# DT2118 Lab3: Continuous Speech Recognition

Thomai Stathopoulou

May 18, 2015

# 1 Training and Testing

## 1.1 Prepare the database

After running the data preparation script, we get different files containing information about the database. That information can be summed up as follows:

- training set:
  - 6160 male speakers (4235 men & 1925 boys)
  - 6389 female speakers (4388 women & 2001 girls)
- testing set:
  - 6236 male speakers (4311 men & 1925 boys)
  - 6311 female speakers (4389 women & 1922 girls)
- 8623 training utterances
- 8700 testing utterances
- 22 phonemes (ah, ao, ay, eh, ey, f, hh, ih, iy, k, n, ow, r, s, sil, sp, t, th, uw, v, w, z)
- 16 nodes and 36 arcs in the recognition network

## 1.2 Train and test G-HMMs with different features

Looking at the configuration files provided for the features and the HMM models we can see that the MFCC features (and their variations) are all created with the same parameters that we used in Lab 1. So their configuration files are identical, except for the field `TARGETKIND`, which denotes the different feature kinds that we want to produce (MFCC, MFCC plus zeroth coefficient, etc.).

The configuration files for the HMM models, describe models of the same basic structure, i.e. all models have five states, of which only three are emitting. Additionally the transition probabilities of the states for every model are the same and the probability distributions for the different observations of every model are also identical. The characteristic that changes is the number of observations. This is a direct result of the feature kind, which every HMM model uses. The number of the features that every feature kind produces, is the number of different observations that every HMM model is expected to observe, hence the different features kinds have the following numbers of features:

`MFCC` 12 features

`MFCC_0` 13 features

`MFCC_0_D` 26 features

`MFCC_0_D_A` 39 features

`MFCC_0_D_A_Z` 39 features

As already mentioned, the basic structure of the HMM models is the same for all the models. So, the number of their parameters is: $5states + 5 \times 5transitionProbabilities = 30freeParameters$. This number, however is calculated without accounting for the different constraints (eg. some of the states are not emitting and some transition probabilities are zero), which would result in a decrease of the parameters.

At the training of the HMM model, the prototype models defined in the configuration files are used. In this step we are more interested in the topology of the models and not the actual parameters. We want to create a three state HMM for every phone, so in `hmm0` we merely place the prototype models for every phone (including "sil" for silence). For this, a set of the training data has been scanned and the means and variances have been replaced by the global speech means and variances.

Having created the prototype model, it is trained with the training data, in order to acquire the correct means and variances. This training procedure is repeated recursively, in order for the models to be re-trained and their parameters re-estimated. This results in models `hmm1` - `hmm3` (`hmm3` contains the final set of initialised monophone HMMs).

In order to make the models more robust, we want to include to "short pause" ("sp") in the training. So we create an extra model just for the short pause with only one state. This model is then connected to the "sil" model. The prototype for this set of models is included in `hmm4`, which then is processed in order for new transition probabilities to be added for the new states (`hmm5`).

After this new addition, the new models are re-trained two more times, giving us `hmm6` and `hmm7`.

The models now are ready to be tested on the testing set. We can observe that for the different feature kinds we get different results. The results are summarized in Table 1.

Table 1: Word accuracy and correct words for different feature kinds

| Feature Kind | Correct Words / All Words (%) | Accuracy (Insertions) |
|---|---|---|
| MFCC | 26392 / 28583 (92.33%) | 91.2 (324) |
| MFCC_0 | 26733 / 28583 (93.53%) | 92.71 (235) |
| MFCC_0_D | **27923 / 28583 (97.69%)** | **96.34 (385)** |
| MFCC_0_D_A | 27789 / 28583 (97.22%) | 93.69 (1010) |
| MFCC_0_D_A_Z | 27706 / 28583 (96.93%) | 92.44 (1285) |

Table 2: Word accuracy and correct words for different numbers of Gaussian components per state, using feature MFCC_0_D_A

| No. of Gaussians | Correct Words / All Words (%) | Accuracy (Insertions) |
|---|---|---|
| 2 | 28141 / 28583 (98.45%) | 96.37 (595) |
| 4 | 28304 / 28583 (99.02%) | 97.68 (384) |
| 8 | 28411 / 28583 (99.4%) | 98.31 (311) |
| 16 | 28465 / 28583 (99.59%) | 98.66 (265) |

It is obvious from this Table, that MFCC_0_D gives the best results in terms of correct words and word accuracy (97.69% and 96.34% respectively). MFCC_0_D_A and MFCC_0_D_A_Z are quite close in terms of correct words, but have lower accuracy, since they have almost three to four times the insertion errors. MFCC and MFCC_0 have the least correct words, but their accuracies are the ones that are closest to the correct word percentage, since they have fewer insertion errors.

