HMM: forward-backward, viterbi and baum-welch algorithms (with gory details)

Eliezer de Souza da Silva

Thursday 3rd March, 2016

Introduction

Consider a Hidden Markov Model (HMM) with hidden states x_t (for $t \in 1, 2, \cdots, T$), initial probability $p(x_1)$, observed states y_t, transition probability $p(x_t|x_{t-1})$ and observation model $p(y_t|x_t)$. This model can be factorized as

$$p(x_{1:T}, y_{1:T}) = p(y_1|x_1)p(x_1) \prod_{t=2}^{t=T} p(y_t|x_t)p(x_t|x_{t-1})$$

[1] In this document we will present the details of methods to answer the some questions about a HMM. Given a set of observations $y_{1:T}$:

- what is the marginal probability, expected value and variance of x_t (in other words, smoothing distribution $p(x_t|y_{1:T})$)
- What is the sequence of most probable hidden states x_t (the conditional mode of X): $\text{argmax}_{x_{1:T}} p(x_{1:T}|y_{1:T}) = \text{argmax}_{x_{1:T}} p(x_{1:T}, y_{1:T})$.
- Which values of the parameters (of the transition and observation distribution) maximizes the probability of observing the data $y_{1:T}$

Forward-backward algorithm: marginal probability, mean and variance

our starting point is the marginal probability $p(x_t|y_{1:T})$ of x_t given all the observations $y_{1:T}$.

$$\begin{aligned} p(x_t|y_{1:T}) &= \frac{p(x_t, y_{1:T})}{p(y_{1:T})} \\ &= \frac{p(x_t, y_{1:t}, y_{(t+1):T})}{p(y_{1:T})} \\ &= \underbrace{p(y_{(t+1):T}|x_t)}_{\beta_t(x_t)} \underbrace{p(x_t, y_{1:t})}_{\alpha_t(x_t)} \frac{1}{p(y_{1:T})} \\ &= \frac{\alpha_t(x_t)\beta_t(x_t)}{p(y_{1:T})} \end{aligned}$$

This marginal can be computed by using dynamic programming to calculate the values of $\beta_t(x_t)$ and

[1] We will use the notation $X = x_{1:T}$ to represent the set $X = \{x_1, x_2, \cdots, x_T\}$.

$\alpha_t(x_t)$. Starting with $\alpha_t(x_t)$ we have

$$\begin{aligned}
\alpha_t(x_t) &= p(x_t, y_{1:t}) \\
&= \sum_{x_{t-1}} p(x_t, x_{t-1}, y_{1:t}) \\
&= \sum_{x_{t-1}} p(x_t, x_{t-1}, y_{1:(t-1)}, y_t) \\
&= \sum_{x_{t-1}} p(y_t|x_t, \cancel{x_{t-1}, y_{1:(t-1)}}) p(x_t|x_{t-1}, \cancel{y_{1:(t-1)}}) \underbrace{p(x_{t-1}, y_{1:(t-1)})}_{\alpha_{t-1}(x_{t-1})} \\
\Rightarrow \quad \underbrace{\alpha_t(x_t)}_{\texttt{alpha[k,t]},\ \mathrm{k}=x_t} &= \underbrace{p(y_t|x_t)}_{\texttt{fyx}} \sum_{x_{t-1}} \underbrace{p(x_t|x_{t-1})}_{\texttt{fxx}} \underbrace{\alpha_{t-1}(x_{t-1})}_{\texttt{alpha[ks,t-1]},\ \mathrm{ks}=x_{t-1}}
\end{aligned}$$

For $\beta_t(x_t)$ we have:

$$\begin{aligned}
\beta_t(x_t) &= p(y_{(t+1):T}|x_t) \\
&= \sum_{x_{t+1}} \frac{p(y_{t+1}, y_{(t+2):T}, x_t, x_{t+1})}{p(x_t)} \\
&= \sum_{x_{t+1}} \frac{p(y_{(t+2):T}|y_{t+1}, x_t, x_{t+1}) p(y_{t+1}|x_{t+1}, x_t) p(x_{t+1}|x_t) p(x_t)}{p(x_t)} \\
&= \sum_{x_{t+1}} \frac{p(y_{(t+2):T}|\cancel{y_{t+1}, x_t}, x_{t+1}) p(y_{t+1}|x_{t+1}, \cancel{x_t}) p(x_{t+1}|x_t) \cancel{p(x_t)}}{\cancel{p(x_t)}} \\
&= \sum_{x_{t+1}} \underbrace{p(y_{(t+2):T}|x_{t+1})}_{\beta_{t+1}(x_{t+1})} p(y_{t+1}|x_{t+1}) p(x_{t+1}|x_t) \\
\Rightarrow \quad \underbrace{\beta_t(x_t)}_{\texttt{beta[k,t]},\ \mathrm{k}=x_t} &= \sum_{x_{t+1}} \underbrace{\beta_{t+1}(x_{t+1})}_{\texttt{beta[ks,t+1]},\ \mathrm{ks}=x_{t+1}} \underbrace{p(y_{t+1}|x_{t+1})}_{\texttt{fyx}} \underbrace{p(x_{t+1}|x_t)}_{\texttt{fxx}}
\end{aligned}$$

