Weather Forecasting Using Hidden Markov Model

J Kou

1. Introduction

A Markov chain is the model that tells us the probability of the sequences of random variables' states. However, in many cases the states may be hidden, and we cannot observe them directly. A hidden Markov model allows us to estimate the states of hidden events we are interested in based on the things we could see. For example, assume a person is locked in a room and he can only estimate the weather outside based on whether the people come in carrying an umbrella or not. As shown in figure 1, there are three types of weather, which are sunny, rainy and foggy.

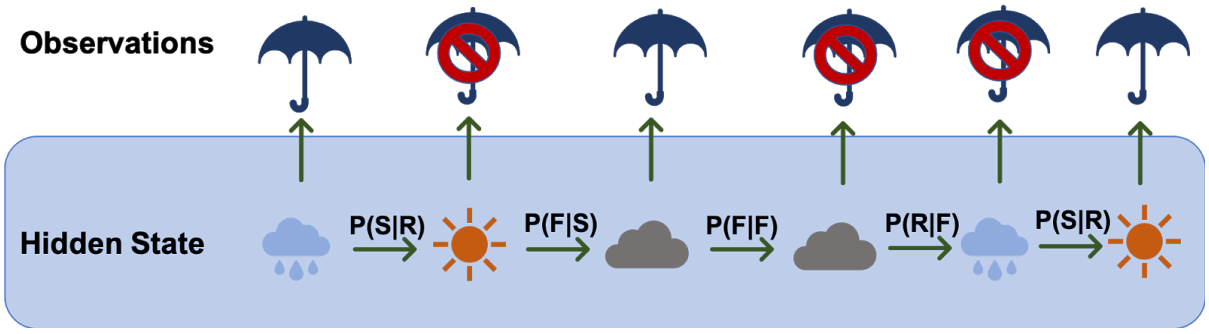

Figure 1. A hidden Markov chain for weather

In order to forecast future's weather states based on the list of observed umbrella status, the expectation-maximization (EM) algorithm was performed to estimate the transition probability which is the probability of moving from a state to the next one (figure 2. black arrow). The status of people taking an umbrella or not can also be predicted by using EM estimated emission probability, that is the probability of an observation being generated from the state of weather (figure 2. green arrow).

The purpose of this project is to estimate the parameters then apply the estimated parameters into the field of forecasting future weather states.

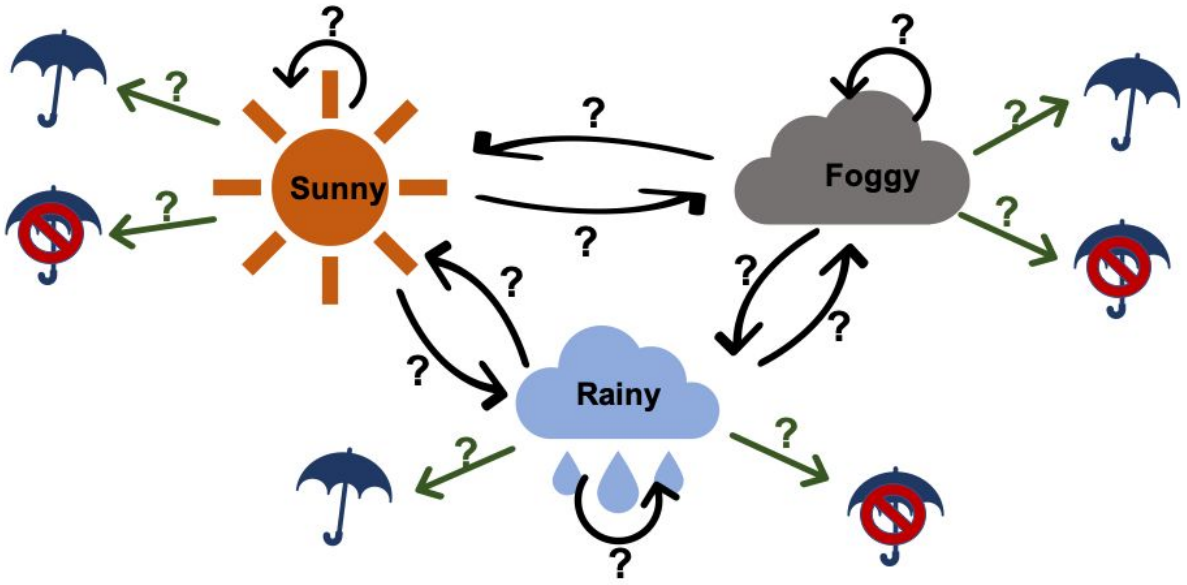

Figure 2. Transition and Emission path of Hidden Markov model

2. Data

The dataset was downloaded from https://inst.eecs.berkeley.edu/~cs188/sp08/projects/hmm/weather-test2-1000.txt, which contains 1000 observations. The first row of data is the actual status of weather which contains three states “sunny”, “rainy” and “foggy”; the second row of data is the status of people taking umbrella where “yes” indicates carrying and “no” means not taking umbrella.

3. Method

3.1 EM Parameter estimation

The parameter was estimated by the following steps: **1)** Initial arbitrary parameters, **2)** use forward-backward method to find most probable path states, **3)** given path states, find maximum likelihood of parameters and **4)** update parameters, go to step 2) until estimation becomes stable.

1) Initial arbitrary parameters

The initial parameter λ_0 were settled as in table1, where the transition probability is in table 1a, the emission probability is in table 1b. And the initial probability π_0 of getting weather sunny, rainy or foggy are equally likely as 1/3.

