Voice Recognition Bot

Ganni Mahima
EE19BTECH11033

January 22, 2020

In order to increase the number of samples for training ,we import the 80 samples of each command we recorded to an array and generate 25 corresponding sound files for each original sound file by adding empty elements at the front and the back in various combinations.

Code for creating 25 files for each of the 80 audio files

The mfcc extrats the features from these audio files.This uses 39 features which would be indepedent of each other.(where some of them are related to the frequencies and amplitudes).
mfcc represents each audio file as a two-dimensional array or a feature vector (49,39) where 49 is the number of time steps or number of frames and 39 is the number of mfcc features.

```
back, sr = sf.read("back__"+str(i)+"_"+str(j)+".wav")
#     back, index = librosa.effects.trim(back)
x=mfcc(y = back, sr = sr, n_mfcc=39)
i=49-x.shape[1]
tp=np.zeros((39,i))
x=np.append(x,tp,axis=1);
data.append(x.T)
label.append(0)
```

Converting into a 3d-array

- Here we are converting that into a (49,39).
- This process continues for all the 10,000 files ,each time each feature vector getting stacked ,thus creating a 3d-array of dimensions (9480,49,39).
- Also label[] which was a two dimensional array is converted to a binary matrix.

Basically here there is a neural network which takes the various inputs given to it,performs a function on them and then applies the activation function to give out the output.

$\mathbf{z}=x^T\mathbf{W}+\mathbf{b}$

where here x is the the input matrix and **W** and **b** are machine based parameters.On this the activation function is applied the softmax function) to generate outputs in the range of 0 to 1.

Softmax function

$$\hat{y}_i = \frac{\Sigma e^{z_i}}{_j\, e^{z_j}}$$

We define the loss function (convex in nature) as follows

$$L(\mathbf{W},\mathbf{b}) = \sum (-\mathbf{y} log(\hat{\mathbf{y}}))$$

where $\mathbf{y}$ is the actual output intended(the label matrix) and $\hat{\mathbf{y}}$ is the probability obtained. (The activation function utilized generates the outputs in the range (0,1))

$$J(\mathbf{W},\mathbf{b}) = \frac{\sum_{i=0}^{m} (-\mathbf{y} log(\hat{\mathbf{y}}))}{m}$$

where $J(\mathbf{W},\mathbf{b})$ is the cost function.Our goal is to minimize this cost function and find such expressions for $\mathbf{W}$ and $\mathbf{b}$ which satisfy the condition.

```
def categorical_cross_entropy(ytrue, ypred, axis=-1):
    return -1.0*tf.reduce_mean(tf.reduce_sum(ytrue * tf.log(ypred), axis))
```

We use LSTM network to develop and train the model from the inputs and outputs we intend it to give out,optimize the cost function (we use rmsprop here), check it with a fraction of the input data(set aside for validation) and evaluate the loss and the accuracy of the process.