Considering a chain of length T and K number of states, both this recursive calculations can be done in $O(TK^2)$ by storing the intermediate results in two vectors `alpha` and `beta` using $O(TK)$ space. The marginal probability of x_t given the observations $y_{1:T}$ can be calculated in $O(TK)$.

```
alpha_beta=function(y,theta,fyx){
  #kx is the number of states for x
  t = length(y)
  kx = length(theta$states)
  alpha = matrix(rep(1,kx*t),nrow = kx,ncol =t)
  beta = alpha

  alpha[,1]=theta$px0
  states=theta$states
  # initial values for alpha and beta
  for(k in 1:kx){
    alpha[k,1]=fyx(y[1],states[k],theta$pary)*(theta$px0)[k]
    beta[k,t]=1
  }

  alpha[,1]=log(alpha[,1])
  beta[,t]=log(beta[,t])


  for(i in 2:t){
    for(k in 1:kx){
      logsum=-Inf
```

```
      for(previousState in 1:kx){
        tvar = (log(fxx(states[k],states[previousState],theta$p))
                + alpha[previousState,i-1])
        if (tvar > -Inf) {
          logsum = tvar+log(1 + exp(logsum - tvar))
          ## avoid numerical problem with product of probabilities
        }
      }
      alpha[k,i]=log(fyx(y[i],states[k],theta$pary))+logsum
      #p(x_i,y_{1:i}) = alpha[,i]
    }
  }
  for(i in (t-1):1){
    for(k in 1:kx){
      logsum=-Inf
      for(forwardState in 1:kx){
        tvar = (log(fxx(states[forwardState],states[k],theta$p))
                +log(fyx(y[i+1],states[forwardState],theta$pary))
                +beta[forwardState,i+1])
        if (tvar > -Inf) {
          logsum = tvar+log(1 + exp(logsum - tvar))
          ## avoid numerical problem with product of probabilities
        }
      }
      beta[k,i]=logsum
    }
  }
  return(list(alpha=alpha,beta=beta))
}

backward_forward=function(y,theta,fyx){
  #kx is the number of states for x
  t = length(y)
  kx = length(theta$states)

  first_moment=rep(1,t)
  variance=rep(1,t)

  states=theta$states

  res=alpha_beta(y,theta,fyx)
  alpha=res$alpha
  beta=res$beta
  probk = beta
  marg_y=0
  for(i in 1:t){
    # p(x_i|y_{1:T}) = alpha(x_i)*beta(x_i)/p(y_{1:T})
    # marginal x_i given observations
    for(currentState in 1:kx){
      probk[currentState,i]=exp(beta[currentState,i]+alpha[currentState,i])
    }
    # p(y_{1:T}) = sum_{x_i} alpha(x_i)*beta(x_i)
    marg_y = sum(probk[,i])
    probk[,i]=probk[,i]/marg_y

    temp=0
    temp2=0
```

```
    for(currentState in 1:kx){
      temp = temp+states[currentState]*probk[currentState,i]
      temp2 = temp2+(states[currentState]^2)*probk[currentState,i]
    }
    first_moment[i]=temp
    variance[i]=temp2-first_moment[i]^2
  }
  return(list(forward=alpha,log_marginal_y=log(marg_y),
              marginal_x=probk,backward=beta,
              expected=first_moment,
              variance=variance))
}
```

Viterbi algorithm: posterior mode $p(x_{1:T}|y_{1:T})$

We are looking for an algorithm to compute $\text{argmax}_x\, p(x, y|\theta)$ for given values of y_t. To do so we first must see the recursive structure of this computation

$$\underset{x_{1:T}}{\text{argmax}}\, p(x, y|\theta) = \underset{x_{1:T}}{\text{argmax}}\, p(y_1|x_1)p(x_1)\prod_{t=2}^{T} p(x_t|x_{t-1})p(y_t|x_t)$$

$$= \underset{x_1}{\text{argmax}}\, p(y_1|x_1)p(x_1)\, \underset{x_2}{\text{argmax}}\, p(x_2|x_1)p(y_2|x_2)\cdots \underset{x_T}{\text{argmax}}\, p(x_T|x_{T-1})p(y_T|x_T)$$

Notice how, by the structure of the original problem, reflected in the factorized model, we can compute the optimal value at each t by relying on the results from $t-1$. Now, we can use dynamic programming to store the optimal solution at intermediate steps of the computation. Finally we can backtrack through all the stored value to find the global solution.

$$\gamma_t(x_t) = p(y_t|x_t)\max_{x_{t-1}}\{p(x_t|x_{t-1})\gamma_{t-1}(x_{t-1})\}$$

Such that

$$\underset{x_1\cdots x_t}{\text{argmax}}\, p(x_{1:t}, y_{1:T}) = \{\underset{x_t}{\text{argmax}}\, \gamma_t(x_t), \text{ for } t \in \{1, \cdots, T\}\}$$

Our implementation `posterior_mode` of the posterior mode receives as parameter the vetor of observations $y_{1:T}$, the parameters of the model and a parametrized function for the pdf ($p(y_t|x_t)$). Instead of working with probabilities and multiplications, we work with log-probabilities and sums to avoid numerical problems.

```
posterior_mode = function(y,theta,fyx){
  t = length(y)

  kx = length(theta$states)
  states=theta$states
  alpha = matrix(rep(1,kx*t),nrow = kx,ncol =t)


  for(k in 1:kx){
    alpha[k,1]=log(fyx(y[1],states[k],theta$pary))+log(theta$px[k])
  }
  #print(alpha[,1])
  #print(log(theta$px0)+log(c(fyx(y[1],0,theta$pary),fyx(y[1],1,theta$pary))))

  back_path = alpha
  back_path[,1]=1:kx

  for(i in 2:t){
    sum=1:2
```

```
    for(k in 1:2){

      for(previousState in 1:2){
        sum[previousState]=alpha[previousState,i-1]
        +log(fxx(states[k],states[previousState],theta$p))
      }
      kmax = which.max(sum)
      alpha[k,i]=sum[kmax]+log(fyx(y[i],states[k],theta$pary))
      back_path[k,i]=kmax
    }
  }
  best_path=1:t
  for(i in 1:t){
    best_path[i]=states[which.max(alpha[,i])]
  }
  return(list(sum_max=alpha,xmax=best_path,all_path=back_path))
}
```

Baum-Welch algorithm: Expectation-Maximization to compute the mode of parameter p given $y_{1:T}$

We will consider a HMM with stationary transition probability governed by a parameter p (or more generally a matrix with elements p_{ij}). We want to compute the configuration of p that maximize the observed data. If we were to use a MAP or ML approach it would be necessary to marginalize the state variables in the complete likelihood $p(x_{1:T}, y_{1:T}|p)$. However this is not feasible in this model, resulting in non analytical formulas. An alternative approach is to use Expectation Maximization:

- M-step:
$$\theta^{\text{new}} \leftarrow \underset{\theta}{\text{argmax}} \sum_{x_{1:T}} p(x_{1:T}|y_{1:T}|\theta) \log p(x_{1:T}, y_{1:T}|\theta^{\text{old}})$$
- E-step: use θ^{new} to update $p(x_{1:T}|y_{1:T})$ (backward-foward algorithm)