Table 1a, initial transition probability

		Tomorrow's Weather		
Today's		*Sunny*	*Rainy*	*Foggy*
Weather	*Sunny*	0.50	0.25	0.25
	Rainy	0.25	0.50	0.25
	Foggy	0.25	0.25	0.50

Table 1b, initial emission probability

	No	*Yes*
Sunny	0.7	0.3
Rainy	0.3	0.7
Foggy	0.5	0.5

2) use forward-backward method to find most probable path states

Given current parameter λ_s, the forward algorithm could get the following probabilities:

a. **the probability of *i*th state is sunny, rainy or foggy:**

$$P(O_1,\ldots,O_i, W_i=sunny|\ \lambda_s),\ P(O_1,\ldots,O_i, W_i=rainy|\ \lambda_s),\ P(O_1,\ldots,O_i, W_i=foggy|\ \lambda_s)$$

where

$P(O_1,\ldots,O_i, W_i=sunny|\ \lambda_s)=$

$\{P(O_1,\ldots,O_{i-1}, W_{i-1}=sunny|\ \lambda_s)*P(sunny|\ sunny)+ P(O_1,\ldots,O_{i-1}, W_{i-1}=rainy|\ \lambda_s)*P(sunny|rainy)+$

$P(O_1,\ldots,O_{i-1}, W_{i-1}=foggy|\ \lambda_s)*P(sunny|\ foggy)\}*P(O_i\ |\ sunny)$

----------(1)

Other two can be calculated in a similar way.

b. **probability of getting the sequence of *t* observations:**

$$P(O_1,\ldots,O_t|\ \lambda_s)$$

Which can be calculated by

$$\boldsymbol{P(O_1,\ldots,O_t|\ \lambda_s)} = P(O_1,\ldots,O_t, W_t=sunny|\ \lambda_s) + P(O_1,\ldots,O_t, W_t=rainy|\ \lambda_s) + P(O_1,\ldots,O_t, W_t=foggy|\ \lambda_s)$$

--------*(2)*

In addition to $P(O_1,\ldots,O_t|\ \lambda_s)$, given current state, the backward algorithm can also get the probability:

c. probability of getting next observation:

$$P(O_{i+1},\ldots,O_t \mid W_i=sunny, \lambda_s),\ P(O_{i+1},\ldots,O_t \mid W_i=rainy, \lambda_s),\ P(O_{i+1},\ldots,O_t \mid W_i=foggy, \lambda_s)$$

Where

$$P(O_{i+1},\ldots,O_t \mid W_i=sunny, \lambda_s) =$$
$$P(sunny|\ sunny)*P(O_{i+1}|sunny)*\ P(O_{i+2},\ldots,O_t \mid W_{i+1}=sunny, \lambda_s) +$$
$$P(rainy|sunny)*P(O_{i+1} \mid rainy)*\ P(O_{i+2},\ldots,O_t \mid W_{i+1}=rainy, \lambda_s) +$$
$$P(foggy|sunny)*P(O_{i+1} \mid foggy)*\ P(O_{i+2},\ldots,O_t \mid W_{i+1}=foggy, \lambda_s)$$ ----------*(3)*

d. Probability of state of each step given observation

Given current parameter λ_s and all the observations, the probability of each step as sunny, rainy or foggy can be calculated:

$$P(W_i=sunny|O,\ \lambda_s),\ P(W_i=rainy|O,\ \lambda_s),\ P(W_i=foggy|O,\ \lambda_s)$$

Where

$$P(W_i=sunny|O,\ \lambda_s) = P(O_1,\ldots,O_i, W_i=sunny|\ \lambda_s) * P(O_{i+1},\ldots,O_t \mid W_i=sunny, \lambda_s) / P(O_1,\ldots,O_t|\ \lambda_s)$$

----------*(4)*

Other two can be calculated in similar way.

e. Probability from one state to another given observation

The probability of transition from one state to another in the observed data can be calculated:

$P(W_i=sunny, W_{i+1}=sunny \mid O, \lambda_s)$, $P(W_i=sunny, W_{i+1}=rainy \mid O, \lambda_s)$, $P(W_i=sunny, W_{i+1}=foggy \mid O, \lambda_s)$, $P(W_i=rainy, W_{i+1}=rainy \mid O, \lambda_s)$, $P(W_i=rainy, W_{i+1}=sunny \mid O, \lambda_s)$, $P(W_i=rainy, W_{i+1}=foggy \mid O, \lambda_s)$, $P(W_i=foggy, W_{i+1}=foggy \mid O, \lambda_s)$,$P(W_i=foggy, W_{i+1}=sunny \mid O, \lambda_s)$,$P(W_i=foggy, W_{i+1}=rainy \mid O, \lambda_s)$

Where $P(W_i=sunny, W_{i+1}=sunny \mid O, \lambda_s)=$

$$P(O_1,\ldots,O_i, W_i=sunny \mid \lambda_s)*P(sunny|sunny)*P(O_{i+1}|sunny)*P(Oi+2,\ldots,Ot|Wi+1=sunny, \lambda s)/P(O1,\ldots,Ot|\lambda s)$$

----------(5)

Others can be calculated in similar way.

3) maximum likelihood of parameters

The likelihood of the probability on the log scale was calculated in the following way

Log likelihood=

*n(yes|sunny)*logP(yes|sunny)+n(yes|rainy)*logP(yes|rainy)+n(yes|foggy)*logP(yes|foggy) +*
*n(no|sunny)*logP(no|sunny)+n(no|rainy)*logP(no|rainy)+n(no|foggy)*logP(no|foggy)*
*+n(sunny to sunny)*logP(sunny|sunny)+n(sunny to rainy))*logP(rainy|sunny)+ n(sunny to foggy))*logP(foggy|sunny)*
*+ n(rainy to rainy)*logP(rainy|rainy)+n(rainy to sunny))*logP(sunny|rainy)+ n(rainy to foggy))*logP(foggy|foggy)*
*+ n(foggy to foggy))*logP(foggy|foggy)+ n(foggy to rainy)*logP(rainy|foggy)+ n(foggy to sunny)*logP(sunny|foggy)+constant*----------(6)

We maximize the expectation of log likelihood, and the maximizer at step *s* for *P(yes|sunny)* can be calculated in the following way:

P(yes|sunny)$=E(n(yes|sunny)/(E(n(yes|sunny)+ E(n(no|sunny))$ ----------(7)

Where

$$E(n(yes|sunny)=sum_{Oi=yes}(P(Wi=sunny|O, \lambda_s))/sum(P(Wi=sunny|O, \lambda_s)) \text{ ----------(8)}$$

The maximizer for P(sunny|sunny) can be calculated in the following way:

P(sunny|sunny) =

$$E(n(sunny\ to\ sunny))/(E(n(sunny\ to\ sunny))+ E(n(rainy\ to\ sunny))+ E(n(foggy\ to\ sunny))) \text{ ----------(9)}$$

Where

$$E(n(sunny\ to\ sunny))=$$

$$sum(P(Wi=sunny, Wi+1=sunny))/(P(Wi=sunny, Wi+1=sunny)+ P(Wi=sunny, Wi+1=rainy)+ P(Wi=sunny, Wi+1=foggy)) \text{ ----------(10)}$$

4) *update parameters*

After finding other maximizers in the similar way, we update the λ_s to λ_{s+1}, then do an expectation step again to get new updated λ_{s+1}, stopping when λ becomes stable.

