SHS homework 9: recognizing vowels

Thirty-one distinct recordings of 12 vowels were made available by the students in homework 7 for use in the course (via licensing option **c** or **d**). We collected the 31 recordings in the dataset `vowels_course`, which you can download from Canvas.

The data is not too good. The recordings were made in noisy environments, typically with built-in laptop microphones. They are therefore much worse than the recordings behind the Pols et al. (1973) data contained in Praat (manual page **Discriminant analysis**). The TextGrids also varied a lot. Some included a bit of the surrounding fricatives, whereas some stayed away from the vowel edges by 20 milliseconds or so, and we kept the data that way (we fixed many other problems manually). We also didn't care about labelling the data as "male" or "female" (which is usually done in the literature), with the idea that good classification methods should learn not to care. Furthermore, we did not care about the accents of the speakers at all. In all, the dataset has the right level of variation and chaos to serve well as a small testbed for simple classification algorithms.

Homework 9.1. Drawing linear discriminant analysis

Work through the Praat manual page **Discriminant analysis** by hand first, and later on by script, using the `vowels_course.Table` data, containing speaker label, vowel label, duration, F1, F2 and F3 for all 372 vowels.

To make the table data suitable for discriminant analysis, you will first have to turn the Table object into raw numbers: select the Table, then do **Down to TableOfReal**, specifying `vowel` as the column for row labels. The result is a TableOfReal object with five columns: speaker, duration, F1, F2 and F3 (click **Inspect** to see what the row and column labels are). In logistic regression we would normally keep all these five predictors, but here we ignore speaker identity by removing the first column of the TableOfReal via **Modify** > **Remove column** (click **Inspect** to check that the speaker column is no longer there, and only duration, F1, F2 and F3 are left). You can now work through the **Discriminant analysis** manual using this TableOfReal.

Include in your report a scatter plot of F1 (or log F1, if you like that better) versus F2 (or log F2), and then a scatter plot along two eigenvectors, after determining which two eigenvectors yield the most convincing vowel space. Make sure that in the picture you use explicit ranges along the axis, not the default "0, 0, 0, 0". Make a third plot of one-sigma ellipses, as in the manual.

Homework 9.2. Assessing the quality of linear discriminant analysis

Determine how well the classifier has been trained, following section 4 of the manual. What is the fraction correct? This number will be the baseline for the more advanced methods that you can try on this data.

Finally, show the confusion table into your report (select your Confusion, then **Draw as numbers** into the Picture window, then copy–paste or PDF).

Homeworks 9.3 (& 9.4), 9.5 (& 9.6) and 9.7 can be done in any order, so if you get stuck with one of them, feel free to move to another.

Homework 9.3. Moving the data to Python

Any more sophisticated training will be done in Python, not in Praat. The first question to solve is how to get the data table into Praat. Write a Python script that takes the `vowels_course.Table` data and draws a scatter plot of F1 (or log F1) versus F2 (or log F2).

Copy–paste your script as well as the scatter plot into your report.

Homework 9.4. Doing discriminant analysis in Python

Determine what modules you need to import in Python to run linear discriminant analysis, and try to replicate in Python the following four things (as in **9.1** and **9.2**):

• A scatter plot of the data points along the two visually most illustrative eigenvectors.

• A plot of the one-sigma ellipses along the same axes.

• The fraction correct classification on the training data.

• The confusion table.

Include the script as well as the pictures in your report.

Homework 9.5. Learning from MFCCs

The duration-and-formant table was not very good input for the classifier. Mistakes in formant measurement due to background noise could, for instance, label an intended F2 as F1 or F3 instead. Perhaps it is better to work with a lower-level spectral representation than three formant levels. Such a lower-level spectral representation is Mel-Scale Cepstral Coefficients. Read Weenink chapter 17 for an explanation (forget about the Dynamic Time Warping and other parts of the text).

An MFCC object is a series of time frames with typically 12 coefficients in each frame (this number of 12 is only purely coincidentally the same as the number of Dutch vowels), just as a Formant object is a series of time frames with typically 5 formant frequencies and 5 bandwidths in each frame. In the same way that you retrieved a median formant value from a Formant object, you may want to retrieve an average value (averaged over all the times in the vowel interval) for each of the 12 MFCCs, simply because these vowels can be regarded as more or less stationary.

You write a Praat script to create a table file with MFCCs. Fortunately, the Praat script that created the table file with the durations and formants is already available as `createFormantTable.praat`, so you "only" have to adapt this script (select the Sound and use **Analyse spectrum** > **To MFCC**). The most difficult part is probably that for formants we had the **Get quantile** command, with which you could "average" over the time interval from *t1* to *t2*, whereas no such command is available for an MFCC object, so you have to average "by hand" in your script: get the relevant frame numbers in the MFCC associated

with the times *t1* and *t2* via **Query** > **Query time sampling** > **Get frame number from time**, then loop over all these frames to get the local values of c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11 and c12 (**Query** > **Get value in frame**) and add these values up to sum[1] through sum[12], dividing these sums at the end by the number of frames considered. This is how you get the 12 mean MFCCs for each of the 360 vowels. You should be able to create a table with the columns `speaker`, `vowel`, `duration`, `c1`, `c2`, `c3`, `c4`, `c5`, `c6`, `c7`, `c8`, `c9`, `c10`, `c11` and `c12`, and 372 rows.

Can you make this table perform better than the formant table did in discriminant analysis? Use either Praat or Python to find out.

(If you like spectrogram better than MFCCs, you can use a Spectrogram object instead of an MFCC object; only five lines of your script, or so, will be different.)

Homework 9.6. Shallow learning from MFCC

Write a shallow learning script in TensorFlow (if you don't have TensorFlow yet, just install it in your Python, forgetting about GPUs) on the basis of the MFCCs (or perhaps the formants, if you didn't manage to get the MFCCs to work, or the spectrogram, if you like that better than MFCCs).

Create a network with a single fully-connected layer, followed by a softmax layer classifying the 13-dimensional input (one duration and 12 MFCCs) into the 12-dimensional output (12 vowel classes). You can combine the softmax layer with the subsequent loss layer using `sigmoid_cross_entropy_with_logits`.

Anything you achieve here will help you next week when you start to work with deep learning.

Homework 9.7. Shallow learning from raw samples

Instead of using MFCCs, how well does it work to have the raw samples of the sound as input. As for Praat programming, you just have to adapt the formant analysis script in such a way that it writes columns for *t1* and *t2*.