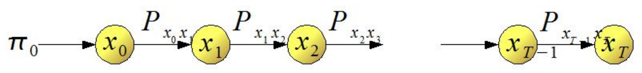
Dynamical system. When a dynamical system

$$x_{t+1} = \phi_t(x_t), \quad t = 0, \dots, T-1,$$

is probabilistic and discrete it may be determined by the **transition probability**

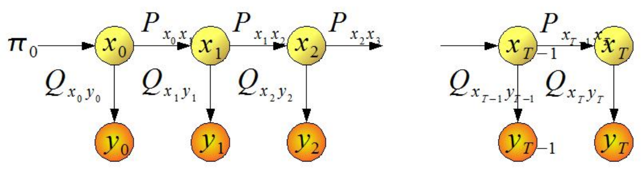
$$P_{ij} = \mathbb{P}(x_{t+1} = j \,|\, x_t = i) = \mathbb{P}(\phi_t(i) = j)$$

for finitely many states i and j with initial distribution $\pi_0(i) = \mathbb{P}(x_0 = i)$. Then it becomes a Markov chain, and the system is called **Markovian**.

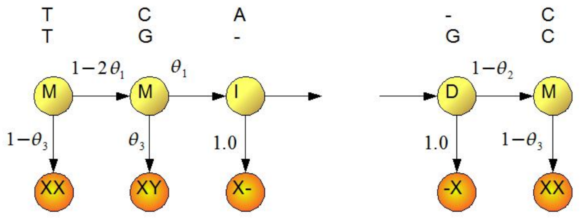


Hidden Markov model (HMM). A hidden Markov model (HMM) has two components: (i) an evolution $x_0, \dots, x_T$ of **hidden states** is Markovian and unobservable, but (ii) a sequence $y_0, \dots, y_T$ of observations is made indirectly from each hidden state x_t according to the transition probability

$$Q_{jk} = \mathbb{P}(y_t = k \,|\, x_t = j)$$



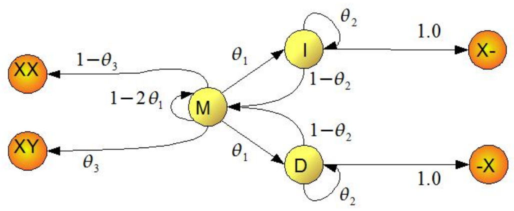
Example: DNA sequence alignment. DNA sequences are composed of four nucleotides, A, C, G, and T, and may have been acquired from the common ancestor. Regions of similarity can be identified as a consequence of evolutionary relationship due to insertions, deletions, and mutations. They are introduced as hidden states, match (M), insertion (I), and deletion (D).



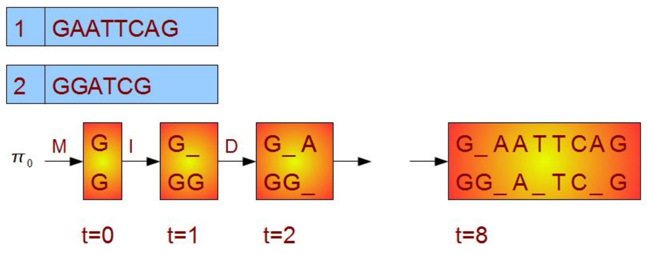
HMM for DNA sequence alignment A family of models is defined by parameter $\theta = (\theta_0, \theta_1, \theta_2, \theta_3)$. The initial distribution π_0 of hidden state x_0 has the form

$$\pi_0(M;\theta) = 1 - 2\theta_0; \quad \pi_0(I;\theta) = \pi_0(D;\theta) = \theta_0,$$

and the transition probabilities $P_{ij}(\theta)$ and $Q_{jk}(\theta)$ are expressed in terms of θ_1, θ_2, and θ_3.



A challenge in sequence alignment. Two DNA's are acquired but they are not aligned. The 1st and 2nd sequence are aligned only when hidden states, M (Match/mismatch), I (Insert to 2nd sequence), or D (Delete in 2nd sequence), are estimated. Thus, the realization $y_0, \ldots, y_T$ of aligned sequence is not fully observed; thus, it becomes a part of "hidden" states, and has to be identified.



