# Code Verification and Signal Database

Pattern Recognition Assignment 4

Rendani Mbuvha, Huijie Wang
rendani@kth.se, huijiew@kth.se

October 2016

## 1 Code Verification

In this section, *backward* function is verified and the corrected code can be seen in the attached files.

### 1.1 Correction

Since the logic of the code is aligned with the procedure of *backward*, firstly, the logical structure of the function is checked. Two steps are included in backward, namely *initialization* and *BackwardStep*. In the **Backword_fixme** function, line 47-53 are definiation and initialization of parameters, line 54-64 are corresponding to *initialization* and line 65-75 are corresponding to *BackwardStep*.

Then we look detailed into algorithm and evaluate the performance. In this step, the error message from Matlab greatly helps when debugging. Two errors are found:

- **Undefined variable 'z'**

  The error is shown as: *Undefined function or variable 'z'. Error in MarkovChainbackward (line 61) z(k)+z(k)+1;*

  The **syntax error** occurs as $z$ has never been defined before, thus, no space is assigned to parameter $z$. On the other hand, the correct opearte of the line should be `z=z+1`.

  However, since discarding the parameter $z$ and $k$ as well as relavant lines has no influence to the accuracy of the function, we simply delete Line 60-62.

- **Index out of Bounds Error**

  The error is shown as: *Attempted to access betaHat(3,1); index out of bounds because size(betaHat)=[2,3]. Error in MarkovChain/backward (line 69) summ=summ+A(i,j)*pX(i,t+1)*betaHat(t+1,j);*

  Index out of bounds error is a kind of **run-time error**. Since `betaHat` is a $nS \times T$ matrix(in our test case $nS = 2$ and $T = 3$), and it stores values with respect to every time step columnwise, `betaHat(t+1,j)` should be replaced by `betaHat(j,t+1)`.

Having fixed these errors, still the function could not provide correct answer even though Matlab doesn't throw errors anymore. That is because **logic errors** exist in the function. These errors won't crash the function but give incorrect operation of parameters. **Logic errors** are fixed according to steps of backwards stated in Course Book and these errors are listed as following, where line numbers are corresponding to the original code.

- Line 55: `betaHat(:,T)=betaHat(:,1)./c(T);`
  Since the initialization for infinite duration HMM is $\hat{\beta}_{i,T} = 1/c_T$, `betaHat(:,1)` should be replaced by `ones(nS,1)`.
- Line 58: `betaHat(i,T)=A(i,nS+1)/(c(1)*c(T+1));`
  `c(1)` should be replaced by `c(T)`.
- Line 65-66: `for i=1:nS, for t=T-1:-1:1`
  Since $\hat{\beta}$ is updated on each time step, the order of the loop should be changed, i.e. `for t=T-1:-1:1, for i=1:nS`
- Line 68: `for j=i:nS`
  What is expected to calculated is $sum_{j=1}^{N}$, which translated to code should be a loop `for j=1:nS`. If we started with $i$, part of the values will simply be ignored.
- Line 69: `summ=summ+A(i,j)*pX(i,t+1)*betaHat(t+1,j);`
  Despite the syntax error stated in the former paragraph, a logic error is that `pX(i,t+1)` should be replaced by `pX(j,t+1)`. The corrected line should be: `summ=summ+A(i,j)*pX(j,t+1)*betaHat(j,t+1);`
- Line 72: `betaHat(i,t)=(1/c(1))*summ;`
  `c(1)` should be replaced by `c(t)`.

At last some redundant lines are removed from the function. Some parameters although doesn't cause any error when running the function, would be redundant. As stated before, Line 60-62:

```
1          for k=1:10
2              z(k)+z(k)+1;
3          end
```

, are deleted for `z` and `k` are redundant. Also, anathor parameter `FinalProb` never has a practical use in the function. Discarding the parameter and relavant lines it has no influence to the accuracy of the function. As a result, the following lines are also discarded:

- Line 48: `FinalProb=.01;`
- Line 59: `FinalProb=.01+ FinalProb;`
- Line 70: `FinalProb=mean(FinalProb);`