## 1.3 Train and test GMM-HMMs with increasing number of Gaussians

We now train GMM-HMM models with different numbers of Gaussian components per state. The results can be seen in Table 2. It is very obvious that the more Gaussian components are used, the better the performance of the model. Both accuracy and correct words percentage increase.

## 1.4 Tune insertion penalty parameter

Using MFCC_0_D , which had the best performance in Section 1.2, we can see that the numbers of insertions and deletions are unbalanced (385 insertions over 190 deletions - almost twice as many insertions as deletions). After running this experiment again using different Insertion Penalty parameters, we get the results shown in Table 3. First we can observe that, the lower the

Table 3: Influence of the Insertion Penalty Parameter on the performance of the models

| Penalty Parameter | Insertions | Deletions | Correct Words | Accuracy |
|---|---|---|---|---|
| 10.0 | 602 | 154 | 97.79% | 95.69% |
| 20.0 | 385 | 190 | 97.69% | 96.34% |
| 25.0 | 290 | 228 | 97.51% | 96.5% |
| **30.0** | **241** | **257** | **97.38%** | **96.54%** |
| 40.0 | 156 | 320 | 97.15% | 96.61% |

penalty parameter, the higher the number of insertions occurring (as would be expected). So as we increase the penalty parameter, the insertions decrease and the deletions increase. Additionally we observe an slight increase in the system's accuracy and a similar decrease int the number of correct words. The closest to a balance we achieve with a penalty parameter 30.0.

### 1.5 Forced Alignment

After using Wavesurfer to check the alignment between the phonemes and the actual sounds, we can conclude that the alignment is quite good. There are, of course some misalignments, that are observed by the naked eye, but in general it seems to be a good correspondence between sounds and phonemes.

## 2 Analysis

### 2.1 Parameter evolution at different iterations

Figures 1 and 2 show the evolution of the means and variances along the iterations respectively. The chosen model implements the phoneme "r" and the chosen state is the second. We can see that the means start at a value close to zero, except for the mean of the zeroth coefficient, which starts off at a higher value. They need 2-3 iterations in order to settle to an almost constant value. The is a slight change in the values at the fifth iteration, but in most case the change is almost negligible.

The variances, on the other hand start of at much higher values. Even though they are more spread out, still most of them seem to be closer together, having the zeroth coefficient as an outlier. In the first three iterations we can observe a big change in all the values. After the third iteration the variances seem to behave similarly to the means. So depending on how much steadiness we need for our parameters, we could stop the training at the third iteration, or let it train more, in order to check weather it becomes more steady.

