Machine Learning, Speech and Language Technology

MLSALT10: Statistical Speech Synthesis

Practical: Parametric Speech Synthesis

1 Introduction

The aim of this practical to investigate a simple HMM-based parametric synthesis system, how the trajectories are generated and used for synthesis. A baseline synthesis system is supplied.

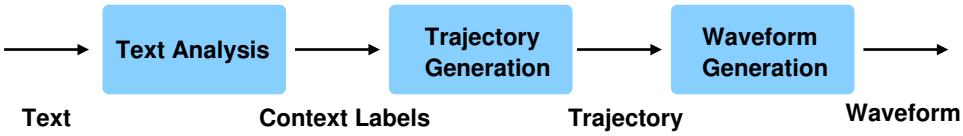

Figure 1: Lattice Based Keyword Spotting System

The synthesis process is split into three distinct stages for this practical, illustrate in in Figure 1. Scripts are supplied for each of these stages, as well as the form of the output. The theory for this practical is described in the lectures, or the reference to the appropriate paper is given.

As part of this practical you will need to use `matlab` (or write your own versions in, for example, `python`), listen to waveforms, and generate spectrograms. The tools that you use for this are up to you.

2 Synthesis System

This practical uses a basic synthesis system built using HTS. The feature vector used to build this system has a 246 dimensional feature vector with a 5 millisecond frame rate (25 millisecond widow size). This feature vector will be split into 5 *streams*:

1. `mel-cepstrum`: 180-dimensional comprising: 60-dimensional `STRAIGHT` derived mel-cepstrum parameter $(c_0, \ldots, c_{59})$, delta and delta-parameters;
2. `log-F0` (1-dimensional): `REAPER` derived log-transformed F0;
3. `delta log-F0` (1-dimensional): delta parameters for log-F0;
4. `delta-delta log-F0` (1-dimensional): delta parameters for log-F0;
5. `aperiodicity`: 63-dimensional comprising: 21-dimensional `STRAIGHT` derived aperiodicity parameter, delta and delta-parameters;

Each of these streams has a separate decision tree. The order of the feature vector is as above. Thus element 127 is the delta-delta values for the 7th mel-cepstrum feature (c_6).

The delta and delta-parameter features, are derived from a five-frame window. The weights associated with these five frames are

$$\begin{array}{rrrrr} 0.000000 & 0.000000 & 1.000000 & 0.000000 & 0.000000 \\ -0.200000 & -0.100000 & 0.000000 & 0.100000 & 0.200000 \\ 0.285714 & -0.142857 & -0.285714 & -0.142857 & 0.285714 \end{array}$$

for the static, delta and delta-delta parameters respectively. For this system each context-dependent model is a single Gaussian component, per stream, with a diagonal covariance matrix.

Note though log-F0 is modelled in the system, during synthesis the trajectory is converted from log-F0 to F0. In `HTK` the `log(0)` is set to a floor value (`LZERO`) to avoid numerical issues.

3 Practical

This practical is based on a TTS systems built from the data supplied for the *Hurricane* speech synthesis challenge. The data is from a single speaker, *Nick*. The text-analysis *frontend* processing is performed using `Festival`. The feature extraction is based on `REAPER` for F0 analysis and `STRAIGHT` for aperiodicity and mel-cepstrum features.

3.1 Infrastructure

The following files are supplied for this practical. They may be found in the directory `/usr/local/teach/MLSALT10/Practical` on the system.

(a) `scripts` - for the three steps to synthesising a waveform

- `txt2lab.sh`: performs text analysis, converts the word text into a sequence of context labels. In addition default duration from `Festival` are generated;
- `lab2traj.sh`: convert the labels in trajectories for the *aperiodicity*, `F0` and `mel-cepstrum` parameters;
- `traj2wav.sh`: convert the trajectories into a waveform file in *wav* format).

