Machine Learning Lab Assignment: Spoken Command Classification Using Artificial Neural Networks

# Objectives

* Feature extraction
* Design the ANN model
* Constructing the cost function
* Writing the gradient updating (back propagation)
* Tuning the hyper-parameters
* Try the model by your own voice

# Preparation

## Preparation 0:

* Study the Coursera course “Machine Learning” week 4 and week 5 to understand the basics of Artificial Neural Networks. (<https://www.coursera.org/learn/machine-learning/home/welcome>).
* Finish the programming exercise 3 and 4. You can reuse the code from there.
* Make MATLAB ready to run.
* Download the dataset and scripts.

## Preparation 1:

* Load the dataset into the MATLAB:
  + E.g. for loading the training data: load(‘Train\_raw\_data.mat’);
  + Do the same thing with validation and testing datasets.
  + Load the label file “Labels.mat”.
* The Speech Commands Dataset[1] consist of plenty of spoken commands, e.g. up, down, yes, no, left, right and so on. Each utterance is 1 second long (sample rate 16kHz), and they are stored in the variables Train\_raw\_data(samples, utterance\_index), Validation\_raw\_data(…, …) and Test\_raw\_data(…, …), you can listen to it by for example:
  + soundsc(Train\_raw\_data(:, 1), 16000);
* The labels are YTrain, YValidation and YTest correspond to the labels for the training set, validation set and testing set separately. The label is an integer number represent a command. Actually, this correspondence is listed below:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| down | go | left | no | off | on | right | stop | up | yes | unknown |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

Each label corresponds to an utterance. For example, utterance Train\_raw\_data(:, 20) has label YTrain(20).

# Feature Extraction

Human speech consists of vowels and consonants Those components are more visible in the frequency domain by using the Fourier Transform. In this command recognition task, one main goal is let the model classify different commands. This could be a subtask of recognition of human spoken languages. If you can first transfer the signal into frequency domain, it will also be easier for the model, the neural network, to learn different properties of different commands.

The features you are going to extract are the auditory spectrogram features. The auditory spectrogram is basically a Short-Time Fourier Transform on the time domain signal. This is followed by a re-scaling of the spectrum into the Mel-scale which have proven to be closer to the human’s perception.

Now we can run the code for feature extraction to get the features. This is done by calling the function speechSpectrograms.m in lab\_nn.m. Note that we also use a log transformation, similar to the dB transformation.

At this point you can consider normalizing the 1 second segments or not.

Inspect the time domain signal and Mel spectrum of an utterance of the same label, say “down”, of different persons in your training set. Can you see some similarities between the same utterances produced by different people?

Here

# Design the ANN model

* Recall the programming exercise 4 in Coursera Machine Learning course week 5, where we have implemented the forward and backward propagation of an ANN with 1 hidden layer (Figure 1).
* 

Figure 1 ANN with 1 hidden layer

You can reuse the network but modify the number of hidden neurons and output neurons. The question is how many output neurons do we need in our spoken command classification task?

* Now implement/reuse the sigmoid.m, sigmoidGradient.m, nnCostFunction.m and predict.m files. You can reuse the code from the Coursera exercise (Please do have submitted the code to Coursera to check if the implementation is correct). However, try to change the for-loops into matrix multiplication to calculate the cost function in nnCostFunction.m, you can observe that using matrix multiplication is much faster than the for-loop implementation.

# Train the ANN model

* The hyperparameters at hand are: size of the hidden layer, lambda and the number of iterations.
* First assume the size of the hidden layer to be equal to 256.
* Also consider normalization vs no normalization of the 1 second segments.
* Please draw the cost vs. iteration curve where the “cost” is a vector returned from the function “fmincg(…)”.
* Use different values for the “lambda” (e.g. 0.0001, 0.001, 0.01, 0.1, 1, 10, 100) and the use different values for the “iteration” (e.g. 400, 800, 1200, 1600, 2000, 2400 ect) . The “lambda” controls the regularization of the leaning parameters, the bigger the “lambda”, the smaller the weights/parameters tend to be. What is the accuracy on the validation data set? What is the accuracy on the training data set for each setting of hyperparameter? What is the best performance for the validation accuracy. How big is the “lambda” and “iteration” you have chosen to get this performance?
* Plot the training accuracy and validation accuracy versus the “lambda” for different “iteration” number as illustrated in figure 2.



Figure 2: Training and the validation set performance for different lambda values.

* Do we have a high variance or a high bias ? How would you improve the performance? Also finally check your accuracy on the test set with all the appropriate hyperparameters selected.
* (Optional) Add an additional hidden layer and check if your performance on the validation data set improves.

Note, to improve calculation power use “parfor” instead of “for” in your implementations. For example to inspect different lambda values. This will allow you to run parallel pools and use all cores on your laptop. Look at the help files for more info. You may need to adjust you code in the for – loop.

# Production of the ANN

You can try to use the trained models on speech you captured yourself by using real\_time\_command\_recognition.m. Note if you trained on normalized 1 second utterances, you also need to normalize the data captured with your laptop microphone. You will also normalize segments where there is only background noise. In that case it would be useful to use a voice activation detection (VAD) algorithm first to select the segments which contain speech before using your trained model.

# Reference

[1] Warden P. "Speech Commands: A public dataset for single-word speech recognition", 2017. Available from http://download.tensorflow.org/data/speech\_commands\_v0.01.tar.gz. Copyright Google 2017. The Speech Commands Dataset is licensed under the Creative Commons Attribution 4.0 license, available here: <https://creativecommons.org/licenses/by/4.0/legalcode>.

[2] Kurt Hornik (1991) "Approximation Capabilities of Multilayer Feedforward Networks", Neural Networks, 4(2), 251–257.