Starting with the log-likelihood we have

$$\log p(x_{1:T}, y_{1:T}|\theta) = \log p(x_1) + \log p(y_1|x_1) + \sum_{t=1}^{t=T} \log p(x_t|x_{t-1}) + \sum_{t=1}^{t=T} \log p(y_t|x_t)$$

So, the expectation over this marginal of $x_{1:T}$ given $y_{1:T}$ is

$$\begin{aligned} Q(\theta) = &\sum_{x_1} p(x_1|y_{1:T}, p_0)(\log p(x_1) + \log p(y_1|x_1, q)) \\ &+ \sum_{t=2}^{t=T} \sum_{x_t, x_{t-1}} p(x_t, x_{t-1}|y_{1:T}) \log p(x_t|x_{t-1}, p) \\ &+ \sum_{t=2}^{t=T} p(x_t|y_{1:T}) \log p(y_t|x_t, q) \end{aligned}$$

Since we want to maximize $Q(\theta)$ in relation to p, all other terms not depending directly on p can be dropped.

$$
\begin{aligned}
Q(\theta) &= \sum_{x_1} p(x_1|y_{1:T}, p_0)(\cancel{\log p(x_1)} + \log p(y_1|x_1, q)) \\
&\quad + \sum_{t=2}^{t=T} \sum_{x_t, x_{t-1}} p(x_t, x_{t-1}|y_{1:T}) \log p(x_t|x_{t-1}, p) \\
&\quad + \sum_{t=2}^{t=T} \cancel{p(x_t|y_{1:T}) \log p(y_t|x_t, q)} \\
Q(\theta) &= \sum_{t=2}^{t=T} \sum_{x_t, x_{t-1}} p(x_t, x_{t-1}|y_{1:T}) \log p(x_t|x_{t-1}, p)
\end{aligned}
$$

Now adding the lagrange multiplier to the constraint with the constraint "$\sum_{x_t} p(x_t|x_{t-1}) = 1$" we obtain

$$
\begin{aligned}
Q(\theta, \lambda) &= Q(\theta) - \sum_{x_{t-1}} \lambda(x_{t-1})(\sum_{x_t} p(x_t|x_{t-1}) - 1) \\
\Rightarrow \frac{\partial Q(\theta, \lambda)}{\partial \lambda(x_{t-1})} &= \sum_{x_t} p(x_t|x_{t-1}) - 1 \\
\Rightarrow \frac{\partial Q(\theta, \lambda)}{\partial p(x_t|x_{t-1})} &= \frac{1}{p(x_t|x_{t-1})} \sum_{t=2}^{t=T} p(x_t, x_{t-1}|y_{1:T}) - \lambda(x_{t-1})
\end{aligned}
$$

If we make $\nabla Q(\theta, \lambda) = 0$ we can find the optimal value for the transition matrix $p(x_t|x_{t-1})$[2]

[2]Notice that, if x_t can be in K different states, $p(x_t|x_{t-1})$ is a $K \times K$ matrix. Also, in our case we are considering also that this transition matrix is the same for all x_t, so this process has an *stationary transition probabilities*

$$\frac{\partial Q(\theta,\lambda)}{\partial \lambda(x_{t-1})} = 0$$
$$\Rightarrow \sum_{x_t} p(x_t|x_{t-1}) = 1$$
$$\frac{\partial Q(\theta,\lambda)}{\partial p(x_t|x_{t-1})} = 0$$
$$\Rightarrow \sum_{t=2}^{t=T} p(x_t, x_{t-1}|y_{1:T}) = \lambda(x_{t-1}) p(x_t|x_{t-1})$$
$$\Rightarrow \sum_{x_t}\sum_{t=2}^{t=T} p(x_t, x_{t-1}|y_{1:T}) = \lambda(x_{t-1}) \underbrace{\sum_{x_t} p(x_t|x_{t-1})}_{=1}$$
$$\Rightarrow \sum_{x_t}\sum_{t=2}^{t=T} p(x_t, x_{t-1}|y_{1:T}) = \lambda(x_{t-1})$$
$$\Rightarrow p(x_t|x_{t-1}) \sum_{x_t}\sum_{t=2}^{t=T} p(x_t, x_{t-1}|y_{1:T}) = \frac{\underbrace{p(x_t|x_{t-1})\lambda(x_{t-1})}}{\sum_{t=2}^{t=T} p(x_t, x_{t-1}|y_{1:T})}$$
$$\Rightarrow p_{\max}(x_t|x_{t-1}) = \frac{\sum_{t=2}^{t=T} p(x_t, x_{t-1}|y_{1:T})}{\sum_{x_t}\sum_{t=2}^{t=T} p(x_t, x_{t-1}|y_{1:T})}$$
$$\Rightarrow p^{\max} = \frac{\sum_{t=2}^{t=T} p(x_t = 0, x_{t-1} = 0|y_{1:T})}{\sum_{x_t}\sum_{t=2}^{t=T} p(x_t, x_{t-1} = 0|y_{1:T})}$$
$$\Rightarrow p_{ij}^{\max} = \frac{\sum_{t=2}^{t=T} p(x_t = i, x_{t-1} = j|y_{1:T})}{\sum_{x_t}\sum_{t=2}^{t=T} p(x_t, x_{t-1} = j|y_{1:T})}$$

Now we need to discover a way to efficiently compute $p(x_t, x_{t-1}|y_{1:T})$:

$$\begin{aligned}
p(x_t, x_{t-1}|y_{1:T}) &= \frac{p(x_t, x_{t-1}, y_{1:(t-1)}, y_t, y_{(t+1):T})}{p(y_{1:T})} \\
&= \frac{p(y_{(t+1):T}|x_t, \cancel{x_{t-1}, y_{1:(t-1)}, y_t}) p(x_t, x_{t-1}, y_{1:(t-1)}, y_t)}{p(y_{1:T})} \\
&= \underbrace{p(y_{(t+1):T}|x_t)}_{\beta_t(x_t)} p(y_t|x_t, \cancel{x_{t-1}, y_{1:(t-1)}}) p(x_t, x_{t-1}, y_{1:(t-1)}) \frac{1}{p(y_{1:T})} \\
&= \beta_t(x_t) p(y_t|x_t) p(x_t|x_{t-1}, \cancel{y_{1:(t-1)}}) \underbrace{p(x_{t-1}, y_{1:(t-1)})}_{\alpha_{t-1}(x_{t-1})} \frac{1}{p(y_{1:T})} \\
\Rightarrow p(x_t, x_{t-1}|y_{1:T}) &= \frac{\beta_t(x_t) p(y_t|x_t) p(x_t|x_{t-1}) \alpha_{t-1}(x_{t-1})}{p(y_{1:T})}
\end{aligned}$$