3.2 Prediction accuracy

The prediction accuracy was calculated on the training dataset containing 1000 observations and a new data containing 10 observations using the Viterbi algorithm. The Viterbi method could estimate the state path which gives the largest probability for observed sequence:

$$W^*=argmax\ \{P\ (O, W|\ \lambda)= P(O|W, \lambda)^*P(O|\ \lambda)\} =argmax\ \{P(W|O, \lambda)\} \text{----------(11)}$$

The prediction accuracy was measured by calculating the percentage of correctly classified numbers.

3.3 Weather forecasting

The probability of tomorrow's weather being sunny, rainy or foggy based on previous observation can be calculated:

$$P(W_{t+1}=sunny|O_1,...,O_t, \lambda),\ P(W_{t+1}=rainy|O_1,...,O_t, \lambda),\ P(W_{t+1}=foggy|O_1,...,O_t, \lambda)$$

Where

$$P(W_{t+1}=sunny|\ O_1,\ldots,O_t,\ \lambda)=$$
$$P(O_1,\ldots,O_t, W_t=sunny|\ \lambda_s)*P(sunny|sunny)$$
$$+\ P(O_1,\ldots,O_t, W_t=rainy|\ \lambda_s)*P(sunny|rainy)$$
$$+\ P(O_1,\ldots,O_t, W_t=foggy|\ \lambda_s)*P(sunny|foggy)\ \text{----------(12)}$$

The probability of tomorrow's umbrella status can be calculated:

$$P(O_{t+1}=yes|O_1,\ldots,O_t,\ \lambda),\ P(O_{t+1}=no|O_1,\ldots,O_t,\ \lambda)$$

Where

$$P(O_{t+1}=yes|O_1,\ldots,O_t,\ \lambda)=$$
$$P(W_{t+1}=sunny|\ O_1,\ldots,O_t,\ \lambda)*P(yes|sunny)$$
$$+\ P(W_{t+1}=rainy|\ O_1,\ldots,O_t,\ \lambda)*P(yes|rainy)$$
$$+\ P(W_{t+1}=foggy|\ O_1,\ldots,O_t,\ \lambda)*P(yes|foggy)\ \text{----------(13)}$$

Other probabilities can be forecasted in a similar way.

4. Results

4.1 Estimated Parameter

The EM algorithm was terminated when the difference of current and previous maximum expectation log likelihood is no bigger than 0.00001. In this project the difference is 0.00000988 and the iteration time is 84. The estimated parameters λ were summarized in table 2 and visualized in figure 3.

Table 2a, estimated transition probability

		Tomorrows Weather		
Todays		*Sunny*	*Rainy*	*Foggy*
Weather	*Sunny*	0.754	0.062	0.184

	Rainy	0.109	0.758	0.133
	Foggy	0.418	0.136	0.446

Table 2b, estimated emission probability

	No	*Yes*
Sunny	0.889	0.111
Rainy	0.293	0.707
Foggy	0.713	0.287

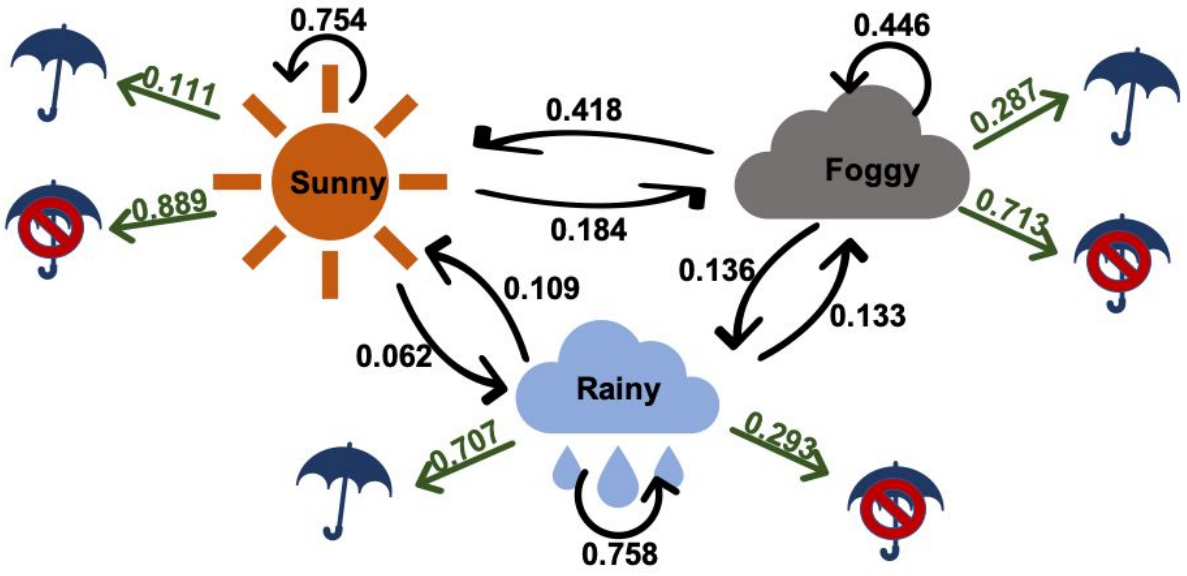

Figure 3. estimated parameter

4.2 prediction accuracy

The accuracy of the Viterbi classifier on the entire dataset is 66.2% and the 50% on the new dataset. The prediction results on new data was summarized in table 3.

