Draft Dec 15 2014

Speech synthesis assignment for S-89.5150

Task

Parametrise speech waveforms for one speaker. Build a triphone (or more complex if desired) HMM-model for vocoder parameters and duration following the HTK tutorial (with some tweaks). Synthesise vocoder parameters for a previously unseen sentence. Generate the speech waveform for that sentence.

Goals

1. Pass the course
2. Gain understanding on the HMM as a generative model
3. Apply and practise problem solving skills with speech data in various forms

Expected reporting

1. The usual stuff as required by the professor
2. Include samples of the synthetic speech
3. Some feedback on the time spent on assignment!

Reading

1. Tokuda, Keiichi, et al. "An algorithm for speech parameter generation from continuous mixture HMMs with dynamic features." (1995).
2. Tokuda, Keiichi, et al. “Speech Synthesis Based on Hidden Markov Models.” Proceedings of the IEEE | Vol. 101, No. 5, May 2013
3. HTKBook
4. HTS document
5. SPTK reference

Data

American English speakers slt (female) or bdl (male). Choose one. For example by flipping a coin. Wav files are provided in **/work/courses/T/S/89/5150/general/synthesis/wav**. The data has been divided to train and test sets. Mono-, tri- and quinphone as well as full context labels are provided for both sets in **labels** directory.

Required software

The following software will be provided in the course directory **/work/courses/T/S/89/5150/general/synthesis**:

- HTK patched with HTS

- SPTK (speech processing toolkit)
- Hhead (simplifies feature packaging)

Suggested workflow

1. Get comfortable with vocoding
    - Use SPTK tools to extract vocoder parameters. Use the extracted parameters to regenerate the speech waveform.
    - Compare different vocoders: MCEP, LPC, LSF. Play around a little with dimensionality. Get comfortable with frame-based representation of feature vectors. Select a vocoder that you feel you understand at least a little.
    - Tips:
        - HMM-based speech synthesis at 16kHz typically uses 25ms frames with 5ms frame step.
        - SPTK requires wav files to be converted to raw files first (wav2raw, sox). The encoding must be 4-bit float (+f).
        - SPTK: “order” vs “length”
        - x2x command is useful for converting data from one numeric encoding to another, or to get an ascii representation (for debugging…). Note that you can order the ascii output to a matrix with N columns with x2x +faN, to make frame-by-frame investigation easier
        - Feature extraction mostly starts with framing and windowing (these should be familiar from the course?)
        - Try plotting some frames: The vocoder parameters and their spectral envelopes. Matlab (with signal processing toolkit?) has tools to create spectral envelopes for the different parametric representations. An understanding of the vocoder (parameter ranges and associated spectral shape of vowel frames etc) will be very useful when debugging the synthetic versions!

2. Feature generation
    - Once you’ve selected a feature type, build the features for all the waveforms.
    - Delta features should be familiar from the course

- You need to combine the spectral components and pitch components for each frame. A sample frame (from [1]):

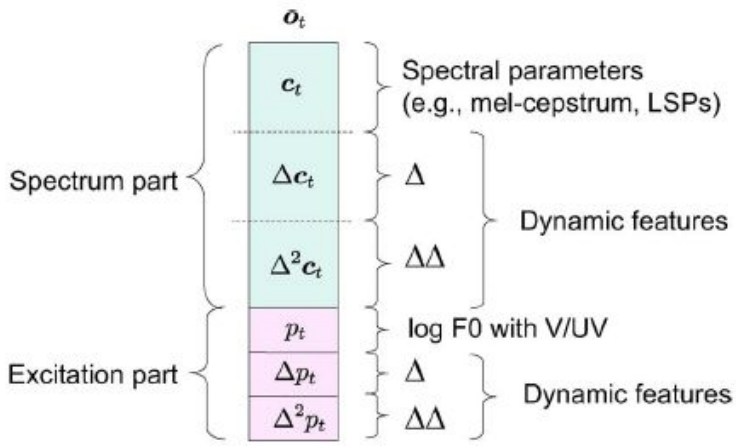

Fig. 3. ***Example of an observation vector at each frame.***

- Create the header for each file. From [3]:

> The HTK file format header is 12 bytes long and contains the following data
> 1. nSamples number of samples in file (4-byte integer)
> 2. sampPeriod sample period in 100ns units (4-byte integer)
> 3. sampSize number of bytes per sample (2-byte integer)
> 4. parmKind a code indicating the sample kind (2-byte integer)

- The numbers can be computed and added to the beginning of the file with SPTK tools, but HHead is provided to simplify the operation; Parameter type is “9” (for “other”)
- Terminology:
  - “Frame” :
  - “Samples” :
  - “Window” :
- Tips:
  - SPTK tools delta produces the delta features
  - SPTK bcp (block copy) or unix command “paste” can be used to combine the feature streams
  - The delta window needs to match those used in the synthesis. Simple windows can be generated like this:
    ```
    echo "-0.5 0 0.5" | x2x +af > delta.win.float
    echo "1.0 -2.0 1.0" | x2x +af > accel.win.float
    ```
  - Check some files for the delta features. If the (absolute) values appear “unreasonable” ie. outside [10^{-10}, 10^{10}], then there is probably an error in your feature extraction script and it saves a lot of time to fix it now!
  - Pitch/F0 is usually converted to log domain where it can be better modelled. Relatively easily done with Awk, perl or python.