So again we can use the backward-forward algorithm to calculate the value of $p(x_t, x_{t-1}|y_{1:T})$ allowing us to estimate the mode of p

```
expectation_maximization=function(y,theta,fyx){
  #kx is the number of states for x
  t = length(y)
  kx = length(theta$states)
  px1x2 = x=array(0,c(kx,kx,t))
  alpha = matrix(rep(1,kx*t),nrow = kx,ncol =t)
```

```
  beta = alpha

  states=theta$states


  p_old=runif(3,0.3,0.8)
  p_old[2]=p_old[1]-0.1
  p_old[3]=p_old[1]+0.1
  marg_y=0
  p_new=0
  while(abs(mean(p_old[1:2])-mean(p_old[2:3]))>0.001){
    p_new = p_old[3]
    # Expectation
    theta$p=p_new
    res=alpha_beta(y,theta,fyx)
    alpha=res$alpha
    beta=res$beta
    marg_y=0

    #maximization
    for(i in 2:t){
      # p(x_i|y_{1:T}) = beta(x_i)*p(y_i|x_i)*p(x_i|x_{i-1})*alpha(x_{i-1})/p(y_{1:T})
      # marginal x_i given observations
      for(currentState in 1:kx){
        for(previousState in 1:kx){
          px1x2[currentState,previousState,i]=(exp(beta[currentState,i]
                              +alpha[previousState,i-1])
                              *(fxx(states[currentState],states[previousState],p_new)
                              *fyx(y[i],states[currentState],theta$pary)))

        }
      }
      marg_y = sum(px1x2[,,i])
      if(marg_y!=0)
        px1x2[,,i]=px1x2[,,i]/marg_y
    }

    p_old[1:2]=p_old[2:3]
    p_old[3] = sum(px1x2[1,1,2:(dim(px1x2)[3])])/sum(px1x2[1,,2:(dim(px1x2)[3])])
    #print(p_old)
    if(is.nan(p_old[3]))
    {
      p_old[3]=p_old[2]
    }


  }
  return(list(forward=alpha,log_marginal_y=log(marg_y),
              marginal_x1x2=px1x2,backward=beta,p=p_old[3]))
}
```

Loss function and optimal bayes estimator (OBE)

Consider the loss function that counts the number of errors

$$L(z,x) = \sum_s 1[z_s \neq x_s]$$

We will calculate is optimal bayes estimator by computing

$$z^* = \underset{z}{\operatorname{argmin}}\, E_{x|y} L(z, x)$$

First we will simplify the expression for $E_{x|y}L(z,x)$,

$$E_{x|y}L(z,x) = \sum_{x_{1:T}} p(x_{1:T}|y_{1:T}) \sum_s 1[z_s \neq x_s]$$

$$\Rightarrow E_{x|y}L(z,x) = \sum_s \sum_{x_{1:T}} p(x_{1:T}|y_{1:T}) 1[z_s \neq x_s]$$

Consider now the inner sum,

$$\sum_{x_{1:T}} p(x_{1:T}|y_{1:T}) 1[z_s \neq x_s] = \sum_{x_1} \cdots \sum_{x_s} \cdots \sum_{x_T} p(x_{1:T}|y_{1:T}) 1[z_s \neq x_s]$$

$$\Rightarrow \sum_{x_{1:T}} p(x_{1:T}|y_{1:T}) 1[z_s \neq x_s] = \sum_{x_s} 1[z_s \neq x_s] \sum_{\{x_1 \cdots x_T\}/\{x_s\}} p(x_{1:T}|y_{1:T})$$

Marginalize all the variables x_i $(1 \geq i \leq T)$ different than x_s,

$$\sum_{\{x_1 \cdots x_T\}/\{x_s\}} p(x_{1:T}|y_{1:T}) = \sum_{\{x_1 \cdots x_T\}/\{x_s\}} \frac{p(x_{1:s-1}, x_s, x_{s+1:T}, y_{1:T})}{p(y_{1:T})}$$

$$\sum_{\{x_1 \cdots x_T\}/\{x_s\}} p(x_{1:T}|y_{1:T}) = p(x_s|y_{1:T})$$

Taking all this partial results in consideration we can compute the expected value conditioned on the data y

$$\Rightarrow \sum_{x_{1:T}} p(x_{1:T}|y_{1:T}) 1[z_s \neq x_s] = \sum_{x_s} 1[z_s \neq x_s] p(x_s|y_{1:T})$$

$$\Rightarrow E_{x|y}L(z,x) = \sum_s \sum_{x_s} 1[z_s \neq x_s] p(x_s|y_{1:T})$$

$$\Rightarrow \underset{z}{\operatorname{argmin}}\, E_{x|y}L(z,x) = \underset{z_1, \cdots, z_T}{\operatorname{argmin}} \sum_{s=1}^{T} \sum_{x_s} 1[z_s \neq x_s] p(x_s|y_{1:T})$$

In order to minimize this sum $\sum_{x_s} 1[z_s \neq x_s] p(x_s|y_{1:T})$ we must make sure that when z_s is equal to x_s, $p(x_s|y_{1:T})$ has the maximum value, so

$$z_s = \underset{x_s}{\operatorname{argmax}}\, p(x_s|y_{1:T})$$

So the mode of the smoothing distribution for each hidden variable in the sequence $x_{1:T}$ is the best choice of value if we want to minimize the expected number of errors.