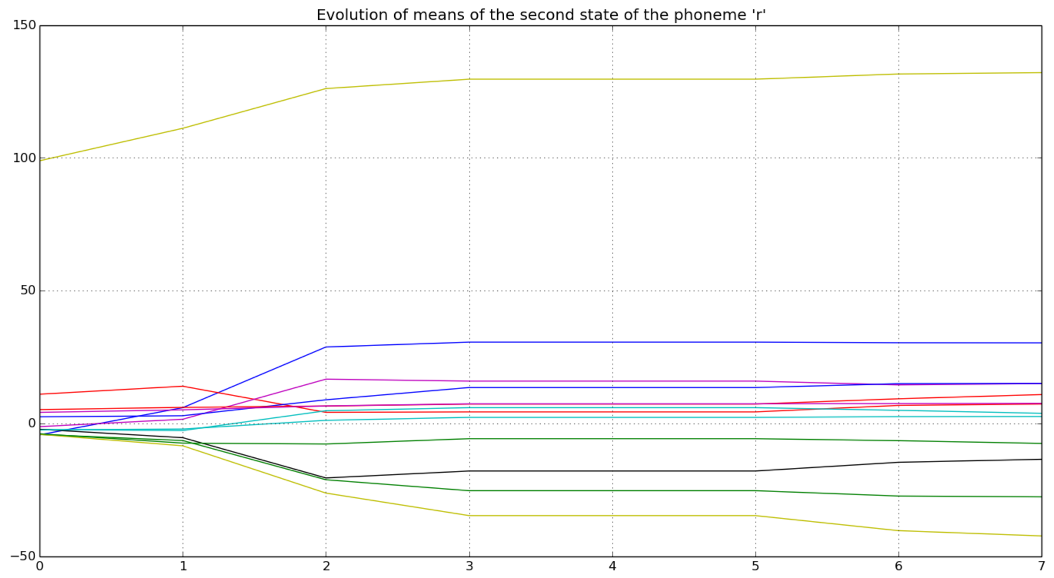

Figure 1: Evolution of means of the second state of the phoneme “r”

## 2.2 Parameters for different phonemes

For the last task of the Lab, we plot the means (MFCC features) of different phonemes. These features can be seen in Figure 3. First we can observe that this Figure is in accordance with the means’ behaviour in Figure 1. We can see that in the last iteration of Figure 1, the zeroth coefficient has a much higher value than the rest (similar to the start off values), which have values around zero.

Now, in Figure 3 we can see a similar behaviour. The zeroth coefficient of all three MFCC features has a high value, whereas the rest of the coefficients are all closer to zero. Of course there is an obvious difference between the “patterns” of the graphs between different phonemes. The graph representing the MFCC of “sil”, has a very low variance and remains almost steadily close to zero (an expected behaviour for this particular phoneme). On the contrary, the behaviour of “ao” has much larger differences between different coefficients. Finally the graph representing “s”, lies somewhere in the middle of the other two. It is not as steady as the “sound of silence”, but is much steadier that the sound of a vowel (“ao”).

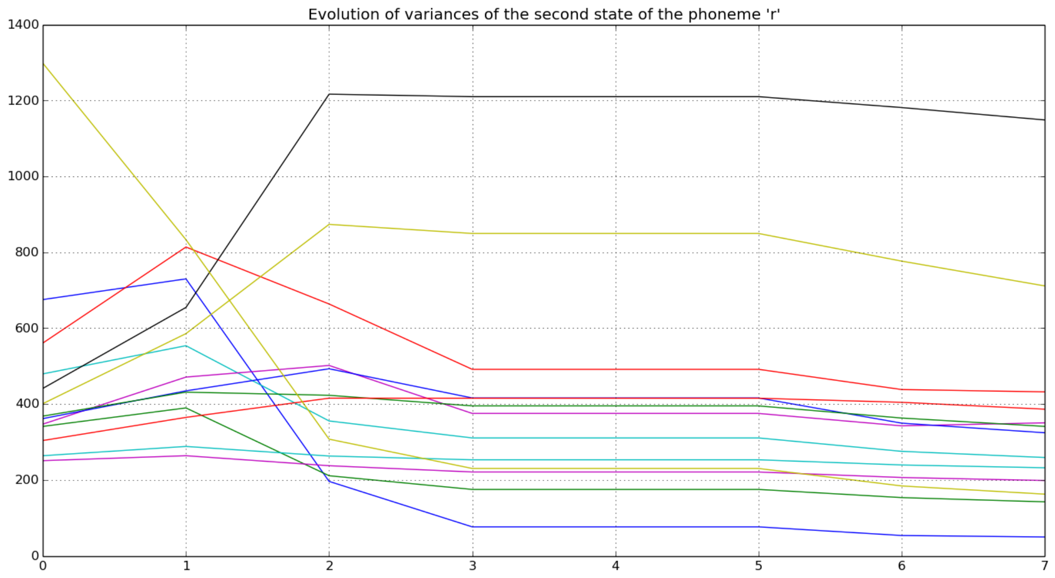

Figure 2: Evolution of variances of the second state of the phoneme "r"

These difference are much more visible in Figure 4, where we see the phonemes' features one step before their final form (which is used for the recognition and classification). Here the behaviours of the graphs, as described in the previous paragraph are more obvious, proving that these feature are indeed indicative of the phoneme that produced them.