        - Pitch/F0 is either 0 (unvoices) or a positive float (voiced); Use “Magic number” definition in delta command for pitch to avoid computing delta features for unvoiced frames.
        - HTK/HTS internal logic uses ridiculously large negative numbers as “magic number” for f0 of unvoiced frames. So replace the zero-valued f0 and its delta and delta-delta values with -1e+10 before packaging in the features. Again, easily done with Awk, perl or python.

3. Model Training
    - Follow the tutorial in HTK book. Monophone, triphone, quinphone and full context labels will be provided, as well as the associated list of phonetic questions for clustering. Go as far as you like in model detail!
    - Tips:
        - Write the training procedure into a script or little program wrapper. This way you can run it easily again if you want to change the vocoder parametrisation.
        - Start by building a monophone model and try to synthesise with it. It won’t sound good, but should resemble speech somewhat. When it works, do the triphone expansion and synthesise with the expanded model set. If you feel brave, you can go to even more detailed context of quinphones or full-context, but this is not required.
        - You probably want to increase the pruning beam with **`-t`**. Find a balance between slowness and amount of rejected data that suits you.
        - You can skip steps 7 and 8 (why?)
        - Pipe the output of the tools to log files, you might have to go back to find errors. A handy tool in any unix machine is “**`tee`**” - It displays input and writes it to a file at the same time.

    Monophone training:
    - The network won’t be needed and the labels are provided already. So start with step 6 in Section 3.2.
    - 3.2.1: A 5-state proto HMM that uses separate streams for pitch/F0 modelling has been provided in course directory. This needs to be modified to fit your speech parametrisation.
      A config file for reading features generated by SPTK is also provided
    - The synthesis models are usually so-called “hidden semi-Markov models” (HSMM) as the duration of the phones is often modelled explicitly by Gaussians. For that, a separate duration model set needs to be created. You can initialise the duration model mmf by adding **`-g hmm1/hmmdefs.dur`** to the first HERest command and the file of that name will be created.

- Add **`-N $sourcedir/hmmdefs.dur -u mvwtdmv`** to following HERest commands to keep using and updating the duration model, and **`-R $targetdir`** to save the updated duration model in the right place
- You can either parse the master label files (**`-I mono.mlf`**) or specify the training tools to use labels straight from the directory (**`-L /work/courses/T/S/89/5150/general/synthesis/labels/train/monophone/`**)

Triphone/Quinphone/Full context training

- The full list of triphone/quinphones/full-context labels can be parsed with your favourite scripting language (Python/Perl?) or simply with command line tools, like this (for full-context labels, for triphone or quinphone use the corresponding directories and file names:): **`awk '{print $3}' /work/courses/T/S/89/5150/general/synthesis/labels/*/full/*lab | sort -u > full1`**
- The HHEd command file for model cloning with "**`CL triphones1/quinphones1/full1`**" does not require a list of transition state tyings - Transitions have already been removed when the duration model was generated.
- HHEd needs "**`-p`**" switch to work with more complex labels
- HHEd needs to be run separately for acoustic model and duration model in triphone/quinphone/full context expansion and in clustering
- Check HERest "**`-m`**" option - You might need to use it at some point!

Making tied-state triphones/Quinphones/Full-context models

Some funky additions by our Japanese friends here, just very very badly documented:

- You need to cluster the vocoder parameter HMM and duration HMM separately. Just run HHEd with different input and output files.
- Minimum Description Length (MDL) criterion is used to control the clustering. Nothing to worry about, it just means leaving out the thresholds from RO and TB commands in the edit file, and including "**`-i -p -s -m -a 1.0`**" (or some other number, if you want to try tweaking the clustering thresholds) to the HHEd command.
  You should get an indication of MDL in the ouput of the HHEd command, something like:

```
(...)
RO 0.00 ''
 Setting outlier threshold for clustering
 RO->LS stats
  and loading state occupation stats
  Stats loaded for 37922 models
  Mean Occupation Count = 3.443303
```

```
TB 0.00 lsf_s2_ {*.state[2].stream[1]}
 Tree based clustering based on MDL criterion,
threshold=5.532142e+02
   setting question*model table...done
(...)
```