Example model

We will use a simple transtion model and two observations model (gaussian and discrete) to generate some value of y_t and apply the algorithms.

$$p(x_{t+1}|x_t) = p^{[x_{t+1}=x_t]}(1-p)^{[x_{t+1}\neq x_t]}$$

and

$$p(y_t|x_t) = \mathcal{N}(y_t|x_t, \sigma^2)$$

or

$$p(y_t|x_t) = q^{[y_t=x_t]}(1-q)^{[y_t=1-x_t]}$$

We will assume that the states for x_t can be 0 or 1.

```
# generative process of x_t
# init_p is the initial value of x_t
# vals are the states of x_t (in this case c(0,1))
xdyn <- function(p,t,init_p,vals){
  x <- rep(0,t)
  rp<-rbinom(t,1,p)
  rpinit <- rbinom(1,1,init_p)
  x[0]=rpinit*vals[1]+(1-rpinit)*vals[2]
  for(i in 2:t){
    if(x[i-1]==vals[1])
      x[i]=rp[i]*x[i-1]+(1-rp[i])*vals[2]
    if(x[i-1]==vals[2])
      x[i]=rp[i]*x[i-1]+(1-rp[i])*vals[1]
  }
  return(x)
}
```

After sampling value for x_t we are able to sample values for y_t using a discrete and gaussian observation model. Function `r_obs1f` implements the sampling using the discrete observation model and `r_obs2f` implements the gaussian observation model.

```
r_obs1f<-function(x,q){
  r<-rbinom(length(x),1,q)
  return(r*x+(1-r)*(1-x))
}
r_obs2f<-function(x,sd){
  r<-rnorm(length(x),mean=0,sd)
  return(x+r)
}
```

Also we implement different functions for the pdf $p(x_{t+1}|x_t)$ and $p(y_t|x_t)$

```
fxx = function(x2,x1,p){ #p(x2|x1,p)
  return(p*(x1==x2)+(1-p)*(x1!=x2))
}
f1yx = function(y,x,q){ #p(y|x,q) is a discrete distribution
  return(q*(y==x)+(1-q)*(y==1-x))
}
f2yx = function(y,x,sd){ #p(y|x,sigma) is a normal
  return(dnorm(y,mean=x,sd=sd))
}
```

```
test_run = function(p,q,n){
  x = xdyn(p,n,init_p=1,c(0,1))
  y = r_obs1f(x,q)
  y2 = r_obs2f(x,q)

  calcs=backward_forward(
    y,list(px0=c(p,1-p),pary=q,p=p,states=c(0,1)),f1yx)

  post=posterior_mode(
    y,list(px0=c(p,1-p),pary=q,p=p,states=c(0,1)),f1yx)

  calcs2=backward_forward(
    y2,list(px0=c(p,1-p),pary=q,p=p,states=c(0,1)),f2yx)

  post2=posterior_mode(
```

```
    y2,list(px0=c(p,1-p),pary=q,p=p,states=c(0,1)),f2yx)
  return(list(x=x,
              y_discrete=y,
              y_gauss=y2,
              backforward_discrete=calcs,
              backforward_gauss=calcs2,
              posterior_discrete=post,
              posterior_gaus=post2))
}
res=test_run(0.9,0.8,100)
```

Results using the gaussian distribution: graphs with marginal posterior mode, expected value and variance

```
plot(res$y_gauss,type='b',col='red')
lines(res$x,type='s',col='magenta')
lines(res$posterior_gaus$xmax,type='o',col='blue')
```

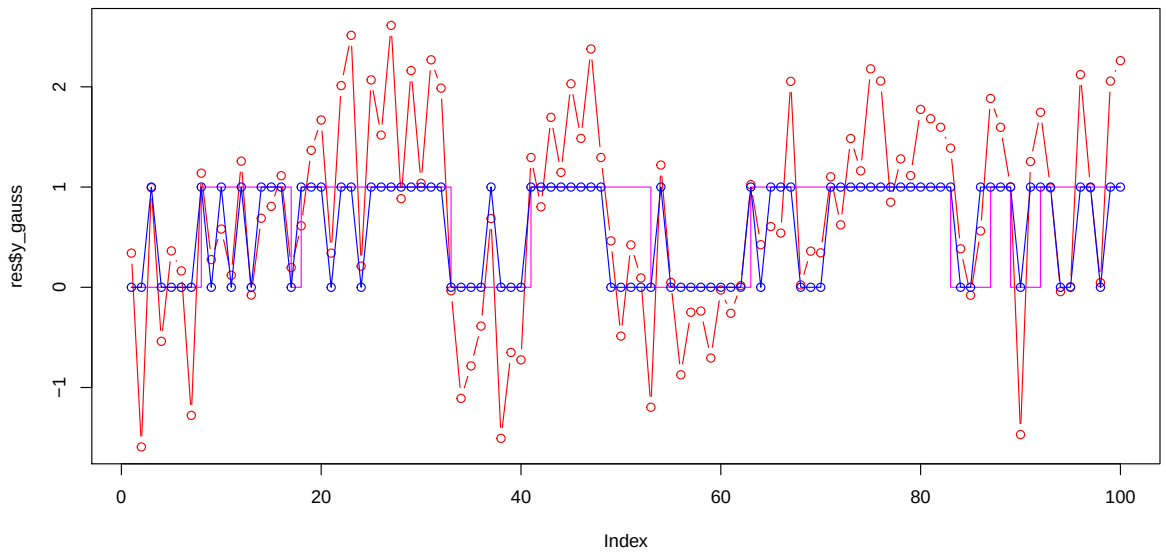

Figure 1: The red line is the y_t values, the magenta line is the original x_t, and the blue line is the posterior mode of $p(x_t|y_{1:T})$, with gaussian observation and $s = 0.2$ and $p = 0.9$

```
plot(res$x,type='o',col='magenta')
lines(res$backforward_gauss$expected,type='o',col='blue')
```