Table 3. Prediction accuracy on new observations

days	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*

Observations	yes	no	no	no	no	yes	yes	yes	no	no
Actual state	rainy	foggy	sunny	sunny	sunny	foggy	foggy	rainy	foggy	sunny
Predicted state	foggy	sunny	sunny	sunny	sunny	rainy	rainy	rainy	sunny	sunny

4.3 Weather forecasting

The results of probability of the next ten days being sunny, rainy or foggy based on previous observations are summarized in table 4. As shown in table 4, there is a higher chance of rain for the next two days then sunny for the following eight days.

Table 4. Weather forecasting for next ten days

days	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
P(Wi+1=Sunny)	0.25	0.33	0.38	0.42	0.45	0.47	0.48	0.49	0.50	0.50
P(Wi+1=Rainy)	0.56	0.46	0.40	0.36	0.33	0.31	0.29	0.28	0.27	0.27
P(Wi+1=Foggy)	0.19	0.21	0.22	0.22	0.22	0.22	0.23	0.23	0.23	0.23

The results of probability of next ten days' status of people taking umbrella or not based on previous observations are summarized in table 5. As shown in table 5, the probability of people taking an umbrella is high for the first two days then stabilized around 30% for the following eight days.

Table 5. Umbrella status forecasting for next ten days

days	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
P(Oi+1=no|O)	0.52	0.58	0.61	0.64	0.65	0.67	0.68	0.68	0.68	0.69
P(Oi+1=yes|O)	0.48	0.42	0.39	0.36	0.35	0.33	0.32	0.32	0.32	0.31

5. summary

The EM algorithm is an efficient way to estimate the parameters of a hidden Markov model, however, the accuracy of prediction accuracy based on estimated parameters is not so high. This might because EM can only be able to find a local maximizer of the expected log likelihood instead of the global maximizer.

6. reference

Lecture notes

7. Appendix R code

#Appendix I: Parameter Estimation Using EM Algorithm

#----------1.1. read data----------

```
setwd("~/Desktop")
obs=read.table("weather-test2-1000.txt",sep=",")
o=obs[,2]
o[1:10]
O=as.numeric(o)
```

#----------1.2. initial lambda----------

```
p11=p22=p33=p12=p13=p21=p23=p31=p32=numeric(0)
e11=e12=e21=e22=e31=e32=numeric(0)
p11[1]=0.5;p12[1]=0.25; p13[1]=0.25; p21[1]=0.25; p22[1]=0.5; p23[1]=0.25; p31[1]=0.25;
p32[1]=0.25; p33[1]=0.5
e11[1]=0.7;e12[1]=0.3;e21[1]=0.3;e22[1]=0.7;e31[1]=e32[1]=0.5
P=t(matrix(c(p11,p12,p13,p21,p22,p23,p31,p32,p33),
nrow=3,dimnames=list(c("Sunny","Rainy","Foggy"), c("Sunny","Rainy","Foggy"))))
E=matrix(c(e11,e12,e21,e22,e31,e32), nrow=3,
byrow=T,dimnames=list(c("Sunny","Rainy","Foggy"),c("no","yes")))
P
#         Sunny   Rainy   Foggy
#Sunny   0.50    0.25     0.25
#Rainy   0.25    0.50     0.25
#Foggy   0.25    0.25     0.50
E
#            no    yes
#Sunny    0.7    0.3
#Rainy    0.3    0.7
#Foggy    0.5    0.5
```

#----------1.3. EM Algorithm----------

```
n=length(O)
elog=seq(1:1000)
dif=seq(1:1000)
j=2
while (dif[j-1]>0.00001) {
```

#----------1.3.1 forward table--------

```
forward=matrix(seq(1,3*n),nrow=3,dimnames=list(c("Sunny","Rainy","Foggy"),seq(1,n)))
forward[1,1]=1/3*E[1,O[1]]
forward[2,1]=1/3*E[2,O[1]]
forward[3,1]=1/3*E[3,O[1]]
for (i in 2:n) {
  forward[1,i]=(forward[1,i-1]*P[1,1]+forward[2,i-1]*P[2,1]+forward[3,i-1]*P[3,1])*E[1,O[i]]
  forward[2,i]=(forward[1,i-1]*P[1,2]+forward[2,i-1]*P[2,2]+forward[3,i-1]*P[3,2])*E[2,O[i]]
  forward[3,i]=(forward[1,i-1]*P[1,3]+forward[2,i-1]*P[2,3]+forward[3,i-1]*P[3,3])*E[3,O[i]]
}
```

#----------1.3.2 backward table--------

```
backward=matrix(seq(1,3*n-3),nrow=3,dimnames=list(c("Sunny","Rainy","Foggy"),seq(1,n-1
)))
backward[1,n-1]=P[1,1]*E[1,O[n]]+P[1,2]*E[2,O[n]]+P[1,3]*E[3,O[n]]
backward[2,n-1]=P[2,1]*E[1,O[n]]+P[2,2]*E[2,O[n]]+P[2,3]*E[3,O[n]]
backward[3,n-1]=P[3,1]*E[1,O[n]]+P[3,2]*E[2,O[n]]+P[3,3]*E[3,O[n]]
for (k in (n-2):1){

backward[1,k]=P[1,1]*E[1,O[k+1]]*backward[1,k+1]+P[1,2]*E[2,O[k+1]]*backward[2,k+1]+P[
1,3]*E[3,O[k+1]]*backward[3,k+1]

backward[2,k]=P[2,1]*E[1,O[k+1]]*backward[1,k+1]+P[2,2]*E[2,O[k+1]]*backward[2,k+1]+P[
2,3]*E[3,O[k+1]]*backward[3,k+1]

backward[3,k]=P[3,1]*E[1,O[k+1]]*backward[1,k+1]+P[3,2]*E[2,O[k+1]]*backward[2,k+1]+P[
3,3]*E[3,O[k+1]]*backward[3,k+1]
}
```

#----------1.3.3 P(0|Lambda) --------

```
po=forward[1,n]+forward[2,n]+forward[3,n]
Po=E[1,O[1]]*backward[1,1]*1/3+E[2,O[1]]*backward[2,1]*1/3+E[3,O[1]]*backward[3,1]*1/
3
```

