

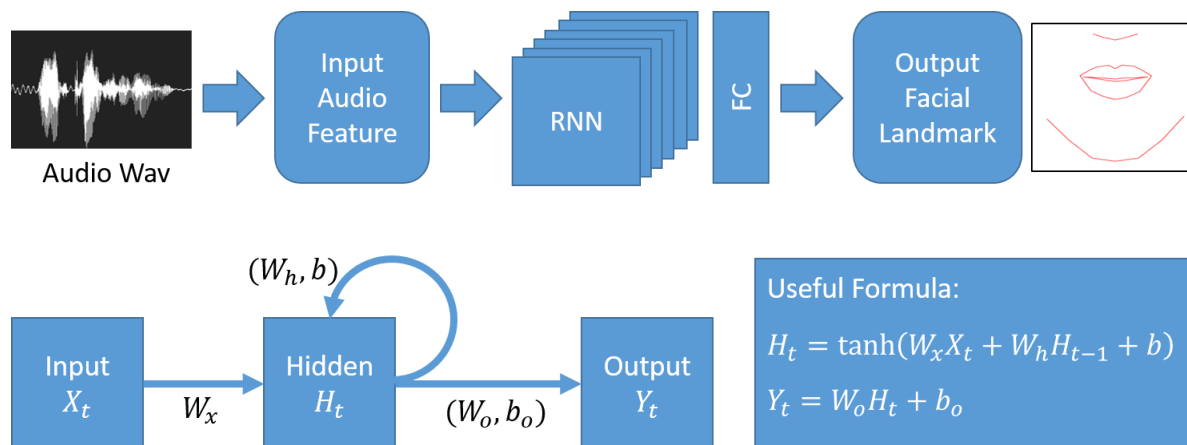
## Intelligent Visual Computing [Spring 2018]

### Assignment 5: RNN for Generating Facial Animations From Audio

#### Overview

In this assignment, you will implement a simple recurrent neural network that produces facial animations given audio features alone. The architecture you will implement is a simplification of the architecture described in the video here: [https://www.dropbox.com/s/pwrwdvlywxucl9/SIGGRAPH\\_18\\_VIDEO.mp4?dl=0](https://www.dropbox.com/s/pwrwdvlywxucl9/SIGGRAPH_18_VIDEO.mp4?dl=0) (SIGGRAPH 2018 submission)

The assignment counts for **12 points** towards your final grade.



#### Instructions

The starter code for this assignment is written in Matlab. Download the code here: [starter\\_code.zip](#). The code has the following scripts and data:

- **starter\_data.mat**: a set of extracted features from the input audio waveform (for more information, see <http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/>) and per-frame facial landmarks representing animated faces talking according to the audio. The dataset has been split into two disjoint sets for training and testing. You will use the data in  $(X_{train}, Y_{train})$  matrices to train your RNN. Once the training is complete, you will use  $(X_{test}, Y_{test})$  to test whether the method can produce good 2D facial animations given audio alone. There are 3 separate test sets stored in  $X_{test\{1\}}$ ,  $X_{test\{2\}}$  and  $X_{test\{3\}}$ . You will report results for each of these test sets.
- **run.m**: script for loading the data and calling the functions below. Check this script to understand how the various training and testing functions are called. You don't need to edit this script.
- **trainRNN.m**: code to train the RNN. **You have to complete this script!**
- **testRNN.m**: code to evaluate the RNN during test time. **You have to complete this script!**
- **rnnForward.m** and **rnnForwardStep.m**: code that must be called by trainRNN.m and testRNN.m to execute forward propagation through time. **You have to complete these scripts!**
- **rnnBackward.m** and **rnnBackwardStep.m**: code that must be called by trainRNN.m to execute backpropagation through time. **You have to complete this script!**

- **CreateAvi.m** and **showFacialLandmarks.m**: output the facial landmark estimation in an AVI video file. You don't need to edit this script.

## What You Need To Do (12 points in total)

**[2 points]** Complete the **rnnForward.m** and **rnnForwardStep.m** scripts to implement forward propagation through time for a "many-to-one" RNN with (a) a single hidden layer and hyperbolic tangent activation functions, and (b) a single fully connected layer implementing a linear transformation of the hidden states to produce output landmarks (check also the comments in this scripts and lecture notes to understand the input/output per layer).

**[5 points]** Complete the **rnnBackward.m** and **rnnBackwardStep.m** scripts to implement backward propagation through time in this network. The script takes as input messages backpropagated from the output layer, and accumulates parameter derivatives. Remember to divide the derivatives by the number of training instances in a batch. In addition, use L2 regularization for the layer parameters (exclude the biases) based on the weight decay argument given in the script (please don't divide the regularization term with the number of training examples).

**[2 points]** Complete the **trainRNN.m** to train the neural network. Use the default options for learning, including rate, momentum, weight decay, number of training epochs listed in the script. Call the forward propagation and backpropagation functions appropriately. Implement gradient descent updates with momentum for the net parameters. Terminate training if the number of gradient descent epochs reaches the maximum number specified in the input options.

**[2 points]** Complete the **testRNN.m** to evaluate the neural network on test audio sequences. The script should call forward propagation and createAvi.m to output a visualization video per test sequence.

**[1 points]** In the written part of your assignment, report the mean square error for each of the 3 test sequences (use the test\_RNN.m script). Include in your submission the video generated for the **FIRST test sequence ONLY (please do not upload all three)**.



Optionally, you can also add the test audio to the generated videos (not required to do this for your submission). You can use audio/video processing software that you are familiar with to do this. For example, you may install and use ffmpeg [https://www.ffmpeg.org/download.html] by specifying in the command line:  
 ffmpeg -i pred\_XXX.avi -i test\_XXX.wav -codec copy pred\_with\_audio\_XXX.avi  
 (change 'XXX' to the test sequence index.)

## Bonus (+3 points in total)

Implement a LSTM using your favorite deep learning package for this task. Feel free to adjust its structure and hyper-parameters. Include mean square error for all three test sequences and video for the first test sequence. Please submit the results for BOTH the above RNN to get the basic 12 points, and the LSTM to get the bonus 3 points in SEPARATE folders inside your zip archive.

## Submission

Please follow the **Submission** instructions in the [course policy](#) to upload your zip file to Moodle. The zip file should contain all the Matlab code plus a **PDF report including your answers for the written part**. Make sure your program runs in Matlab.

-  [starter\\_code.zip](#) 

## Submission status

Submission status	No attempt
Grading status	Not graded
Due date	Friday, April 6, 2018, 11:55 PM

---

Time remaining	5 days 2 hours
----------------	----------------

---

Last modified	-
---------------	---

---

Submission comments	► <a href="#">Comments (0)</a>
---------------------	--------------------------------

Add submission

Make changes to your submission