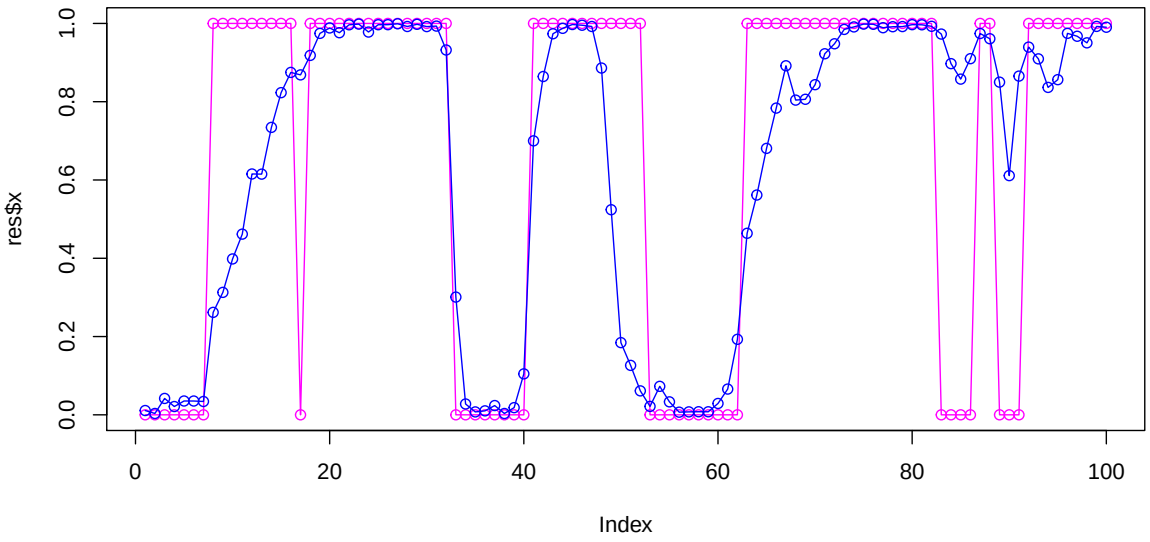

Figure 2: expected value (blue), variance (red) and original value of $x_{1:T}$ (magenta) for the gaussian observation and $s = 0.2$ and $p = 0.9$

```
plot(res$backforward_gauss$variance,type='o',col='red')
```

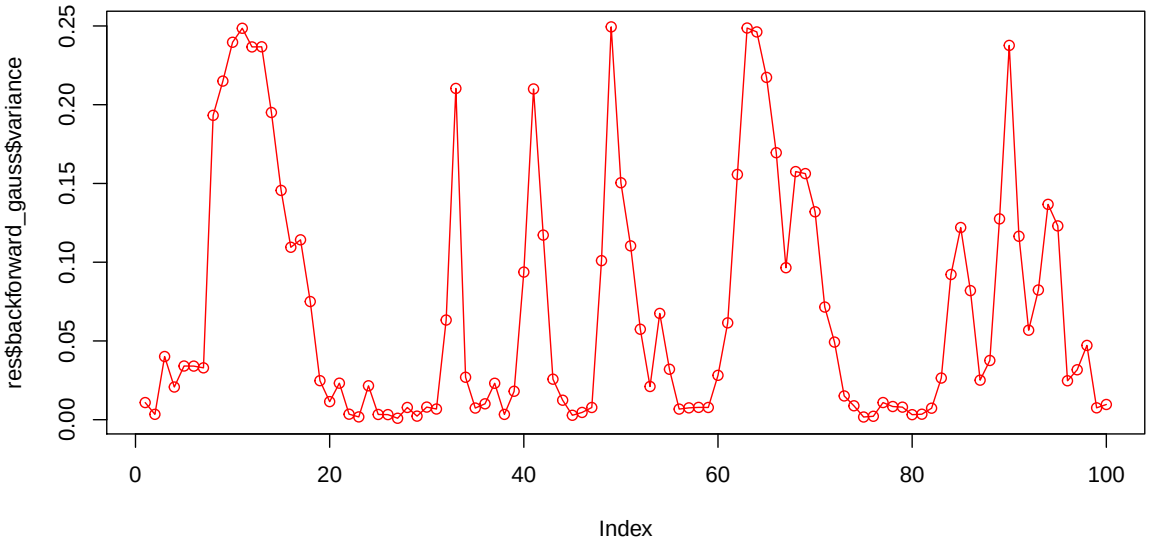

Figure 3: variance (red) of $x_{1:T}$ forthe gaussian observation and $s = 0.2$ and $p = 0.9$

Results using the discrete distribution Posterior mode using the discrete observation distribution with $q = 0.4$ and transition probability $p = 0.6$ (the red line is the y_t values, the magenta line is the original x_t, and the blue line is the posterior mode of $p(x_t|y_{1:T})$)

```
res=test_run(0.9,0.2,100)
```

```
plot(res$y_discrete,type='o',col='red')
```

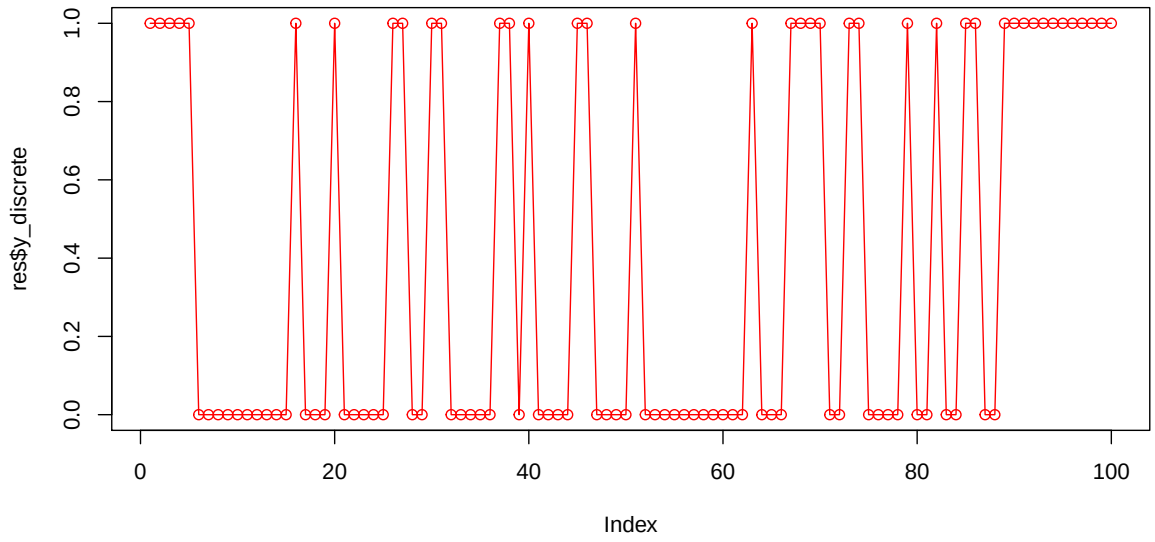

Figure 4: $y_{1:T}$ for the discrete distribution with q=0.2 and p=0.9

```
plot(res$x,type='o',col='magenta')
```

