HMM digit recogniser

Soufian SALIM

November 16, 2013

Abstract

Report on the HMM digit recogniser project for the Signal et Langue course, ATAL master.

Contents

1 Symbol sequence extraction

1.1 Extend the simu_symbol function to simulate a third digit, select for instance a "6"

```
simu_symbol <- function()
{
  (...)

  digit_6 <- rbind(
    stroke(0.25, 0.85, -0.4, -0.2, 10),
    stroke(-0.4, -0.2, 0, -1, 7),
    stroke(0, -1, 0.3, -0.5, 7),
    stroke(0.3, -0.5, -0.1, 0.3, 6)
  )

  dimnames(digit_6) <- list(num=1:nrow(digit_6), point=c("x", "y"))

  plot(digit_6, type="l", col="red", xlim=c(-1, 1), ylim=c(-1, 1))

  points(digit_6)

  return(list(d1=digit_1, d2=digit_4, d3=digit_6))
}
```

1.2 Consider the [*compute_symbol*] function to transform one digit in a sequence of symbols

Analyse this function and explain how it works.

The function takes the following as input: a digit trace, a number of rows (*nr*), and a number of columns (*nc*). It then takes the following steps:

1. It creates a matrix of *nr* * *nc* dimensions, each box containing a number from 1 to 15, from the top left to the bottom right
2. It calculates the number of points in the inputted digit
3. It computes the horizontal position of each point
4. It computes the vertical position of each point
5. It computes, for each point, the corresponding box in the first matrix, and outputs it

Test it with the simulated digits.

In order, output for digit "1", digit "4" then digit "6":

```
[1] 11 11 14 14 14 14 14 14 14 14 14 14 14 14 11 11 11 11  8  8  8
     8  5  5  5  5  2  2  2  2
[1] 14 14 14 11 11 10  7  7  7  4  4  4  4  4  5  8  8  8  9  9  8
     8  8  5  5  5  2  2  2  2
[1] 14 14 14 11 11 11  8  8  8  7  7  5  5  2  2  2  2  2  2  2  2
     2  5  5  5  5  8  8  8 11
```

From your point of view what are the strengths and weaknesses of this method.

This method allows for efficient resampling of a graphical digit (or other character). With enough rows and columns, it could technically differenciate between a digit and and exponent, or any other form of mathematical notation.

Another advantage of this method is that the number of columns and rows does not affect the length of the outputted sequence, which is solely dependent on the number of points in the digit trace.

Its weakness, however, is that it can only be used for "online" writing (or would require strong preprocessing), since the order of each point is critical to the sequence construction.

1.3 Now let us consider [*compute_symbol_dir*] to transform one digit in a sequence of symbols

Analyse this function and explain how it works.

The function takes the following as input: a digit trace, and a number of angle classes (*nangles*). It then computes the angle of a segment between two points, and classify it in one of *nangles* angle classes.

Test it with the simulated digits.

In order, output for digit "1", digit "4" then digit "6":

```
[1] "1:"
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
    23 24 25 26 27 28 29 30
 2  2  2  2  2  2  2  2  2  7  7  7  7  7  7  7  7  7  7  7  7  7
     7  7  7  7  7  7  7  7
[1] "4:"
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
    23 24 25 26 27 28 29 30
 6  6  6  6  6  6  6  6  6  5  1  1  1  1  1  1  1  1  1  4  7  7
     7  7  7  7  7  7  7  7
[1] "6:"
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
    23 24 25 26 27 28 29 30
 6  6  6  6  6  6  6  6  6  1  8  8  8  8  8  8  1  2  2  2  2  2
     2  1  4  4  4  4  4  4
```

From your point of view what are the strengths and weaknesses of this method.

Like with the previous method, one of the advantages is that the length of the sequence is not affected by the number of angle classes. Also like with the previous method, the order of each point is important and it cannot be used if the correct order is unknown.

Unlike the previous method, since this one uses angles to resample the digits, it might allow for more subtle distinction between symbols (for example, the "1" and the british-style handwritten "7": both might be in the same "boxes" after resampling with the first method, but this one will show an angular difference).

2 Selection of a HMM topology and initial values

2.1 Give a graphical representation of this model

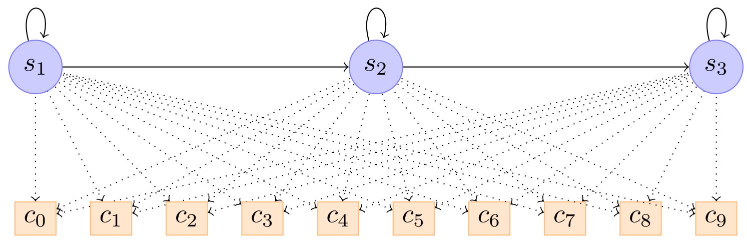

Figure 1: An HMM with 3 states which can emit 10 symbols c. In this particular HMM, states can only reach themselves or the next state. Each state can emit any symbol.