#--1.3.4 p(qi=sunny|o,lambda),p(qi=rainny|o,lambda),p(qi=foggy|o,lambda)-

```
Psrf=matrix(seq(1,3*n),nrow=3,dimnames=list(c("P(qi=S|O)","P(qi=R|O)","P(qi=F|O)"),seq(1
,n)))
for (l in 1:(n-1)){
  Psrf[1,l]=forward[1,l]*backward[1,l]/po
  Psrf[2,l]=forward[2,l]*backward[2,l]/po
  Psrf[3,l]=forward[3,l]*backward[3,l]/po
  Psrf[1,n]=forward[1,n]/po
  Psrf[2,n]=forward[2,n]/po
  Psrf[3,n]=forward[3,n]/po
}
```

#---------- 1.3.5 transition table --------

```
transition=matrix(seq(1,(n-2)*9),nrow=9,dimnames=list(c("Pss","Psr","Psf","Prr","Prs","Prf",
"Pff","Pfs","Pfr"),seq(1,n-2)))
for (t in 1:(n-2)){
  transition[1,t]=forward[1,t]*P[1,1]*E[1,O[t+1]]*backward[1,t+1]/po
  transition[2,t]=forward[1,t]*P[1,2]*E[2,O[t+1]]*backward[2,t+1]/po
  transition[3,t]=forward[1,t]*P[1,3]*E[3,O[t+1]]*backward[3,t+1]/po
  transition[4,t]=forward[2,t]*P[2,2]*E[2,O[t+1]]*backward[2,t+1]/po
  transition[5,t]=forward[2,t]*P[2,1]*E[1,O[t+1]]*backward[1,t+1]/po
  transition[6,t]=forward[2,t]*P[2,3]*E[3,O[t+1]]*backward[3,t+1]/po
  transition[7,t]=forward[3,t]*P[3,3]*E[3,O[t+1]]*backward[3,t+1]/po
  transition[8,t]=forward[3,t]*P[3,1]*E[1,O[t+1]]*backward[1,t+1]/po
  transition[9,t]=forward[3,t]*P[3,2]*E[2,O[t+1]]*backward[2,t+1]/po
}
```

#---------- 1.3.6 maximizer----------

```
p11[j]=sum(transition[1,])/sum(transition[1:3,])
p12[j]=sum(transition[2,])/sum(transition[1:3,])
p13[j]=sum(transition[3,])/sum(transition[1:3,])
p22[j]=sum(transition[4,])/sum(transition[4:6,])
p21[j]=sum(transition[5,])/sum(transition[4:6,])
p23[j]=sum(transition[6,])/sum(transition[4:6,])
p33[j]=sum(transition[7,])/sum(transition[7:9,])
p31[j]=sum(transition[8,])/sum(transition[7:9,])
p32[j]=sum(transition[9,])/sum(transition[7:9,])

e11[j]=sum(Psrf[1,c(which(O==1))])/sum(Psrf[1,])
e12[j]=sum(Psrf[1,c(which(O==2))])/sum(Psrf[1,])
e21[j]=sum(Psrf[2,c(which(O==1))])/sum(Psrf[2,])
e22[j]=sum(Psrf[2,c(which(O==2))])/sum(Psrf[2,])
e31[j]=sum(Psrf[3,c(which(O==1))])/sum(Psrf[3,])
e32[j]=sum(Psrf[3,c(which(O==2))])/sum(Psrf[3,])
```

#---------- 1.3.7 update lambda --------

```
P=t(matrix(c(p11[j],p12[j],p13[j],p21[j],p22[j],p23[j],p31[j],p32[j],p33[j]),
nrow=3,dimnames=list(c("Sunny","Rainy","Foggy"), c("Sunny","Rainy","Foggy"))))

E=matrix(c(e11[j],e12[j],e21[j],e22[j],e31[j],e32[j]), nrow=3,
byrow=T,dimnames=list(c("Sunny","Rainy","Foggy"),c("no","yes")))
```

#---------- 1.3.8 maximized E(log likelihood)--------

```
elog[j]=sum(e11[j]*log(e11[j]),e12[j]*log(e12[j]),e21[j]*log(e21[j]),
        e22[j]*log(e22[j]),e31[j]*log(e31[j]),e32[j]*log(e32[j]),
        p11[j]*log(p11[j]),p12[j]*log(p12[j]),p13[j]*log(p13[j]),
        p21[j]*log(p21[j]),p22[j]*log(p22[j]),p23[j]*log(p23[j]),
        p33[j]*log(p33[j]),p31[j]*log(p31[j]),p32[j]*log(p32[j]))
```

#---------- 1.3.9 check stability --------

```
dif[j]=abs(elog[j]-elog[j-1])
j=j+1
}
```

#---------- 1.3.10 iteration time --------

```
dif[j-1]  #[1] 9.882749e-06
j-1   #[1] 84
```

#---------- 1.3.11 final lambda --------

```
P
#              Sunny       Rainy       Foggy
#Sunny    0.7540976  0.06190326   0.1839992
#Rainy    0.1092816  0.75817739   0.1325410
#Foggy    0.4181348  0.13562309   0.4462422

E
#                no         yes
#Sunny    0.8886417  0.1113583
#Rainy    0.2927066  0.7072934
#Foggy    0.7130629  0.2869371
```

#Appendix II: Prediction Accuracy

#----------2.1. prediction accuracy on overall dataset----------

```
setwd("~/Desktop")
obs=read.table("weather-test2-1000.txt",sep=",")
overall=obs[,1]
overall[1:5]
[1] rainy foggy sunny sunny  sunny
Levels: foggy rainy sunny
```

#---------- log scale lambda ----------

```
P=t(matrix(log(c(0.7540976,0.06190326,0.1839992,0.1092816,0.75817739,0.1325410,0.418
1348,0.13562309,0.4462422)), nrow=3,dimnames=list(c("Sunny","Rainy","Foggy"),
c("Sunny","Rainy","Foggy"))))

E=matrix(log(c(0.8886417,0.1113583,0.2927066,0.7072934,0.7130629,0.2869371)),
nrow=3, byrow=T,dimnames=list(c("Sunny","Rainy","Foggy"),c("no","yes")))

P
#               Sunny       Rainy        Foggy
#Sunny    -0.2822335   -2.7821824   -1.6928239
#Rainy    -2.2138272   -0.2768379   -2.0208632
#Foggy    -0.8719514   -1.9978756   -0.8068934
E
#                no          yes
#Sunny   -0.1180612   -2.1950023
#Rainy   -1.2285845   -0.3463097
#Foggy   -0.3381856   -1.2484923
```

