powerful workstation.

```
https://github.com/raktimgg/Voice−recognition
    −bot/blob/master/AI−ML/codes/code.py
```

1.8 This will generate two files**W1.out** and **b.out**. Save the files in same directory.

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 the following table.

back	0
forward	1
left	2
right	3
stop	4

2.2 Now Save the files 'W1.out' and 'b.out' in raspberry pi.

2.3 Implementation on raspberry pi.(yet to be done)

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 1×5.

W and **b** are the machine parameters. Then the machine makes a decision based on

$$\mathbf{y1} = \mathbf{x}^T\mathbf{W} + \mathbf{b} \tag{3.1}$$

3.1.2 Now, apply the sigmoid function to all the elements of the output matrix (y1) to scale the values between 0 and 1.

$$\hat{\mathbf{y}} = 1 \div (1 + exp(-\mathbf{y1})) \tag{3.2}$$

3.1.3 The problem is to estimate **W** and **b**. This is done by considering the error(cost) function,

$$J(\mathbf{W}, \mathbf{b}) = \frac{1}{2}\|\mathbf{y} - \hat{\mathbf{y}}\|^2 \tag{3.3}$$

3.1.4 We need to find the minimum of error function for optimising the equations.

3.2 Solution: Gradient Descent

3.2.1 **W** and **b** can be estimated from (3.3) using

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \frac{\alpha}{2}\frac{\partial J(\mathbf{W},\mathbf{b})}{\partial \mathbf{W}}\mathbf{W}(n) \tag{3.4}$$

$$\mathbf{b}(n+1) = \mathbf{b}(n) - \frac{\alpha}{2}\frac{\partial J(\mathbf{W},\mathbf{b})}{\partial \mathbf{b}} \tag{3.5}$$

Show that (3.4) can be expressed as

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \alpha\Big[\mathbf{x}^T(n)\mathbf{x}(n)\mathbf{W}(n) + \mathbf{x}^T(n)\mathbf{b}(n) - \mathbf{x}^T(n)\mathbf{y}(n)\Big] \tag{3.6}$$

$$\mathbf{b}(n+1) = \mathbf{b}(n) - \alpha\left[\mathbf{x}\mathbf{W} - \mathbf{b} - \mathbf{y}\right] \tag{3.7}$$

Solution: From (3.3) and (3.2),

$$J(\mathbf{W},\mathbf{b}) = \frac{1}{2}\|\mathbf{y} - \hat{\mathbf{y}}\|^2 \tag{3.8}$$

$$= (\mathbf{x}\mathbf{W} + \mathbf{b} - \mathbf{y})^T(\mathbf{x}\mathbf{W} + \mathbf{b} - \mathbf{y}) \tag{3.9}$$

$$= \left(\mathbf{W}^T\mathbf{x}^T + \mathbf{b}^T - \mathbf{y}^T\right)(\mathbf{x}\mathbf{W} + \mathbf{b} - \mathbf{y}) \tag{3.10}$$

$$= \mathbf{W}^T\mathbf{x}^T\mathbf{x}\mathbf{W} + \mathbf{W}^T\mathbf{x}^T\mathbf{b} - \mathbf{W}^T\mathbf{x}^T\mathbf{y} \tag{3.11}$$

$$+ \mathbf{b}^T\mathbf{x}\mathbf{W} + \mathbf{b}^T\mathbf{b} - \mathbf{b}^T\mathbf{y} - \mathbf{y}^T\mathbf{x}\mathbf{W} \tag{3.12}$$

$$- \mathbf{y}^T\mathbf{b} + \mathbf{y}^T\mathbf{y} \tag{3.13}$$

Using

$$\frac{\partial}{\partial \mathbf{W}}\mathbf{W}^T\mathbf{x}^T\mathbf{x}\mathbf{W} = \tag{3.14}$$

3.3 Dataset

3.3.1 Record as many audio files of each of the words given below.
1)Forward
2)Left
3)Right
4)Back
5)Stop

3.3.2 Recreate these by adding empty elements in the front and back in many different cobinations to create a dataset.

3.4 Training

3.4.1 Import these soundfiles to an array in a python program using $soundfile$ library and convert to mfcc format using $pythonspeechfeatures$ library.

3.4.2 Extend the program to train the data, taking help from *1.6*. Also add a function to test the accuracy.

3.4.3 Note the accuracy.

3.4.4 Add to the program, some code to store the generated weights(W1) and biases(b).