(b) `models` - contains the model sets that can be used for synthesis. This model was trained in the same fashion using a subset of the *Hurricane* data, approximately 2 hours of data,

- `htk`: HMM-based synthesis model built using a toolkit distributed by Edinburgh University, and the `HTS` extension to `HTK`
- `hts`: HMM model converted to correct format for `hts_engine`

(c) `matlab` - contains the matlab scripts:

- `load_htkdata.m` reads the HTK format *cmp* files
- `load_traj.m` reads the generated mel-cepstrum trajectory

(d) `original` - contains the following directories

- `txt`: the word text associated with the utterance
- `wav`: the original waveform for the utterance
- `cmp`: the parametrised waveform for the utterance
- `lab`: the label file generated using force-alignment to obtain context labels, and phone start and end times

(e) `txt` - an example of a text input to the system is given in `example/bass.txt`

To get started you will need to link some of these directories, and copy the `scripts` and `matlab` directories into the directory where you are going to run the practical:

```
ln -s /usr/local/teach/MLSALT10/Practical/bin
ln -s /usr/local/teach/MLSALT10/Practical/models
ln -s /usr/local/teach/MLSALT10/Practical/original
cp -r /usr/local/teach/MLSALT10/Practical/scripts .
cp -r /usr/local/teach/MLSALT10/Practical/matlab .
```

The `txt` directory gives you the text from one of samples played in the lectures.

3.2 Experiments: Synthesis and Trajectories

The aim of this part of the practical is to investigate how an HTS (HMM-based) parametric synthesis system performs and the nature of the trajectories that are generated.

1. Initially a single utterance will be examined `utt1`. The waveform and text for this file are in

   ```
   original/wav/utt1.wav
   original/txt/utt1.txt
   ```

   Check that you can play this waveform, and examine the spectrogram for this waveform [1].

2. Generate the labels for the reference sentence using the supplied text analysis tool (`Festival`)

   ```
   ./scripts/txt2lab.sh original/txt/utt1.txt lab
   ```

   This generates a label file for the utterance in

   ```
   lab/utt1.lab
   ```

[1] You are welcome to use any tool you like to generate spectrograms, and play waveforms. `audacity` is the tool that was used when designing this practical.

You can see from this file that the context models used for synthesis are significantly more complicated than those in speech recognition.

3. Generate the trajectory of the model parameters using the label file above:

```
./scripts/lab2traj.sh  -hmmdir models/hts -labdir lab -outdir traj -filename utt1
```

This command generates a number of files. The ones of interest are

- `traj/utt1.mcep`: the (60 dimensional - $c_0, c_1, \ldots, c_{59}$) mel-cepstrum parameter trajectories
- `traj/utt1.apf`: the (21 dimensional) aperiodicity parameter trajectories
- `traj/utt1.f0.txt`: the (1 dimensional) f0 parameter trajectories

The *mcep* and *apf* files are stored in 32-bit float format. You can use the `load_traj.m` command to read the trajectories. Note *size* in this command refers to the number of trajectory (for exmaple 60 for the mel-ceptrum).

The file *utt1.f0.txt* is stored in text format. Examine the nature of the trajectories. Are they what you expect?

4. Generate the waveform from the trajectories

```
./scripts/traj2wav.sh  -trajdir traj -outdir wav -filename utt1
```

This generates the waveform in

`wav/utt1.wav`

You should play this waveform. How does it compare to the original waveform? What about the spectrogram?

5. It is also possible to manipulate the trajectories. For example replace

`traj/utt1.f0.txt`

by a file of the same length, but just containing zeros. Use this modified trajectory to generate a waveform. Discuss the quality of the synthesised waveform, against the previously generated one and the attributes of the signal.

6. It is possible to contrast the generated trajectories with the original waveforms. To load the original parametrised waveform features in

`original/cmp/utt1.cmp`

into matlab use the `load_htkdata.m` command. The lengths of the number of samples in the generated trajectory differs from the original, making it awkward to generate distance between the synthesis trajectories. *As part of the write-up you should discuss how this could be solved.* It is possible to generate trajectories based on forced alignment of the waveform data with the acoustic model. The resulting label file with start and end-times is in:

`original/lab/utt1.lab`

This can then be used to generate trajectories that have (about) the same length as the original waveform using

```
./scripts/lab2traj.sh -labdur -hmmdir models/hts -labdir original/lab \\
    -outdir traj-dur -filename utt1
```

Due to *end-effects* the lengths may be slightly different. By considering the distances between the generated feature trajectory and the original features, synchronise the two trajectories and comment on the general forms of the two trajectories [2].

