Viterbi Algorithm

전기생체공학부 전기공학전공

홍종혁

목차

1. 전체 코드

```
%% Settings
clear
clc
error=0;
A=[0.975 0.025;0.025 0.975];
p=0.1 or 0.05 or 0.01; (셋 중 하나로 설정)
B=[1-p p;p 1-p];
testnum=1000;

%% Evaluate the Average Error Probability
for i=1:testnum
    [y,z]=make(p);
    zhat=viterbi(A,B,y);
    cnt=test(zhat,z);
    if cnt~=0
        error=error+cnt;
    end
end
s=num2str(error/(testnum*0.01));
disp("Error probability is "+s+"%")

%% Check Error
function cnt=test(zhat,z)
cnt=0;
len=length(z);
for t=1:len
    if zhat(t)~=z(t)
        cnt=1;
        break
    end
end
end

%% Viterbi Algorithm
function zhat=viterbi(A,B,y)
len=length(y);
zhat=zeros(1,len);
prob=zeros(2,len);
prob(:,1)=B(:,1).*[1;0];
index=zeros(2,len);
for t=2:len
    [prob(1,t),index(1,t)]=max(prob(:,t-1).*A(:,1).*B(1,y(t)+1));
    [prob(2,t),index(2,t)]=max(prob(:,t-1).*A(:,2).*B(2,y(t)+1));
end
[~,argmax]=max(prob(:,len));
zhat(len)=argmax-1;
for t=len:-1:2
    zhat(t-1)=index(zhat(t)+1,t)-1;
end
end

%% Make Hidden Sequence and Observation Sequence
function [y,z]=make(p)
z=zeros(1,100);
```

```
y=zeros(1,100);
y(1)=0;
previous=0;
state=[1 0];
for t=2:100
    zd=rand(1);
    if zd<0.975
        z(t)=previous;
    else
        z(t)=state(previous+1);
        previous=z(t);
    end
    yd=rand(1);
    if yd<1-p
        y(t)=z(t);
    else
        y(t)=state(z(t)+1);
    end
end
end
```

2. 알고리즘 설명

(1) Hidden Sequence 및 Observation Sequence 생성

```
%% Make Hidden Sequence and Observation Sequence
function [y,z]=make(p)
z=zeros(1,100);
y=zeros(1,100);
y(1)=0;
previous=0;
state=[1 0];
for t=2:100
    zd=rand(1);
    if zd<0.975
        z(t)=previous;
    else
        z(t)=state(previous+1);
        previous=z(t);
    end
    yd=rand(1);
    if yd<1-p
        y(t)=z(t);
    else
        y(t)=state(z(t)+1);
    end
end
end
```

이 함수는 Observation에 연관된 확률 p를 입력받는다. 우리는 z(1)=0 임을 알고 있다. 따라서 y(0)의 값도 사전에 설정할 수 있다. y(1)=0이다. 그 후 z(t-1)로부터 z(t)의 값을

얻기 위해서, 이전 state의 값을 의미하는 변수 previous를 지정한다.

for 문을 이용하여, t=2에서 t=100까지 반복한다. z(t)를 결정하기 위한 변수 zd를 생성한다. rand 함수를 이용해서 0과 1 사이의 랜덤 난수를 zd에 저장한 후, 랜덤 난수가 0.975보다 작다면 z(t-1)과 같게, 크거나 같으면 z(t-1)과 다르게 설정한다. observation y도 마찬가지로 rand 함수를 이용하여 1-p보다 작다면 z(t)와 같은 값을, 크거나 같으면 z(t)와 다른 값을 y(t)에 저장한다.

이 함수는 Hidden Sequence z와 Observation Sequence y를 반환한다.

(2) Viterbi Algorithm

```
%% Viterbi Algorithm
function zhat=viterbi(A,B,y)
len=length(y);
zhat=zeros(1,len);
prob=zeros(2,len);
prob(:,1)=B(:,1).*[1;0];
index=zeros(2,len);
for t=2:len
    [prob(1,t),index(1,t)]=max(prob(:,t-1).*A(:,1).*B(1,y(t)+1));
    [prob(2,t),index(2,t)]=max(prob(:,t-1).*A(:,2).*B(2,y(t)+1));
end
[~,argmax]=max(prob(:,len));
zhat(len)=argmax-1;
for t=len:-1:2
    zhat(t-1)=index(zhat(t)+1,t)-1;
end
end
```

Viterbi Algorithm을 수행하는 함수이다. State Transition matrix A와 Measurement matrix B, Observation y를 입력받는다.

y의 길이를 저장할 변수 len을 선언하고 Estimation 값을 저장할 배열 zhat을 1 by len size로, Dynamic Programming을 위해 확률을 담을 행렬을 prob이라는 이름으로 2 by len size로 선언한다. 반복문을 수행하기 전에 사전 확률을 설정해 둔다. 우리는 z(1)=0임을 알고 있으므로, 사전 확률은 **B(:,1).*[1;0]**, 즉 **[0.9;0]**이 된다. 그리고 max확률이 어느 경로로 왔는지 index를 저장할 행렬을 2 by len size로 선언한다.