- Duration model requires its own statistics file; But it does not require a real stat file, a fake one that lists only 1 frame per model. Can be generated for example with awk like this: **`awk '{print NR "\t" $1 "\t1\t1"}' triphones1 > stats.dur`**
- The tree.hed files are similar to those of HTKbook page 41, but because of MDL clustering, use threshold of 0 for RO and TB commands.
  Also, for more efficient clustering, generate single trees per state with "**`-s`**" switch to HHEd. That also means that clustering commands are shorter, as all models in the same state are tied together. For example, if you name your vocoder parameters "mcep" and "lf0", you'd have:

```
TB 0 mcep_s2_ {*.state[2].stream[1]}
TB 0 mcep_s3_ {*.state[3].stream[1]}
…
TB 0 mcep_s6_ {*.state[6].stream[1]}
TB 0 lf0_s2_ {*.state[2].stream[2-4]}
TB 0 lf0_s3_ {*.state[3].stream[2-4]}
…
TB 0 lf0_s6_ {*.state[6].stream[2-4]}
```

  in your tree.hed file, ie. you have to do the tying separately for your prime vocoder stream (number 1) and the pitch/F0 stream (MDS streams 2-4). Respectively, for duration model, each model has 5 streams to represent a duration model for states of the acoustic model hmm, so the tying commands require:

```
TB 0 dur_s2_ {*.state[2].stream[1]}
TB 0 dur_s3_ {*.state[2].stream[2]}
…
```

  in your duration HHEd edit file.
- As there are far less TB commands for the single tree clustering setup, I recommend to construct this by hand. The structure is:

`RO 0 stats`	Load stats, don't remove outliers

`QS "L-Vowel" {e-*,…`
`…`
`QS "R-pau"  {*+pau}`
	Copy the contents of question file here

`TB 0 mcep_s2_ {*.state[2].stream[1]}`	Add the clustering commands for each state and stream

… **TB 0 lf0_s6_ {*.state[6].stream[2-4] }**	
AU triphone1 **CO tiedlist.triphone** **ST trees.triphone**	Guess models for unseen triphones; Get rid of identical models; Save trees for future use

- This needs to be constructed also for the duration model: That will look very similar but there will be less clustering commands and the file names should be different to avoid overwriting.
- To bring all that together, you should have two HHEd commands that should like something like this:
  HHEd -T 1 -H hmm12/macros -H hmm12/hmmdefs -a 1.0 -i -p -s -M hmm13 tree.hed triphones1 | tee log/hmm13
  HHEd -T 1 -H hmm12/hmmdefs.dur -a 1.0 -i -p -s -M hmm13 tree.dur.hed triphones1 | tee log/hmm13.dur

- For quinphones and full context labels follow the same procedure but take the labels from a different directory. Don't overwrite files!

4. Parameter generation:
   - Use HMGenS tool to synthesise vocoder parameters for some labels. Use SPTK tools to generate the speech waveform.
   - The command for generating the vocoder parameters is (for monophone system) (using a single test label):
     HMGenS -T 1 -C config -C generation.conf -H hmm5/macros -H hmm5/hmmdefs -N hmm5/hmmdefs.dur -M . monophones0 monophones0 /work/courses/T/S/89/5150/general/synthesis/labels/test/monophone/arctic_b0500.lab
   - And for triphone or more complex system:
     HMGenS -T 1 -C config -C generation.conf -H hmm15/macros -H hmm15/hmmdefs -N hmm15/hmmdefs.dur -M . tiedlist tiedlist.dur /work/courses/T/S/89/5150/general/synthesis/labels/test/monophone/arctic_b0501.lab
   - A sample configuration file for synthesis s given in **/work/courses/T/S/89/5150/general/synthesis/misc_files/generation.conf**. You need to edit the order and feature names ie. replace ${featurename} with lsf, mcep or lpc, whatever you are using, and the ${order+1} with the number of parameters for the vocoding (ie Length, which in SPTK is often order+1).
   - Tips:

- There are a gazillion parameters to tweak. Don't be scared if the first output sounds a little funny, but try to see if the problems are in HMM synthesis or vocoder synthesis. Here is helps if you selected a vocoder whose parameters you understand and can plot some generated frames.
- Check the output format of HMGenS, make sure to convert to 4-bit float if necessary
- As you should be using a log-scale pitch/F0, remember to exponentiate it before using it as excitation for vocoder synthesis with SPTK tools!

Help!

You are the guinea pigs for this assignment so you might face some challenges that I have not noticed. I do not have fixed office hours, so it is best to email or phone me to arrange meetings in advance. Feel free to email me about problems, I can probably provide quick answers to many of the common problems.

Happy working,

Reima
reima.karhila@aalto.fi
+358 50 430 3384