7. It is possible to replace elements of the synthesised trajectories with the original waveform parameters, and then generate waveforms. You will need to either pad or strip the trajectories to make them the same length. What is the impact on the waveform of replacing particular trajectories, or all the trajectories? Discuss whether this is what you expect.

3.3 Experiments: Trajectory Generation

This part of the practical examines the generation of the trajectories of the model parameters and how *global variance* alters the form of the trajectory.

1. Write code that generates a trajectory given the *experts* for a particular dimension of the mel-cepstral features. For this model the experts were based on static, delta and delta-delta parameters. See the synthesis description section 2. Check that the window weights make sense to you (consider the delta and delta-delta feature extraction from the speech recognition module). There is a script supplied that will extract the parameters of the expert for a particular dimension of the system.

   For this section only the mel-cepstrum trajectories will be considered, *stream* 1. To extract the expert parameters for dimension 4 (c_3) of stream 1 use

   ```
   ./scripts/getexpert.sh -hmmdir models/htk -labdir original/lab -stream 1 -dimension 4 \\
       -outdir expts  -filename utt1
   ```

   This script generates two files containing the experts:

   - `expts/utt1.cmp.expt`: these are the experts for mel-cepstrum 4 for each of the models and states. The format is:

     ```
     static-mean delta-mean delta-delta-mean static-var delta-var delta-delta-var
     ```

   - `expts/utt1.dur.expt`: these are the experts for the (emitting) state durations

[2] The synchronisation is just a shift (in trajectory start) in this case - the "end-effects" means that some silence frames may not be generated. To get an idea of what is going on, pick one dimension of the mcep and compare the two trajectories in matlab.

```
state-means (2-6) state-variances (2-6)
```

The acoustic models are five emitting state HMMs. Check that you understand the form of the experts provided. You will need to ensure that the number the frames for state experts add up to the phone durations in the label file.

After you have generated the trajectory compare the parameters with those from the trajectory generation process supplied, and the trajectories derived from the audio. You need to think about the choice of start/end smoothing, *end-effects*, when comparing the trajectories, and the best approach to measure the distances between the trajectories.

How efficient is this trajectory generation process? Is it possible to make it more efficient? [3].

2. In addition to the state experts it is possible to have a global variance (GV) model (see lecture notes). The application of this model can be either as an:

   (a) **expert** within a product of experts framework;

   (b) **constraint** within the optimisation approach - see

   `http://mi.eng.cam.ac.uk/~sms46/papers/shannon2013fast.pdf`

   Implement and contrast both approaches.

   The parameters of the experts for the mel-cepstrum parameters are in the file

   ```
   models/hts/gv-mcep.pdf
   ```

   This file contains the 60 means, follows by the 60 variances of the mel-cepstrum GV experts. To read these you can use commands like

   ```
   ./bin/x2x +da   models/hts/gv-mcep.pdf
   ```

   The parameters of the GV model, the mean and variance, for the fourth mel-cepstrum (c_3) are

   ```
   0.169537
   0.00293601
   ```

   Compare the performance of both approaches in terms of the distance to the actual speech trajectory for this sentence, and the original generated trajectory. What about the distance of the global variance of the generated trajectory compared to the expert, and the original parametrised waveform?

   Do you expect global variance to influence all cepstral parameters equally? Examine other dimensions of the system and comment on what you see.

3. In addition to the data for `utt1`, there are eight other utterances that can be examined (`utt2` to `utt9`). Do you see the same trends for all utterances?

[3] This will, of course, depends on the efficiency of you original implementation!

4 Write-Up

Your report for this practical should include details of the code that you have written as well as the experiments that you have run. Ensure that you do as much analysis as possible when discussing you results.

In total your report should be around 2,000 to 3,000 words long. You should include the location of the code that you have written for trajectory generation and global variance application (and make sure that they are readable!).