Politecnico di Torino

ICT for Smart Societies

ICT for Health

Final report on the ICT for Health laboratories: Hidden Markov Machine (HMM)

Marzia Maffei
s260321

Professor: Monica Visintin

A. Y. 2018/2019

1 Introduction: Parkinson's disease

Parkinson's disease is a neurodegenerative condition which affects an estimated 1.2 million people in Europe alone. The incidence increases rapidly over the age of 65 and the ageing population will cause the number of people affected by Parkinson's to double by 2050.

The disease results from the degeneration of dopaminergic neurons in the substancia nigra and one of the main consequences of Parkinson's disease is the general loss of muscles control. In fact, patients affected by it often show resting tremor, rigidity, akinesia and bradykinesia and this loss of muscles control includes vocal chords, which causes speech impairment and voice disorders in 90% of the patients.

Treatment is available for a number of aspects of the disease, for example by using the drug Levodopa, but there is no actual cure.

1.1 Objective

The pronunciation of a vowel requires the vocal chords to periodically open and close the vocal tract, an ability that people affected by Parkinson's disease may not have. Therefore, the goal of this laboratory is to try and find a way to classify patients as ill or healthy starting from voice samples. These voice samples can be easily recorded, for example using a mobile phone, by the patients themselves and the diagnosis of the disease would be simpler for medical doctors.

1.2 The dataset

The dataset used in this laboratory is made of 20 .wav files. The files contain records of vowel "a" pronounced for a long time (given the patient's possibility) by 10 healthy people and 10 with Parkinson's disease. The sound was recorded using a mobile phone with sampling frequency equal to 8 kHz.

2 The algorithm

The algorithm aims at performing classification between healthy people and people affected by Parkinson's disease by means of Hidden Markov Machines (HMMs). A HMM models the system as a finite state machine with hidden (i.e. unobserved) states.

First all the 20 voice samples are processed and quantized by the function `pre_process_data`, then the quantized samples are used in the main code to train and test the two HMMs, one for healthy and one for ill people.

2.1 Function `pre_process_data`

The Fourier transform of the voice signal is characterized by several peaks, the one at the lowest frequency gives the fundamental frequency f_0. The signal is then analysed in the time domain and the peaks are found approximately at time intervals of $1/f_0$. Each interval is interpolated taking $N_s - 1$ samples and the newly obtained signal $x[n]$ is quantized using N_q values.

The function outputs two cell vectors: `hq` with the 10 quantized voice signals of the healthy people and `pq` with the 10 quantized voice signals of the people affected by Parkinson's disease.

2.2 Main code

In the main code, the values for N_s and N_q are set, along with the samples to be used for training and testing. Both for healthy and ill people, out of 10 samples, 7 are used for training and 3 for testing. The parameters for the HMM (tolerance, number of iterations and chosen algorithm) are also set at the beginning. The algorithm used in this laboratory is the Baum-Welch algorithm.

In the training phase the initial transition and emission matrices are created. The emission matrix $\mathbf{B}$ is generated randomly as a N_sxN_q matrix and then normalized along the rows. The transition matrix $\mathbf{A}$ is a N_sxN_s matrix and is generated in two different ways: randomly (and again normalized along the rows) and circularly. The circulant transition matrix has as first row the sequence $[q\, p\, q\, ...\, q]$, where p (set equal to 0.9) is the probability of going from state 1 to state 2 and $q = (1-p)/(N_s - 1)$, then all the other rows have the same sequence, each time shifted by one position to the right. The Matlab function `hmmtrain` is used to train the HMMs using the training sets of healthy and ill people and the two different transition matrices. As outputs the function gives the final emission and transition matrices to be used in the next step.

The testing phase uses the Matlab function `hmmdecode` to obtain the logarithm of the probability that a given sample belongs to that machine at the end of the process. The function is applied to training and testing sets, once using the matrices obtained from the randomly generated $\mathbf{A}$ and once using those obtained from the circulant $\mathbf{A}$. To evaluate the specificity the algorithm compares the probability of the true negative with the probability of the false positive, while the evaluation of the sensitivity is done comparing the probability of the true positive with the probability of the false negative.

3 Results

The algorithm has been implemented using three different sets for training and testing, as a sort of 3-fold cross validation:

- samples 1-7 as training and 8-10 as testing
- samples 4-10 as training and 1-3 as testing
- samples 1-3, 7-10 as training and 4-6 as testing

Several trials have been made by changing the parameters and all the following results are shown as an average over the three different training and testing sets.

The mean values of specificity and sensitivity for the two transition matrices and for different values of $N_s = N_q$ are displayed in *Table* 1. The results are very similar and, as it is to be expected, in both cases they are better for the training sets than for the testing ones. A difference can be seen looking at the specificities

and sensitivities for the different values of $N_s = N_q$. A small number of states gives poor results in comparison to the ones obtained with bigger values. At the same time it can be seen that values larger than $N_s = 8$ don't improve significantly the specificity or the sensitivity, actually they start to decrease.