## 1.2 Verification

### 1.2.1 Verification of finite HMM

Verification of the code is carried out according to the project instruction. A finite duration HMM is given by:

```
q = [1;0];
A = [0.9  0.1  0
      0  0.9  0.1];
mc = MarkovChain(q, A);
g1 = GaussD('Mean', 0, 'StDev', 1) ;%Distribution for state=1
g2 = GaussD('Mean', 3, 'StDev', 2) ;%Distribution for state=2
h = HMM(mc, [g1; g2]); %The HMM
```

Given observation sequence $x = [-0.2, 2.6, 1.3]$;, as a result, the output gives:

```
alfaHat =
    1.0000    0.3847    0.4189
         0    0.6153    0.5811
c =
    1.0000    0.1625    0.8266    0.0581
betaHat =
    1.0000    1.0389         0
    8.4154    9.3504    2.0818
```

, which is align to the project instruction.

### 1.2.2 Verification of infinite HMM

In order to test infinite duration HMM, we define a discrete HMM model:

```
    q = [1;0];
    A = [0.9  0.1
         0.1  0.9];
    B  = [0.2 0.5 0.3;
          0.7 0.1 0.2];
```

, and given observation sequence `x = [1, 3, 2];`, the result of the function is:

```
    c =0.2, 0.29, 0.437931
    betaHat =
        5.0000    3.6220    2.2835
        1.5354    1.1024    2.2835
```

To verify whether it is correct, we manually calculated `betaHat` to compare for the result:
**Initialization**: $\hat{\beta}_{1,T} = \hat{\beta}_{2,T} = 1 \div c_T = 1 \div 0.437931 = 2.28346$, where $T = 3$.
**Backward step:** The claculation is updated as: $\hat{\beta}_{i,t} = \frac{1}{c_T} \sum_{j=1}^{N} a_{ij} b_j(x_{t+1}) \hat{\beta}_{1,t+1}$.
The calculation is shown as:

- **t = 2**: observed $x = 2$ when $t = 3$,

$$b_j(x_{t+1}) = \begin{pmatrix} 0.5 \\ 0.1 \end{pmatrix}$$

$\hat{\beta}_{1,2} = \frac{1}{0.29} \times (0.9 \times 0.5 \times 2.28346 + 0.1 \times 0.1 \times 2.28346) = 3.62204$
$\hat{\beta}_{2,2} = \frac{1}{0.29} \times (0.1 \times 0.5 \times 2.28346 + 0.9 \times 0.1 \times 2.28346) = 1.10236$
**t = 1**: observed $x = 3$ when $t = 2$,

$$b_j(x_{t+1}) = \begin{pmatrix} 0.3 \\ 0.2 \end{pmatrix}$$

$$\hat{\beta}_{1,1} = \frac{1}{0.2} \times (0.9 \times 0.3 \times 3.62204 + 0.1 \times 0.2 \times 1.10236) = 4.99999$$
$$\hat{\beta}_{2,1} = \frac{1}{0.2} \times (0.1 \times 0.3 \times 3.62204 + 0.9 \times 0.2 \times 1.10236) = 1.53543$$

As a result, $\hat{\beta}$ is very similar to the result produced by matlab.

## 2 Character Database

We have created a character database that would be utilized for a simple online Handwritten Calculator. The calculator will perform three mathematical operations - Addition, Multiplication and division.

The user will simply enter the numerical characters together with the operators required to calculate the value of the expression.

The database will therefore consist of the following characters "1", "2", "3", "4", "5", "6", "7" ,"8","9","0","+", −,"×" and "÷".

Each of the 13 characters will be repeated 15 times for training of the HMMs. These sample will be obtained from different individuals. Naturally there will be some variation some random hand movements when the pen is down. However the overall patterns and style of drawing will remain similar.

For characters where the are distinctly different drawing styles like "8" and "÷" we will consider creating different databases for such characters and thus all styles will map to the same character. Below are two sets of samples that were obtained from the character database.