반복문을 t=2에서 t=len까지 수행한다. 이 때 두 가지의 식

$\delta_j(t) = \max_i\{\delta_i(t-1)a_{ij}b_{jy_t}\}, \psi_j(t) = \operatorname{argmax}_i\{\delta_i(t-1)a_{ij}b_{jy_t}\}$를 state의 개수만큼 수행한다. state의 개수가 2개이므로 for 문을 사용하지 않고 그냥 작성하였다.

for문이 끝난 후, prob 행렬의 마지막 열의 값 중 가장 큰 값의 index를 얻어와 zhat의 마

지막 위치에 얻어 온 index에 해당하는 값을 저장한다. 그 후 다시 for문을 사용하여 Backward direction으로 최적의 경로를 찾아서 zhat에 Estimated Value를 저장한다.

이 함수는 Estimated Sequence zhat을 반환한다.

(3) Error 확인

```
%% Check Error
function cnt=test(zhat,z)
cnt=0;
len=length(z);
for t=1:len
    if zhat(t)~=z(t)
        cnt=1;
        break
    end
end
end
```

오류 측정을 위한 함수이다. Estimated Sequence zhat과 Hidden Sequence z를 입력받는다.

오류 측정을 위한 변수 cnt를 선언한 후, z의 길이를 변수 len에 저장한 후 t=1에서 t=len까지 반복한다. $zhat(t) \neq z(t)$인 경우가 단 한 번이라도 있었다면, cnt=1로 설정하고, 반복문을 종료한다.

이 함수는 cnt값을 반환한다.

(4) Model Setting

```
%% Settings
clear
clc
error=0;
A=[0.975 0.025;0.025 0.975];
p=0.1 or 0.05 or 0.01; (셋 중 하나로 설정)
B=[1-p p;p 1-p];
testnum=1000;
```

모든 변수와 명령창을 초기화 해주고, error 발생 횟수의 측정을 위한 변수 error를 선언한다. A와 B, 그리고 p를 문제에서 주어진 대로 설정해준다. testnum은 검사 시행 횟수이다.

(5) Evaluate the Average Error Probability

```
%% Evaluate the Average Error Probability
for i=1:testnum
    [y,z]=make(p);
    zhat=viterbi(A,B,y);
    cnt=test(zhat,z);
    if cnt~=0
        error=error+cnt;
```

```
    end
end
s=num2str(error/(testnum*0.01));
disp("Error probability is "+s+"%")
```

for문을 활용하여 testnum 만큼 반복한다. make 함수, viterbi 함수를 사용해서 Estimated Sequence zhat을 얻는다. test 함수를 사용해서 cnt를 얻는다. if문을 사용해서 cnt가 0이 아니라면, 즉 cnt=1인 경우 error값이 1 증가하도록 설정한다.

반복문이 종료되면, error/(testnum*0.01)을 string 자료형으로 변경한다. 그 후 disp함수를 활용하여 Error probability가 몇 %인지 출력한다.

3. 결과

결과의 신뢰성을 위해 Error Probability를 계산하는 코드를 다음과 같이 임시로 변경하였다.

```
for j=1:5
    error=0;
    for i=1:testnum
        [y,z]=make(p);
        zhat=viterbi(A,B,y);
        cnt=test(zhat,z);
        if cnt~=0
            error=error+cnt;
        end
    end
    js=num2str(j);
    s=num2str(error/(testnum*0.01));
    disp("Trial "+js+"; Error probability is "+s+"%")
end
```

시행 시 총 5회 분량의 Error Probability가 출력될 것이다. 시행 결과는 다음과 같다.

(1) $P_{avg}(0.01)$

시행 횟수(회)	1	2	3	4	5
결과(%)	12.9	13.1	11.3	14.4	14.1

(2) $P_{avg}(0.05)$

시행 횟수(회)	1	2	3	4	5
결과(%)	38.6	36.3	39.4	38.4	39.4

(3) $P_{avg}(0.1)$

시행 횟수(회)	1	2	3	4	5
결과(%)	55.9	57.8	55.3	58.1	56.3

z(t)의 값이 올바르게 관측될 확률은 p가 아니라 1-p이다. 따라서 p가 커짐에 따라 1-p의 값은 작아지므로, Error Probability가 높게 측정되는 것을 확인할 수 있었다.

4. 부록

Error Probability	시행 결과
$P_{avg}(0.01)$	Trial 1; Error probability is 12.9% Trial 2; Error probability is 13.1% Trial 3; Error probability is 11.3% Trial 4; Error probability is 14.4% Trial 5; Error probability is 14.1% fx >>
$P_{avg}(0.05)$	Trial 1; Error probability is 38.6% Trial 2; Error probability is 36.3% Trial 3; Error probability is 39.4% Trial 4; Error probability is 38.4% Trial 5; Error probability is 39.4% fx >>
$P_{avg}(0.1)$	Trial 1; Error probability is 55.9% Trial 2; Error probability is 57.8% Trial 3; Error probability is 55.3% Trial 4; Error probability is 58.1% Trial 5; Error probability is 56.3% fx >>