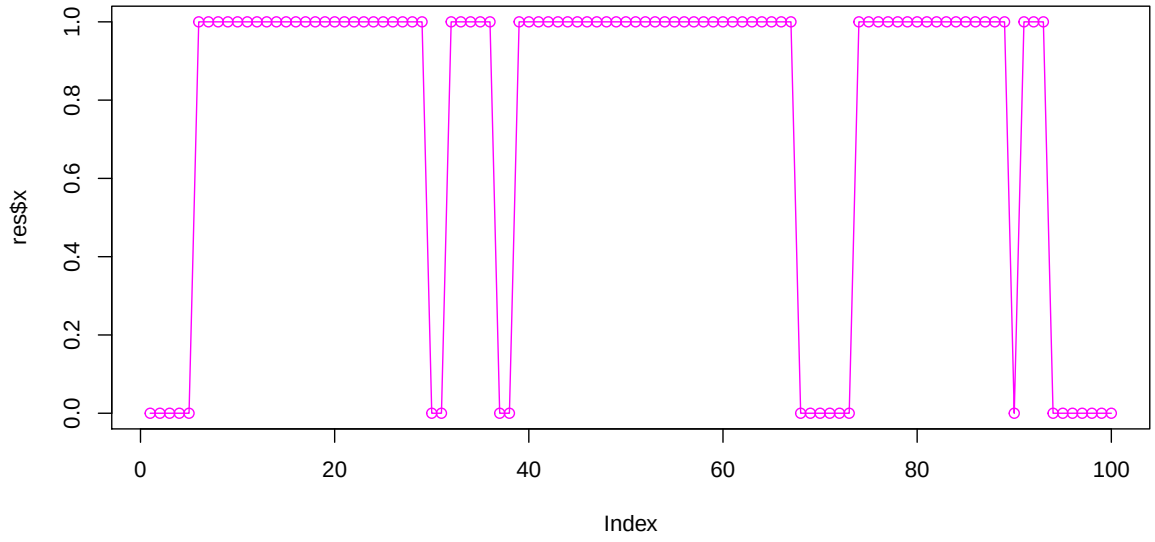

Figure 5: original value (magenta) of $x_{1:T}$ for the discrete distribution with q=0.2 and p=0.9

```
plot(res$x,type='o',col='magenta')
lines(res$posterior_discrete$xmax,type='o',col='blue')
```

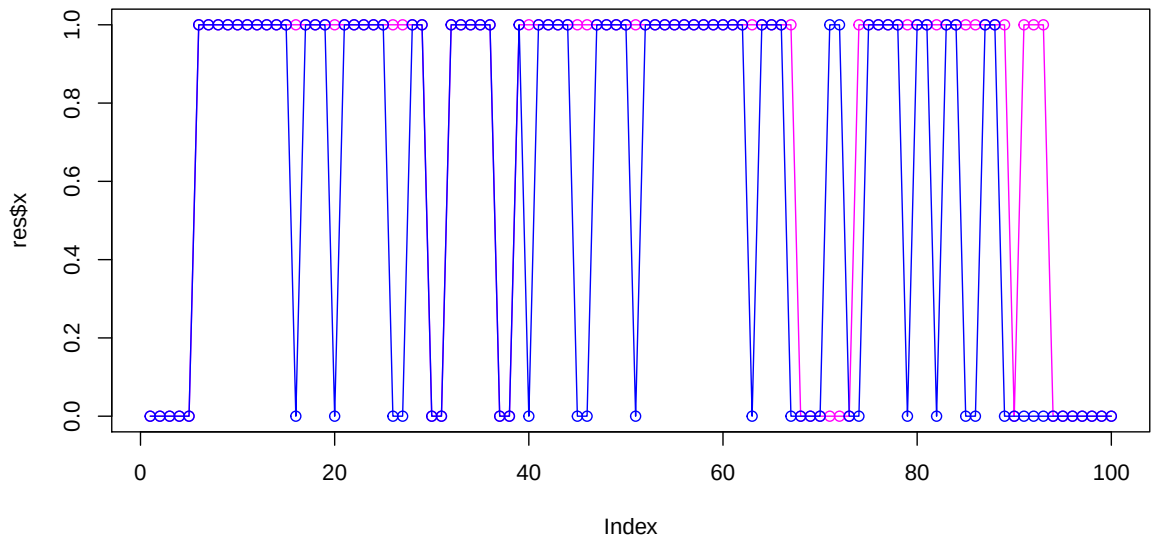

Figure 6: posterior mode (blue) and original value of $x_{1:T}$ for the discrete distribution with q=0.2 and p=0.9

```
plot(res$x,type='o',col='magenta')
lines(res$backforward_discrete$expected,type='o',col='blue')
```

