UNIVERSITÉ DE NANTES

Master ATAL – X9IT110

Project

HMM digit recogniser

Introduction

The goal with this exercise is to implement a full digit recogniser, to train and test it on the IRONOFF dataset.

Figure 1 shows an overview of the global system that will be developed

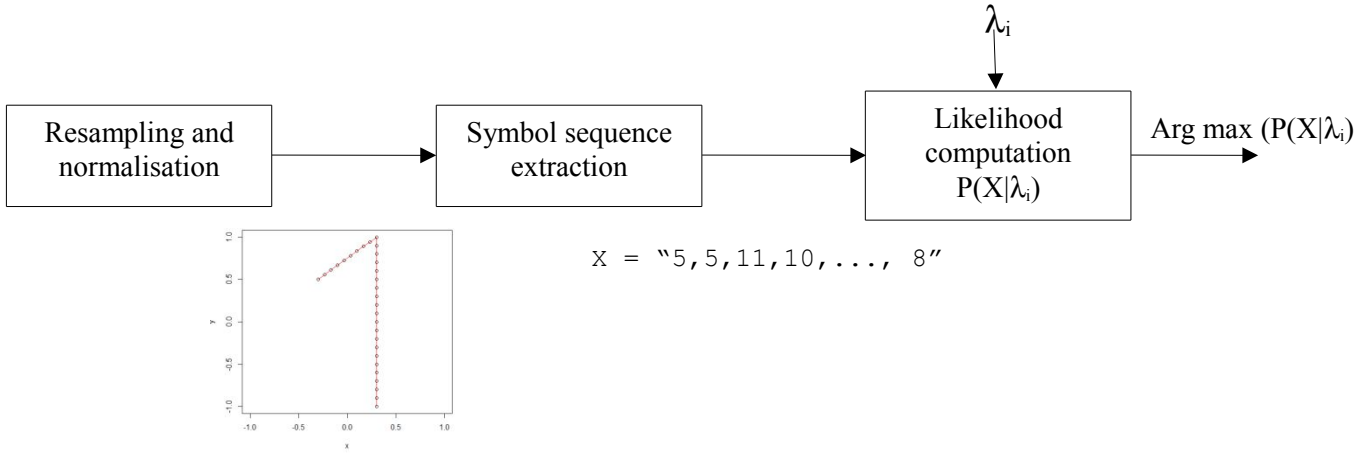

Figure 1. An HMM digit recogniser

We suppose that re-sampling and normalisation have already being done, so that each digit is centred in the box [-1,+1] ×[-1,+1] and contains exactly 30 equidistant points.

I. Symbol sequence extraction

The functions given below simulate and display the two digits: '1' and '4'.

```
stroke <-function(x0=-1,y0=-1,x1=1,y1=1,N=10)
{
  strk <- matrix(c(seq(x0,x1,length=N),seq(y0,y1,length=N)),ncol=2)
  return(strk)
```

```
}
simu_symbol <- function()
{
  digit_1 <- rbind(stroke(-0.3,0.5,0.3,1.0,10),stroke(0.3,0.9,0.3,-1.0,20))
  dimnames(digit_1) <- list(num=1:nrow(digit_1),point=c("x","y"))
  plot(digit_1,type="l",col="red",xlim=c(-1,1),ylim=c(-1,1))
  points(digit_1)
  digit_4 <- rbind(stroke(0.2,1.0,-0.8,-0.3,10),stroke(-0.85,-0.32,0.5,-
0.1,10),stroke(0.3,0.1,0.2,-1.0,10))
  dimnames(digit_4) <- list(num=1:nrow(digit_4),point=c("x","y"))
  plot(digit_4,type="l",col="red",xlim=c(-1,1),ylim=c(-1,1))
  points(digit_4)
  return(list(d1=digit_1,d2=digit_4))
}
```

1. Extend the `simu_symbol` function to simulate a third digit, select for instance a '6'.
2. Consider the following function to transform one digit in a sequence of symbols.

```
compute_symbol <- function (trace,nr=5,nc=3)
{
  LUT <- matrix(1:(nr*nc),nrow=nr,ncol=nc,byrow=T)
  NB <- length(trace[,"x"])
  Ix <- pmax(pmin(1+floor((trace[,"x"]-(-1))*nc/2),rep(nc,NB)),rep(1,NB))
  Iy <- pmax(pmin(1+floor((trace[,"y"]-(-1))*nr/2),rep(nr,NB)),rep(1,NB))
  return(LUT[matrix(c(Iy, Ix),ncol=2)])
}
```

Analyse this function and explain how it works. Test it with the simulated digits. From your point of view what are the strengths and weaknesses of this method.