2.2 How would you initialise the vector of initial probabilities, and the state transition matrix, knowing that the length of the sequences are T = 30?

Initial probabilities:

1	0	0

Given that each state is in charge of generating one part of the digit, the first state models the beginning of the digit. Therefore the first state must always have an initial probability of 1, and the others of 0.

State transition matrix:

	s1	s2	s3
s1	0.9	0.1	0.0
s2	0.0	0.9	0.1
s3	0.0	0.0	1.0

Since a sequence is of length T = 30, and there are three states, on average there should be 1 in 10 chances of moving to the next state and 9 in 10 of staying on the same state (length of sequence / number of state), because we want to stay on each state roughly the same amount of time.

2.3 How about the observation matrix?

Without any more information on the digits we can only start with a uniform distribution for the observation matrix.

3 Training of the models

3.1 Create a first function *initHMMDigit* which takes as input the number of states and of possible observations, and initialise the transition matrix with a left-right architecture and with uniform distribution of observations from each state

```
initHMMDigit <- function(nsymbols=15, nstates=3, uniform=F)
{
  states <- c(paste("Tier", 1:nstates))
  symbols <- 1:nsymbols
  startProbs <- c(1, c(rep(0, nstates-1)))

  if (uniform) distr <- c(0.5, 0.5) else  distr <- c(1 - nstates /
      30, nstates / 30)

  transProbs <- matrix(c(
    rep(c(distr, rep(0.0, nstates-1)), nstates-1), 1.0
    ), nrow=nstates, ncol=nstates, byrow=T
  )

  emissionProbs <- matrix(
    rep(1/nsymbols, nsymbols * nstates),
    nrow=nstates, ncol=nsymbols, byrow=T
  )

  hmm <- initHMM(states, symbols, startProbs, transProbs,
      emissionProbs)
}
```

3.2 Have a look at the different models after training. Do they make sense for you?

Their probability to move on to the next state has slightly reduced, therefore limiting lightly the time spent on the last state.

The emission probabilities are no longer uniformly distributed; altough not that significantly.

3.3 Select a specific model of digit, and consider the symbols that will be most often emitted during states 1, 2 and 3

For the 1, for example, using a model trained on data generated with the first sampling method (3x5), we can see that emissions probabilities are higher for symbols 2, 5, 8, 11 and 14. These symbols correspond to the "middle boxes" in the sampling method.

4 Recognition performances

4.1 Use the corresponding test files, and run the forward algorithm to build the confusion matrix for these four cases, display the global recognition rates

For the "3_5" type, with 3 states, optimal distribution:

0	1	2	3	4	5	6	7	8	9	
0	**121**	1	0	0	0	2	1	0	4	0
1	1	**104**	23	0	1	0	0	3	1	3
2	4	1	**102**	1	8	0	4	2	1	15
3	0	0	2	**128**	1	0	0	3	0	10
4	1	3	13	11	**104**	1	9	4	0	5
5	1	3	1	33	0	**72**	8	1	14	4
6	2	0	0	1	3	0	**135**	1	1	0
7	0	0	0	0	11	0	2	**111**	3	0
8	2	0	0	0	1	0	0	2	**123**	0
9	0	0	8	8	0	0	0	1	2	**110**

Recognition Rate : 81.4978%

For the "dir_8" type, with 3 states, optimal distribution:

0	1	2	3	4	5	6	7	8	9	
0	**28**	1	0	28	1	0	40	0	31	0
1	53	**2**	0	10	4	6	4	0	57	0
2	26	6	**0**	36	1	5	25	0	39	0
3	14	28	0	**35**	0	0	9	1	57	0
4	43	2	0	18	**23**	24	11	8	22	0
5	31	9	20	8	0	**4**	17	0	48	0
6	21	1	0	39	1	3	**22**	0	56	0
7	39	25	2	12	2	5	15	**0**	26	1
8	16	5	8	26	1	3	19	0	**50**	0
9	9	0	0	48	0	1	14	0	57	**0**

Recognition Rate : 12.04112%

4.2 Replace “the optimal initial values” of the models by uniform values (in initHMMDigit). Redo the training and check the new corresponding results

4.3 Keep the best initialisations, and repeat these same experiences with different number of states, and try to find the optimal number of states

5 Classifier combination

5.1 Sum rule

5.2 Borda count

5.3 Train a ANN using the normalized (in [-1,1]) observations as inputs and combine it with HMM thanks to on of the previous rule

6 Attachments

Code, data and models can be found within this folder.