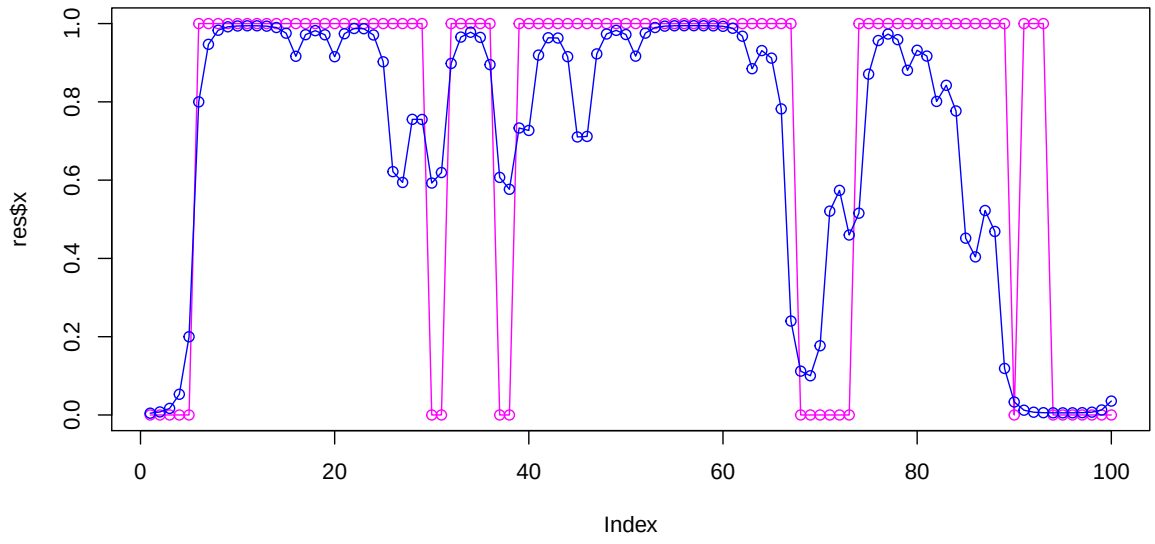

Figure 7: expected value (blue), variance (red) and original value of $x_{1:T}$ (magenta) for the discrete distribution with q=0.2 and p=0.9

```
plot(res$backforward_discrete$variance,type='o',col='red')
```

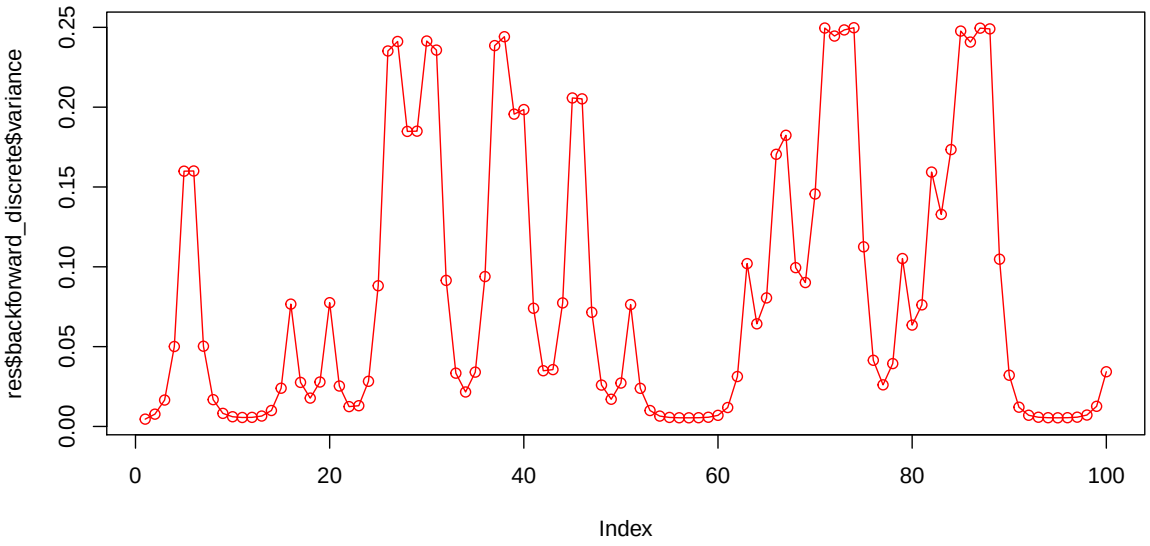

Figure 8: variance (red) of $x_{1:T}$ for the discrete distribution with q=0.2 and p=0.9

To calculate the modal configuration of p and x we can combine the expectation-maximization algorithm to find the mode of p given observation $y_{1:T}$ with the algorithm to find the mode of $x_{1:T}$ given the observation and the mode of p.

```
test_run_em = function(p,q,n){
  x = xdyn(p,n,init_p=p,c(0,1))
  y = r_obs1f(x,q)
  y2 = r_obs2f(x,q)

  #discrete case
  calcs=expectation_maximization(
    y,list(px0=c(p,1-p),pary=q,p=p,states=c(0,1)),f1yx)

  post=posterior_mode(
    y,list(px0=c(calcs$p,1-calcs$p),pary=q,p=calcs$p,states=c(0,1)),f1yx)

  #gaussian case
  calcs2=expectation_maximization(
    y2,list(px0=c(p,1-p),pary=q,p=p,states=c(0,1)),f2yx)

  post2=posterior_mode(
    y2,list(px0=c(calcs2$p,1-calcs2$p),pary=q,p=calcs2$p,states=c(0,1)),f2yx)

  return(list(x=x,
              y_discrete=y,
              y_gauss=y2,
              posterior_discrete=post,
              posterior_gauss=post2,
              em_discrete=calcs,
              em_gauss=calcs2))
}
```

Using generated data we will print the results for distinct setups

```
x=test_run_em(p=0.7,q=0.2,n=10)
# gaussian
print(x$y_gauss)

##  [1]  0.1265771  1.1176850  0.3525154  0.1142038 -0.1887704  0.1136986
##  [7]  0.8626002  0.6899610  1.0473679  1.1342504

print(x$posterior_gauss$xmax)

##  [1] 0 1 0 0 0 0 1 1 1 1

print(x$em_gauss$p)

## [1] 0.7482808

#discrete
print(x$y_discrete)

##  [1] 0 0 0 1 0 1 0 0 0 0

print(x$posterior_discrete$xmax)

##  [1] 1 1 1 0 1 0 1 1 1 1

print(x$em_discrete$p)

## [1] 0.001297367
```

```
x=test_run_em(p=0.8,q=0.8,n=10)
# gaussian
print(x$y_gauss)

##  [1]  0.20754867 -0.02648853  0.74223389  0.66032820  0.79002730
##  [6]  2.03812371  1.51320006  1.11037136  2.29841556  1.07168238

print(x$posterior_gauss$xmax)

##  [1] 0 0 1 1 1 1 1 1 1 1

print(x$em_gauss$p)

## [1] 0.9909202

#discrete
print(x$y_discrete)

##  [1] 0 0 1 0 0 1 0 1 1 1

print(x$posterior_discrete$xmax)

##  [1] 0 0 1 0 0 1 0 1 1 1

print(x$em_discrete$p)

## [1] 0.9999999
```