#----------A table ----------

```
A=matrix(data=0, nrow=3, ncol=n-1)
A[1,1]=max(log(1/3)+P[1,1]+E[1,O[1]],
log(1/3)+P[2,1]*E[2,O[1]],log(1/3)+P[3,1]+E[3,O[1]])+E[1,O[2]]
A[2,1]=max(log(1/3)+P[1,2]+E[1,O[1]],
log(1/3)+P[2,2]*E[2,O[1]],log(1/3)+P[3,2]+E[3,O[1]])+E[2,O[2]]
A[3,1]=max(log(1/3)+P[1,3]+E[1,O[1]],
log(1/3)+P[2,3]*E[2,O[1]],log(1/3)+P[3,3]+E[3,O[1]])+E[3,O[2]]
for (i in 2:(n-1)){
  A[1,i]=max(A[1,i-1]+P[1,1], A[2,i-1]+P[2,1], A[3,i-1]+P[3,1])+E[1,O[i+1]]
  A[2,i]=max(A[1,i-1]+P[1,2], A[2,i-1]+P[2,2], A[3,i-1]+P[3,2])+E[2,O[i+1]]
  A[3,i]=max(A[1,i-1]+P[1,3], A[2,i-1]+P[2,3], A[3,i-1]+P[3,3])+E[3,O[i+1]]
}
A[,1:3]
 #           [,1]          [,2]          [,3]
#[1,]  -0.4500036  -0.8502982  -1.250593
#[2,]  -2.2313252  -3.7367476  -4.861065
#[3,]  -0.7369534  -1.8820324  -2.881308
```

#----------q table ----------

```
q=matrix(data=0, nrow=3,ncol=n-1)
for (i in 2:(n-1)){
  q[1,1]=which.max(c(log(1/3)+P[1,1]+E[1,O[1]],
log(1/3)+P[2,1]*E[2,O[1]],log(1/3)+P[3,1]+E[3,O[1]]))
  q[2,1]=which.max(c(log(1/3)+P[1,2]+E[1,O[1]],
log(1/3)+P[2,2]*E[2,O[1]],log(1/3)+P[3,2]+E[3,O[1]]))
  q[3,1]=which.max(c(log(1/3)+P[1,3]+E[1,O[1]],
log(1/3)+P[2,3]*E[2,O[1]],log(1/3)+P[3,3]+E[3,O[1]]))
  q[1,i]=which.max(c(A[1,i-1]+P[1,1], A[2,i-1]+P[2,1], A[3,i-1]+P[3,1]))
  q[2,i]=which.max(c(A[1,i-1]+P[1,2], A[2,i-1]+P[2,2], A[3,i-1]+P[3,2]))
  q[3,i]=which.max(c(A[1,i-1]+P[1,3], A[2,i-1]+P[2,3], A[3,i-1]+P[3,3]))
}
q[,1:10]
#      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#[1,]    2    1    1    1    1    1    1    2    1    1
#[2,]    2    2    1    1    1    2    2    2    2    2
#[3,]    2    3    1    1    1    3    2    2    3    3
```

#----------estimated states ----------

```
Qs=numeric(n)
Qs[n]=which.max(c(A[1,n-1],A[2,n-1],A[3,n-1]))
for (i in (n-1):1){
  if(Qs[i+1]==1) {Qs[i]=q[1,i]}
  if(Qs[i+1]==2) {Qs[i]=q[2,i]}
  if(Qs[i+1]==3) {Qs[i]=q[3,i]}
}
Qstar=character(n)
for (i in 1:n){
  if (Qs[i]==1) {Qstar[i]="sunny"}
  if (Qs[i]==2) {Qstar[i]="rainy"}
  if (Qs[i]==3) {Qstar[i]="foggy"}
}
Qstar[1:10]
#[1] "rainy" "sunny" "sunny" "sunny" "sunny" "rainy" "rainy" "rainy" "sunny" "sunny"
```

#----------accuracy ----------

```
k.2=numeric(n)
for (i in 1:n){
  k.2[i]=(overall[i]==Qstar[i])
}
acc.2=sum(k.2)/10
acc.2 #[1] 66.2
```

#----------2.1. prediction accuracy on new short observation----------

```
O.new=c('yes', 'no', 'no', 'no', 'no', 'yes', 'yes', 'yes', 'no', 'no')
O.new=as.factor(O.new)
O.new=as.numeric(O.new)
actual=c('rainy', 'foggy', 'sunny', 'sunny', 'sunny', 'foggy', 'foggy', 'rainy', 'foggy', 'sunny')
```

#---------- lambda ----------

```
P=t(matrix(c(0.7540976,0.06190326,0.1839992,0.1092816,0.75817739,0.1325410,0.418134
8,0.13562309,0.4462422), nrow=3,dimnames=list(c("Sunny","Rainy","Foggy"),
c("Sunny","Rainy","Foggy"))))

E=matrix(c(0.8886417,0.1113583,0.2927066,0.7072934,0.7130629,0.2869371), nrow=3,
byrow=T,dimnames=list(c("Sunny","Rainy","Foggy"),c("no","yes")))

P
#              Sunny        Rainy         Foggy
#Sunny    0.7540976   0.06190326    0.1839992
#Rainy    0.1092816   0.75817739    0.1325410
#Foggy    0.4181348   0.13562309    0.4462422

E
#                no          yes
#Sunny    0.8886417   0.1113583
#Rainy    0.2927066   0.7072934
#Foggy    0.7130629   0.2869371
```