3. Now let us consider another function to transform one digit in a sequence of symbols.

```
compute_symbol_dir <- function (trace,nangle=8)
{
  NB <- length(trace[,"x"])
  delta <- trace
  delta[1:(NB-1),] <- delta[2:NB,]
  delta <- delta - trace
  delta[NB,] <- delta[NB-1,]
  angle <- atan2(delta[,"y"],delta[,"x"]) + pi/nangle
  angle[angle < 0] <- angle[angle < 0] + 2*pi
  angle <- pmin(1 + floor(angle*nangle/(2*pi)),nangle)
  return(angle)
}
```

Analyse this function and explain how it works. Test it with the simulated digits. From your point of view what are the strengths and weaknesses of this method.

II. Selection of a HMM topology and initial values

First, we select a HMM with N = 3 states, and a left-right topology.
We assume that each state is in charge of generating one part of the digit: first state models the beginning of the digit, second state models the middle part and the third state models the end of the digit.

1. Give a graphical representation of this model.
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?
3. What about the observation matrix?

For the next questions, we will use a new HMM toolbox similar to the one we studied during the last practical work session. Thus, the script *newHMM.r* is provided to you and it contains the main functions for working with HMMs. Note that the new function *baumWelchList* does a similar training as baumWelch function but using a list of observation sequences (as a matrix) instead just one observation sequence. This allows you to train an HMM using all symbols from a class.

III. Training of the models

We have already pre-computed the sequences of symbols for the whole IRONOFF digit dataset. Several versions are available. They correspond either to `compute_symbol` or to `compute_symbol_dir` functions, with different parameter values.

Train_compute_symbol_5_3DigitX.txt (15 symbols)
Train_compute_symbol_5_4 DigitX.txt (20 symbols)
Train_compute_symbol_dir_8 DigitX.txt (8 symbols)
Train_compute_symbol_dir_16 DigitX.txt (16 symbols)

with X = 0 .. 9

These files can loaded with function `Load_Obs` form script file usefullTools.r

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. As a parameter you can choose between two kind of transition matrix:

1. probability to stay in the same state of 0.5 and to go to the next state of 0.5 (called uniform configuration)
2. probability to stay in the same state of 0.9 and to go to the next state of 0.1 (called optimal)

For each training files, initialize and train the 10 different models (λ_0 to λ_9) with the Baum-Welch algorithm and the optimal configuration.

Have a look to the different models after training. Do they make sense for you? Select a specific model of digit, and consider the symbols that will be most often emitted during states 1, 2 and 3.

Save the 10 HMM models in a matrix structure (using *rbind* concatenation R function).

IV. Recognition performances

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.
   Hints: define a function *classify* which takes as input the 10 HMMs and one observation sequence and give as output the probability fro each HMM of observing the sequence.

2. Replace "the optimal initial values" of the models by uniform values (in *initHMMDigit)*. Redo the training and check the new corresponding results.

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

V. Classifier combination

Select the two best configurations from your previous experiments, and implement a combination of these classifiers, using:

1. A sum rule
2. A Borda count
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

Describe these two combination rules and compare the different results.

VI. Any other proposals are welcome

- compare results on test sets and training sets
- display Top(N) recognition rates
- imagine new symbol sets and define a function to compute the corresponding sequence
- combine more than two classifiers
- add a reject criteria to increase recognition rate, display ROC curve
- train an ANN to do the combination of several classifiers
- offline description : a 32 observation sequence, with a quantization on N values (8 may be enough) => ask us for the raw data to define your own features...

Figure 2.12: Illustration of periphery features [LWC+10].