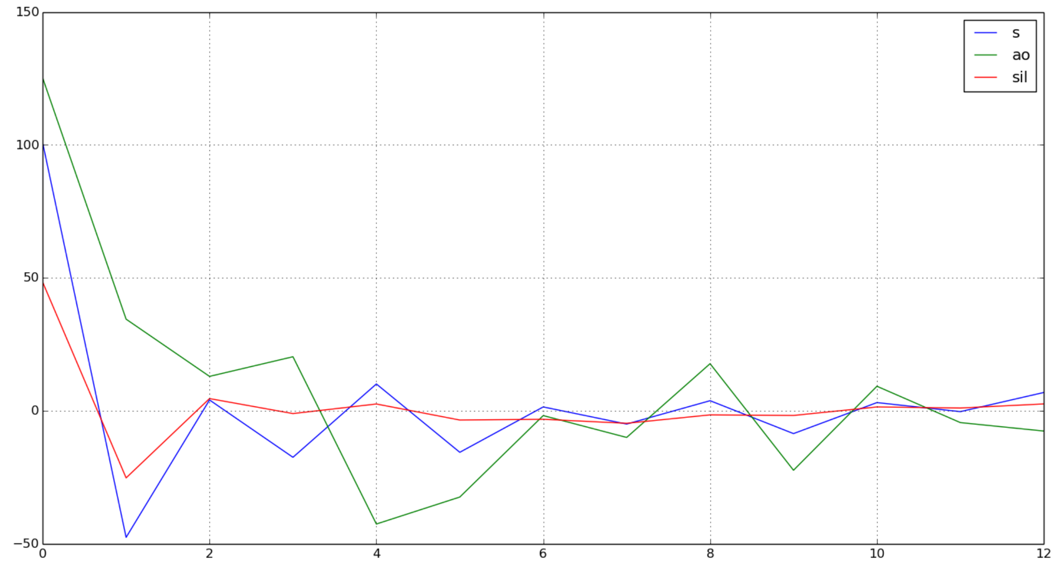

Figure 3: MFCC features for different phonemes

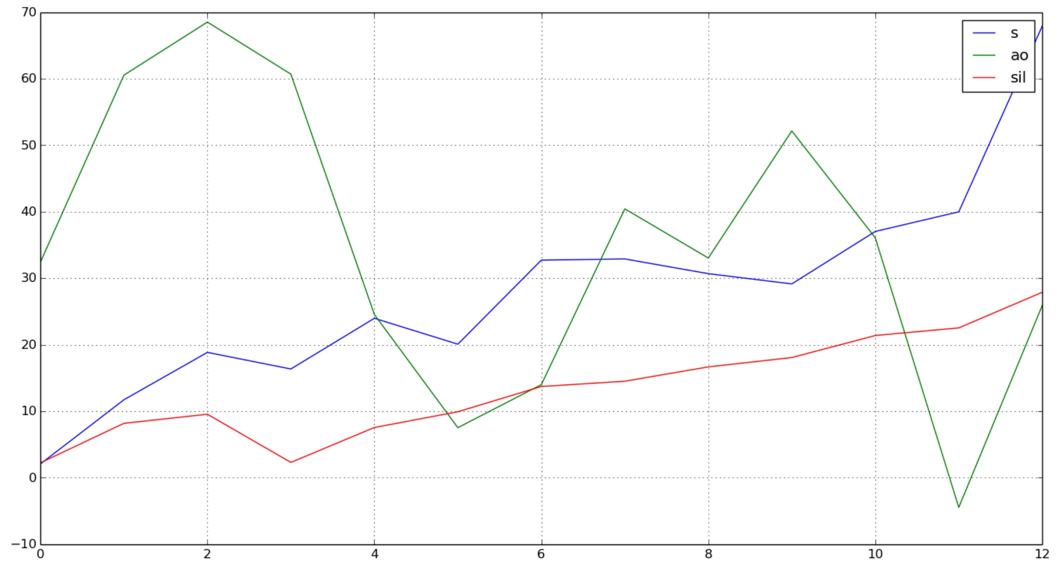

Figure 4: Filterbank outputs of features for different phonemes