#----------A table ----------

```
A=matrix(data=0, nrow=3, ncol=9)
A[1,1]=max(1/3*P[1,1]*E[1,O.new[1]],
1/3*P[2,1]*E[2,O.new[1]],1/3*P[3,1]*E[3,O.new[1]])*E[1,O.new[2]]
A[2,1]=max(1/3*P[1,2]*E[1,O.new[1]],
1/3*P[2,2]*E[2,O.new[1]],1/3*P[3,2]*E[3,O.new[1]])*E[2,O.new[2]]
A[3,1]=max(1/3*P[1,3]*E[1,O.new[1]],
1/3*P[2,3]*E[2,O.new[1]],1/3*P[3,3]*E[3,O.new[1]])*E[3,O.new[2]]
for (i in 2:9){
  A[1,i]=max(A[1,i-1]*P[1,1], A[2,i-1]*P[2,1], A[3,i-1]*P[3,1])*E[1,O.new[i+1]]
  A[2,i]=max(A[1,i-1]*P[1,2], A[2,i-1]*P[2,2], A[3,i-1]*P[3,2])*E[2,O.new[i+1]]
  A[3,i]=max(A[1,i-1]*P[1,3], A[2,i-1]*P[2,3], A[3,i-1]*P[3,3])*E[3,O.new[i+1]]
}
A
#[,1]       [,2]        [,3]       [,4]        [,5]        [,6]        [,7]        [,8]        [,9]
#[1,] 0.04110093 0.026889229 0.0175915892 0.011508846 0.0009978161 8.651058e-05
7.500461e-06 1.762990e-05 1.153391e-05
#[2,] 0.02380408 0.004587842 0.0008842303 0.000303312 0.0005580454 3.024682e-04
1.639419e-04 3.159708e-05 6.089813e-06
#[3,] 0.02644941 0.008961413 0.0038263537 0.002503294 0.0005874667 7.139838e-05
1.145621e-05 1.731037e-05 5.864985e-06
```

```
#------------------------q table --------------------
q=matrix(data=0, nrow=3,ncol=9)
for (i in 2:9){
  q[1,1]=which.max(c(1/3*P[1,1]*E[1,O.new[1]], 1/3*P[2,1]*E[2,O.new[1]],
1/3*P[3,1]*E[3,O.new[1]]))
  q[2,1]=which.max(c(1/3*P[1,2]*E[1,O.new[1]], 1/3*P[2,2]*E[2,O.new[1]],
1/3*P[3,2]*E[3,O.new[1]]))
  q[3,1]=which.max(c(1/3*P[1,3]*E[1,O.new[1]], 1/3*P[2,3]*E[2,O.new[1]],
1/3*P[3,3]*E[3,O.new[1]]))
  q[1,i]=which.max(c(A[1,i-1]*P[1,1], A[2,i-1]*P[2,1], A[3,i-1]*P[3,1]))
  q[2,i]=which.max(c(A[1,i-1]*P[1,2], A[2,i-1]*P[2,2], A[3,i-1]*P[3,2]))
  q[3,i]=which.max(c(A[1,i-1]*P[1,3], A[2,i-1]*P[2,3], A[3,i-1]*P[3,3]))
}
q
#      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
#[1,]    3    1    1    1    1    1    1    2    1
#[2,]    2    2    2    1    1    2    2    2    2
#[3,]    3    3    1    1    1    3    2    2    3

#----------------------estimated states --------------------

Qs=numeric(10)
Qs[10]=which.max(c(A[1,9],A[2,9],A[3,9]))
for (i in 9:1){
  if(Qs[i+1]==1) {Qs[i]=q[1,i]}
  if(Qs[i+1]==2) {Qs[i]=q[2,i]}
  if(Qs[i+1]==3) {Qs[i]=q[3,i]}
}
Qs #[1] 3 1 1 1 1 2 2 2 1 1
Qstar=character(10)
for (i in 1:10){
  if (Qs[i]==1) {Qstar[i]="sunny"}
  if (Qs[i]==2) {Qstar[i]="rainy"}
  if (Qs[i]==3) {Qstar[i]="foggy"}
}
Qstar
#[1] "foggy" "sunny" "sunny" "sunny" "sunny" "rainy" "rainy" "rainy" "sunny" "sunny"
#----------------------accuracy --------------------

k=numeric(10)
for (i in 1:10){
  k[i]=(actual[i]==Qstar[i])
}
acc.1=sum(k)/10
acc.1 #[1] 0.5
```

#Appendix III: Weather Forecast for Next 10 Days

#----------3.1. estimated lambda----------

```
setwd("~/Desktop")
obs=read.table("weather-test2-1000.txt",sep=",")
o=obs[,2]
o[1:10]
O=as.numeric(o)

P=t(matrix(c(0.7540976,0.06190326,0.1839992,0.1092816,0.75817739,0.1325410,0.418134
8,0.13562309,0.4462422), nrow=3,dimnames=list(c("Sunny","Rainy","Foggy"),
c("Sunny","Rainy","Foggy"))))

E=matrix(c(0.8886417,0.1113583,0.2927066,0.7072934,0.7130629,0.2869371), nrow=3,
byrow=T,dimnames=list(c("Sunny","Rainy","Foggy"),c("no","yes")))

P #transition matrix
#              Sunny       Rainy       Foggy
#Sunny     0.7540976  0.06190326   0.1839992
#Rainy     0.1092816  0.75817739   0.1325410
#Foggy     0.4181348  0.13562309   0.4462422

E #emission matrix
#                no         yes
#Sunny     0.8886417  0.1113583
#Rainy     0.2927066  0.7072934
#Foggy     0.7130629  0.2869371
```

#----------3.2 forward table--------

```
n=length(O)
forward=matrix(seq(1,3*n),nrow=3,dimnames=list(c("Sunny","Rainy","Foggy"),seq(1,n)))
forward[1,1]=1/3*E[1,O[1]]
forward[2,1]=1/3*E[2,O[1]]
forward[3,1]=1/3*E[3,O[1]]
for (i in 2:n) {
  forward[1,i]=(forward[1,i-1]*P[1,1]+forward[2,i-1]*P[2,1]+forward[3,i-1]*P[3,1])*E[1,O[i]]
  forward[2,i]=(forward[1,i-1]*P[1,2]+forward[2,i-1]*P[2,2]+forward[3,i-1]*P[3,2])*E[2,O[i]]
  forward[3,i]=(forward[1,i-1]*P[1,3]+forward[2,i-1]*P[2,3]+forward[3,i-1]*P[3,3])*E[3,O[i]]
}
forward[,1:3]

#                1           2           3
#Sunny   0.03711943  0.08330944  0.08274024
#Rainy   0.23576447  0.05679119  0.01639889
```

```
#Foggy   0.09564570   0.05758664   0.03462179
```