Bayesian approach for hidden states. Suppose that a pair (X, Y) of discrete random variables has a **joint probability mass function (pmf)** $p(x, y)$. Having observed $Y = y$, we can introduce the **conditional pmf**

$$p(x|y) = \frac{p(x,y)}{p(y)}$$

Here the marginal pmf $p(y)$ is known as the **normalizing constant** in the sense that it guarantees

$$\sum_x p(x|y) = 1$$

Since $p(x|y)$ is **proportional** to $p(x, y)$, we simply write

$$p(x|y) \propto p(x,y)$$

with the symbol "$\propto$" to express "proportional to."

Conditional distribution of hidden states. The joint distribution of the evolution x_0, …, x_T, of hidden states, and the observed values y_0, …, y_T

$$\begin{aligned} &p(x_0, \dots, x_T, y_0, \dots, y_T) \\ &\quad = \pi_0(x_0) P_{x_0,x_1} Q_{x_0,y_0} \cdots P_{x_T,y_T} Q_{x_T,y_T} \end{aligned}$$

is proportional to the conditional distribution

$$\propto p(x_0, \dots, x_T \,|\, y_0, \dots, y_T)$$

of the evolution $x_0, \dots, x_T$ of the hidden states given the observed values y_0, …, y_T.

Bayesian principle: Filtering recursion. The **filtering** problem is to obtain the conditional distribution of the hidden state $x_T = i$ at the present time T

$$\begin{aligned} \pi_{T|T}(i) &= p(x_T = i \,|\, y_0, \dots, y_T) \\ &= \sum_{x_0,\dots,x_{T-1}} p(x_0, \dots, x_{T-1}, i \,|\, y_0, \dots, y_T) \end{aligned}$$

given the observed values y_0, …, y_T. Provided that there is a prior distribution π_0 for X_0, it is recursively formulated by the forward algorithm.

1. $\pi_{0|0}(i) \propto \pi_0(i) Q_{i,y_0}$
2. $\pi_{t|t}(j) \propto \sum_i \pi_{t-1|t-1}(i) P_{i,j} Q_{j,y_t}$ for $t = 1, \dots, T$.

Bayesian principle: Smoothing recursion. Let $t < T$. The **smoothing** problem is to compute the conditional distribution of the hidden state $x_t = i$ at some time t in the past

$$\begin{aligned} \pi_{t|T}(i) &= p(x_t = i \,|\, y_0, \dots, y_T) \\ &= \sum_{x_0,\dots,x_{t-1},x_{t+1},\dots,x_T} p(x_0, \dots, x_{t-1}, i, x_{t+1}, \dots, x_T \,|\, y_0, \dots, y_T) \end{aligned}$$

given the entire observation y_0, …, y_T. It combines $\pi_{t|t}(i)$ of filtering recursion forward with the following backward recursion.

1. $\beta_{T|T}(j) = 1$
2. $\beta_{k-1|T}(i) = \sum_j \beta_{k|T}(j) P_{i,j} Q_{j,y_k}$ for $k = T, \dots, t+1$.

Then we can formulate by the forward-backward procedure

$$\pi_{t|T}(i) \propto \pi_{t|t}(i) \beta_{t|T}(i)$$

Baum-Welch algorithm. Let $t < T$. The **bivariate smoothing** problem is to compute the transition probability

$$\lambda_{t|T}(i,j) = p(x_t = i, x_{t+1} = j \,|\, y_0, \dots, y_T)$$

conditioned upon the observation y_0, …, y_T. The forward-backward algorithm can be applied to obtain

$$\lambda_{t|T}(i,j) \propto \pi_{t|t}(i) P_{i,j} Q_{j,y_{t+1}} \beta_{t+1|T}(j)$$

The summation gives the univariate smoothing

$$\pi_{t|T}(i) = \sum_j \lambda_{t|T}(i,j)$$

Maximum likelihood for Markov chain. The transition probabilities P_{ij}'s are considered as unknown model parameters. Having observed the evolution x_0, x_1, …, x_T, we can infer the parameters P_{ij}'s by maximizing the likelihood

$$L = P_{x_0,x_1} \times P_{x_1,x_2} \times \cdots \times P_{x_{T-1},x_T}$$

Maximum likelihood estimate (MLE) can be obtained by maximizing $\log L$ subject to $\sum_j P_{ij} = 1$, and applying the method of Lagrange multiplier. MLE becomes proportional to the occurrence count

$$\tilde{P}_{ij} \propto \sum_{t=1}^{T} I(x_{t-1} = i, x_t = j)$$

where $I(x_{t-1} = i, x_t = j) = 1$ or 0, indicating the occurrence of transition.

