Homework 3 Part 2
Utterance to Phoneme Mapping

11-785: Introduction to Deep Learning (Spring 2019)

OUT: March 11th, 2019
DUE: **March 31st, 2019, 11:59 PM**

1 Introduction

In this homework you will again be working with speech data. We are going to be using unaligned labels in this contest, which means the correlation between the features and labels is not given explicitly and your model will have to figure this out by itself. Hence your data will have a list of phonemes for each utterance, but not which frames correspond to which phonemes.

Your main task for this assignment will be to predict the phonemes contained in utterances in the test set. You are not given aligned phonemes in the training data, and you are not asked to produce alignment for the test data.

2 Dataset

Similar to HW1P2, you will be provided with mel-spectrograms that have 40 band frequencies for each time step of the speech data. However in this assignment, the labels will not have a direct mapping to each time_step of your feature, instead they are simply the list of phonemes in the utterance [0-45]. There are 46 phoneme labels. The phoneme array will be as long as however many phonemes are in the utterance. We provide a look-up, mapping each phoneme to a single character for the purposes of this competition.

The feature data is an array of utterances, whose dimensions are `(frames, time_step, 40)`, and the labels will be of the dimension `(frames, frequencies)`. The second dimension, `viz.`, frequencies will have variable length which has no correlation to the time_step dimension in feature data.

You can download the data files from the Kaggle competition link here or as a tar file here.

2.1 File Structure

In total, we have 7 files for this assignment, including 1 '.csv' file, 1 '.py' file and 5 '.npy' files. Their functions and structure of organization are explained below.

- `wsj0_train.npy`: This file contains your feature data for training the model. It will be of the shape `(frames, time_step, 40)`, where the second dimension will be variable as in HW1P2.
- `wsj0_dev.npy`: This file is similar to `wsj0_train.npy`, but should be used to calculate your validation losses and accuracy.
- `wsj0_test.npy`: This file is similar to `wsj0_train.npy`, but should be used to predict the phoneme labels for the final Kaggle submission.
- `wsj0_train_merged_labels.npy`: This file contains the labels or phoneme list for each utterance of the `wsj0_train.npy` file. The dimensions of the data in this file will be of the form `(frames, frequencies)` where the second dimension is variable.

- `wsj0_dev_merged_labels.npy`: This file is similar to the one above, but instead will map the labels to the `wsj0_dev.npy` file. You can use this for predicting validation losses and accuracy.
- `sample_submission.csv`: This is an empty submission file that contains the headers in the first row, followed by the test utterance Id and predictions for each utterance of test data.
- `phoneme_list.py`: This file contains the phoneme list and the mapping of each phoneme in the list to their respective sounds. Your submission file should contain these sounds as output and not the phoneme or their corresponding integer.

3 Getting Started

3.1 CTC Loss

As described above, there is no alignment between utterances and their corresponding phonemes. Thus, train your network using CTC loss. Decode your predictions, preferably using beam search. Use the list of phonemes provided on the data page to make each prediction into a text string.

Tensor flow has built-in CTC loss function. But for Pytorch, you can install the CTC-Loss library. The required CTC-loss library, Warp-CTC, can be installed here. Follow the download instruction. NOTE: Make sure you download the GPU supported version.

3.2 Using Warp CTC in pytorch

The module assumes the blank position is at 0. If there are n labels, the model output should be of size n+1. The targets passed to the loss function should be from 1 to n instead of from 0 to n-1.

The loss function takes four inputs. The first input is a tensor of your model network output. The second input is a tensor of all the labels should be concatenated into one single 1D tensor. The third input is a tensor containing the size of each output sequence from the network. The fourth input is a tensor containing the label of each target.

Your submission should contain the predicted strings for each item in the test set.

3.3 CTC Decoding

If you are using PyTorch you can manually install the library, ctcdecode, here. They have an implementation of beam search, use it in your code to decode the output of your model. If you are using Tensorflow, it has it's own implementation of beam search.

4 Evaluation & Submission

You will be evaluated using Kaggle's character-level string edit distance. Since we mapped each phoneme to a single character, that means you are being evaluated on phoneme edit-distance.

We are using Levenshtein distance, which counts how many additions, deletions and modifications are required to make one sequence into another.

Your submission should be a CSV file. The headers should be "Id", and "Predicted" - `Id` refers to the 0-based index of utterance in the test set and `Predicted` is the phoneme string. Please note that the headers are case-sensitive.

See sample submission for details.

5 Conclusion

That's all. As always, feel free to ask on Piazza if you have any questions.

Good luck and enjoy the challenge!