#----------3.3 backward table--------

```
backward=matrix(seq(1,3*n-3),nrow=3,dimnames=list(c("Sunny","Rainy","Foggy"),seq(1,n-1
)))
backward[1,n-1]=P[1,1]*E[1,O[n]]+P[1,2]*E[2,O[n]]+P[1,3]*E[3,O[n]]
backward[2,n-1]=P[2,1]*E[1,O[n]]+P[2,2]*E[2,O[n]]+P[2,3]*E[3,O[n]]
backward[3,n-1]=P[3,1]*E[1,O[n]]+P[3,2]*E[2,O[n]]+P[3,3]*E[3,O[n]]
for (k in (n-2):1){

backward[1,k]=P[1,1]*E[1,O[k+1]]*backward[1,k+1]+P[1,2]*E[2,O[k+1]]*backward[2,k+1]+P[
1,3]*E[3,O[k+1]]*backward[3,k+1]

backward[2,k]=P[2,1]*E[1,O[k+1]]*backward[1,k+1]+P[2,2]*E[2,O[k+1]]*backward[2,k+1]+P[
2,3]*E[3,O[k+1]]*backward[3,k+1]

backward[3,k]=P[3,1]*E[1,O[k+1]]*backward[1,k+1]+P[3,2]*E[2,O[k+1]]*backward[2,k+1]+P[
3,3]*E[3,O[k+1]]*backward[3,k+1]
}
backward[,1:4]
```

```
 #                1                2                3                4
#Sunny   6.980523e-257   8.818381e-257   1.092718e-256   1.278231e-256
#Rainy   2.394857e-257   3.671708e-257   7.339664e-257   2.105266e-256
#Foggy   5.858813e-257   7.656819e-257    1.038727e-256   1.509113e-256
```

#----------3.4 P(0|Lambda) --------

```
po=forward[1,n]+forward[2,n]+forward[3,n]
po #[1] 1.384106e-257
Po=E[1,O[1]]*backward[1,1]*1/3+E[2,O[1]]*backward[2,1]*1/3+E[3,O[1]]*backward[3,1]*1/
3
Po #[1] 1.384106e-257
```

#--3.5 p(qi=sunny|o,lambda),p(qi=rainny|o,lambda),p(qi=foggy|o,lambda)-

```
#p(qi=s|o,lamda),p(qi=r|o,lamda),p(qi=f|o,lamda)
Psrf=matrix(seq(1,3*n),nrow=3,dimnames=list(c("P(qi=S|O)","P(qi=R|O)","P(qi=F|O)"),seq(1
,n)))
for (l in 1:(n-1)){
  Psrf[1,l]=forward[1,l]*backward[1,l]/po
  Psrf[2,l]=forward[2,l]*backward[2,l]/po
  Psrf[3,l]=forward[3,l]*backward[3,l]/po
  Psrf[1,n]=forward[1,n]/po
  Psrf[2,n]=forward[2,n]/po
  Psrf[3,n]=forward[3,n]/po
```

```
}
Psrf[,n]

#P(qi=S|O) P(qi=R|O) P(qi=F|O)
#0.1251379 0.6929691 0.1818930
```

#----------3.5 predict next ten days weather --------

```
#predict state
predict.state=matrix(seq(1,3*10),nrow=3,dimnames=list(c("P(qi+1=S|O)","P(qi+1=R|O)","P(
qi+1=F|O)"),seq(1,10)))
for(i in 2:10){
  predict.state[1,1]=Psrf[1,n]*P[1,1]+Psrf[2,n]*P[2,1]+Psrf[3,n]*P[3,1]
  predict.state[2,1]=Psrf[1,n]*P[1,2]+Psrf[2,n]*P[2,2]+Psrf[3,n]*P[3,2]
  predict.state[3,1]=Psrf[1,n]*P[1,3]+Psrf[2,n]*P[2,3]+Psrf[3,n]*P[3,3]

predict.state[1,i]=predict.state[1,i-1]*P[1,1]+predict.state[2,i-1]*P[2,1]+predict.state[3,i-1]*
P[3,1]

predict.state[2,i]=predict.state[1,i-1]*P[1,2]+predict.state[2,i-1]*P[2,2]+predict.state[3,i-1]*
P[3,2]

predict.state[3,i]=predict.state[1,i-1]*P[1,3]+predict.state[2,i-1]*P[2,3]+predict.state[3,i-1]*
P[3,3]
}
predict.state
#                1       2       3       4       5       6       7       8       9      10
#P(qi+1=S|O) 0.2461508 0.3285513 0.3849784 0.4237064 0.4503107 0.4685929 0.4811581
0.4897945 0.4957306 0.4998108
#P(qi+1=R|O) 0.5578088 0.4647432 0.4007302 0.3567189 0.3264644 0.3056681 0.2913735
0.2815481 0.2747945 0.2701525
#P(qi+1=F|O) 0.1960404 0.2067056 0.2142915 0.2195748 0.2232251 0.2257392 0.2274686
0.2286577 0.2294752 0.2300371
```

#----------3.6 predict next ten days observation --------

```
predict.obs=matrix(seq(1,2*10),nrow=2,dimnames=list(c("P(Oi+1=no|O)","P(Oi+1=yes|O)"),
seq(1,10)))
for(i in 1:10){
  predict.obs[1,i]=predict.state[1,i]*E[1,1]+predict.state[2,i]*E[2,1]+predict.state[3,i]*E[3,1]
  predict.obs[2,i]=predict.state[1,i]*E[1,2]+predict.state[2,i]*E[2,2]+predict.state[3,i]*E[3,2]
}
predict.obs
#                1       2       3       4       5       6       7       8       9      10
#P(Oi+1=no|O)  0.5218033 0.5753918 0.6122075 0.6375078 0.6548967 0.6668485
0.6750635 0.6807101 0.6845913 0.6872591
#P(Oi+1=yes|O) 0.4781967 0.4246082 0.3877925 0.3624923 0.3451035 0.3331517
0.3249367 0.3192902 0.3154090 0.3127413
```