	$N_s = 5$	$N_s = 7$	$N_s = 8$	$N_s = 9$	$N_s = 11$
specificity train	90.47%	95.24%	100.00%	100.00%	100.00%
specificity test	66.67%	66.67%	66.67%	77.78%	66.67%
sensitivity train	80.95%	90.47%	100.00%	95.24%	100.00%
sensitivity test	55.56%	88.89%	84.13%	88.89%	77.78%

(a) With randomly generated $\boldsymbol{A}$

	$N_s = 5$	$N_s = 7$	$N_s = 8$	$N_s = 9$	$N_s = 11$
specificity train	95.24%	95.24%	100.00%	100.00%	100.00%
specificity test	55.56%	77.78%	100.00%	88.89%	77.78%
specificity test	66.67%	100.00%	100.00%	95.24%	100.00%
specificity test	66.67%	88.89%	88.89%	88.89%	77.78%

(b) With circulant $\boldsymbol{A}$

Table 1: Mean values of specificity and sensitivity over the three different training and testing sets, with $N_q = N_s$, $tolerance = 10^{-3}$ and $number\,of\,iterations = 200$

	specificity train	specificity test	sensitivity train	sensitivity test
random **A**	95.24%	88.89%	80.95%	66.67%
circulant **A**	95.24%	66.67%	85.71%	66.67%

(a) $N\_s = 5$, $N\_q = 10$

	specificity train	specificity test	sensitivity train	sensitivity test
random **A**	100.00%	88.89%	90.47%	77.78%
circulant **A**	100.00%	88.89%	90.47%	77.78%

(b) $N\_s = 10$, $N\_q = 5$

Table 2: Mean values of specificity and sensitivity evaluated for $N_s \neq N_q$, with $tolerance = 10^{-3}$ and $number\,of\,iterations = 200$

Changes in the results are more clearly visible when using $N_s \neq N_q$, as shown in *Table* 2. If N_s is larger than N_q, specificity and sensitivity are generally higher than in the opposite situation. Comparing these results with the ones in *Table* 1, it is possible to see that there is no significant improvement in using $N_s > N_q$ rather than $N_s = N_q$, while, in general, values of N_s smaller than N_q yield slightly worse results, because there aren't enough states to describe the system.

Using different values of tolerance for the `hmmtrain` function gives the results shown in *Table* 3. As the tolerance increases, the specificities and sensitivities obtained tend to decrease slightly.

	specificity train	specificity test	sensitivity train	sensitivity test
random **A**	80.95%	77.78%	100.00%	88.89%
circulant **A**	100.00%	100.00%	100.00%	88.89%

(a) $tolerance = 10^{-2}$

	specificity train	specificity test	sensitivity train	sensitivity test
random **A**	90.47%	55.56%	80.95%	66.67%
circulant **A**	100.00%	77.78%	95.24%	88.89%

(b) $tolerance = 10^{-1}$

Table 3: Mean values of specificity and sensitivity for two different values of tolerance, with $N_s = N_q = 8$ and $number\,of\,iterations = 200$

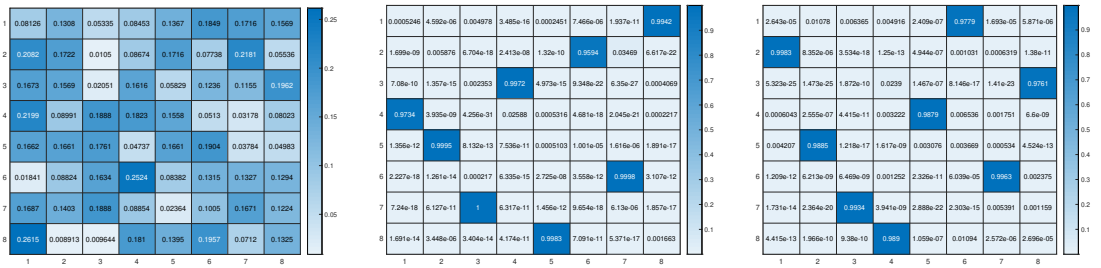
The value of N_s influences the running time of the algorithm, as *Table* 4 shows. The time elapsed for the function `pre_process_data` does not change significantly and neither does the time necessary for the testing phase. What makes the difference is the training phase. As the number of states increases, the dimensions of

	pre processing	training random **A**	training circulant **A**	testing	total
$N_s = 5$	4.1720	3.7599	0.9455	0.4126	8.3445
$N_s = 7$	4.2551	12.6753	1.4220	0.9541	19.3065
$N_s = 8$	4.3065	10.0792	1.7144	1.0200	17.1201
$N_s = 9$	4.3877	18.6249	3.2731	1.2881	27.5738
$N_s = 11$	4.6917	36.3316	6.1781	1.9067	49.1081

Table 4: Time elapsed in the different steps of the algorithm, using as parameters: $N_s = N_q$, $tolerance = 10^{-3}$ and $number\,of\,iterations = 200$

the transition matrix **A** increase too and this leads to a higher running time in the `hmmtrain` function. This is particularly evident in the training of the randomly generated **A**, because the initial probabilities are uniformly distributed, as it can be seen in *Figure* 1a.

The two final **A** obtained from the `hmmtrain` function are shown in *Figures* 1b and 1c.



(a) *Initial randomly generated* **A** (b) *Final random* **A** *for healthy samples* (c) *Final random* **A** *for ill samples*

Figure 1: Transition matrix **A** before and after the training phase

4 Conclusions

In general, variances in the parameters do not correspond to sensible changes on specificities and sensitivities for training and testing sets. Among all the parameters, it could be said that N_s affects the algorithm more than the others. It is possible to say that a small number of states gives poorer results, as does a number of states smaller than the number of quantization levels. For the dataset used in this laboratory, $N_s = N_q = 8$ was found to be the best choice, yielding to generally high values of specificity and sensitivity for all training and testing sets. The parameter N_s is also the one that mostly affects the running time of the algorithm, so setting it correctly may be a crucial step to obtain an efficient classification system.

The absence of really significant variations on specificity and sensitivity may be due to the small dataset used for this laboratory. The number of samples (10 for healthy and 10 for ill people) is not representative and is not enough to effectively train and test an efficient algorithm that could actually help the medical doctors in the diagnosis of the Parkinson's disease. Moreover, not all patients affected by the disease show speech impairment, so a classification algorithm based on voice signals should just be a starting point for the diagnosis.