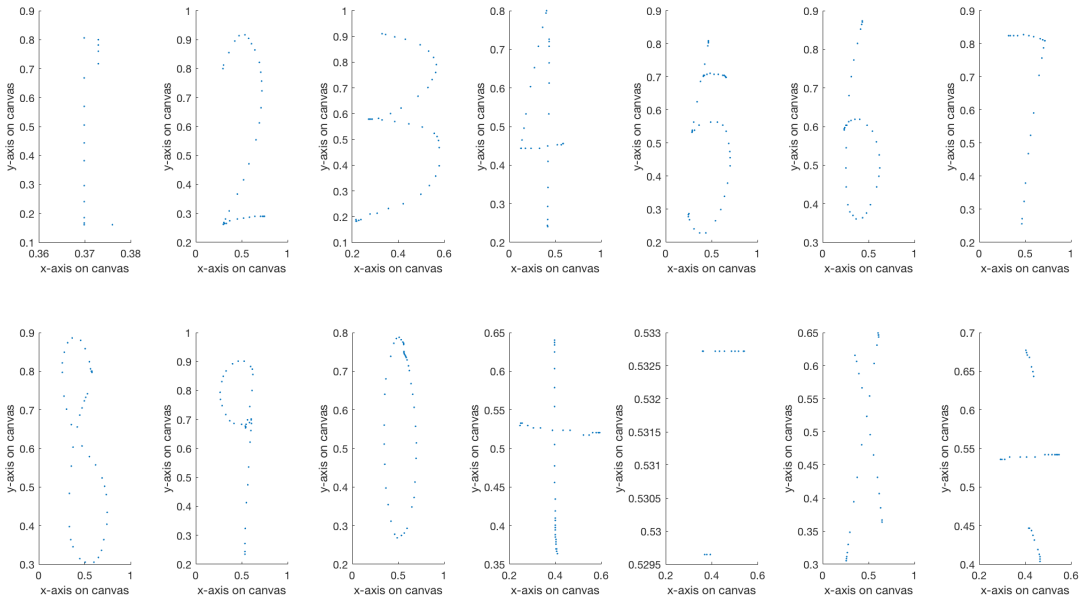


Figure 1: Sample 1 from the character database

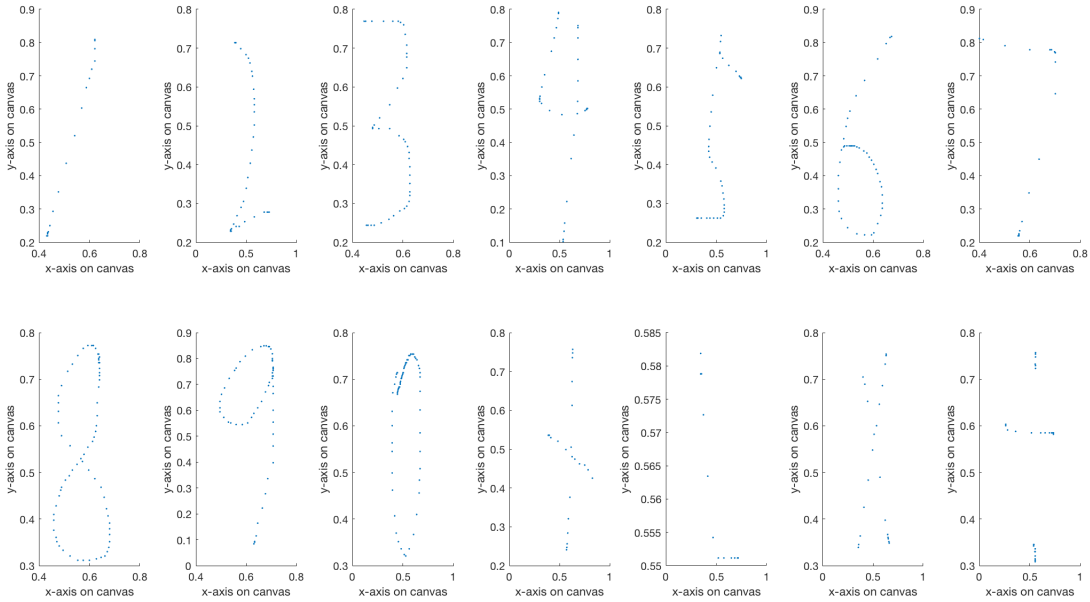


Figure 2: Samples 2 from the character database