Parameter inference for HMM. In general the initial probabilities $\pi_0(i;\theta)$, and the transition probabilities $P_{ij}(\theta)$ and $Q_{jk}(\theta)$ can be formulated by a vector θ of model parameters. Given the observation y_0, …, y_T,

$$L(\theta) = \sum_{x_0,\dots,x_T} \pi_0(x_0;\theta) P_{x_0,x_1}(\theta) Q_{x_0,y_0}(\theta) \cdots P_{x_{T-1},x_T}(\theta) Q_{x_T,y_T}(\theta)$$

is the likelihood. The evolution x_0, …, x_T of hidden states is not observed, and viewed as **latent variables**. It is not tractable to maximize the likelihood $L(\theta)$ with probability constraints for $\pi_0(i;\theta)$ (i.e., $\pi_0(i;\theta) \geq 0$ and $\sum_i \pi_0(i;\theta) = 1$) as well as for $P_{ij}(\theta)$, and $Q_{jk}(\theta)$. We cannot find the estimate of θ analytically.

Conditional expectation with latent variables. If the hidden transitions $x_0, \dots, x_T$ are assumed to be known, the maximum likelihood estimate for P_{ij} can be formulated with the occurrence of hidden transitions. Then having evaluated $\pi_0(i;\theta)$, $P_{ij}(\theta)$, and $Q_{jk}(\theta)$, the estimate $\tilde{\theta}$ of model parameters can be updated by the conditional expectation of occurrence count as follows.

$$\begin{aligned} P_{ij}(\tilde{\theta}) &\propto E\left[\sum_{t=1}^{T} I(x_{t-1} = i, x_t = j) \,\middle|\, y_0, \dots, y_T \right] \\ &= \sum_{t=1}^{T} \lambda_{t-1|T}(i,j) \end{aligned}$$

where the conditional probabilities $\lambda_{t-1|T}(i,j)$ can be obtained via Baum-Welch algorithm.

Baum-Welch training.

1. Estimate Internal States: Given the current estimate $\pi_0(i;\tilde{\theta})$, $P_{ij}(\tilde{\theta})$, and $Q_{jk}(\tilde{\theta})$, compute $\lambda_{t-1|T}(i,j)$ by Baum-Welch algorithm
2. Update model parameter θ by the conditional expectations:

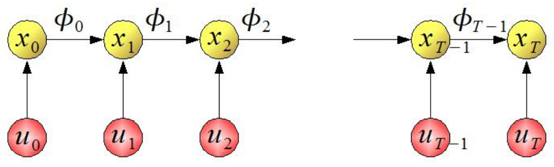
$$\begin{aligned} \pi_0(i;\tilde{\theta}) &\propto \pi_{0|T}(i) && \text{(smoothing recursion)} \\ P_{ij}(\tilde{\theta}) &\propto \sum_{t=0}^{T-1} \lambda_{t|T}(i,j) && \text{(Baum-Welch algorithm)} \\ Q_{jk}(\tilde{\theta}) &\propto \sum_{t:y_t=k} \pi_{t|T}(j) && \text{(smoothing recursion)} \end{aligned}$$

3. Repeat the above steps until it converges.

Dynamical system with control. Knowing the state x_t at time t, the control value u_t is used to determine x_{t+1}. Then the evolution of states is governed by

$$x_{t+1} = \phi_t(x_t, u_t)$$

where the maps ϕ_t's are deterministic.



Optimal control problem. Given the initial state x_0 and the control sequence $u_0, \ldots, u_T$, we obtain the trajectory $x_0, \ldots, x_T$. Then the real value

$$V = c_0(x_0, u_0) + \cdots + c_T(x_T, u_T) + k_{T+1}(x_{T+1})$$

is defined over the horizon from $t = 0$ to T, and viewed as the aggregation of the running reward c_t's and the terminal reward k_{T+1} (or cost). The **optimal control problem** is to find the control sequence $u_0, \ldots, u_T$ to maximize the reward V (or, to minimize the cost).

Dynamic programming principle. Starting from the terminal reward $k_{T+1}(x_{T+1})$, we can work backward and obtain the optimal reward $V_0(x_0)$. Then from any time t on the remaining control sequence becomes optimal.

1. $V_{T+1}(i) = k_{T+1}(i)$

2. Compute the optimal reward $V_t(x_t)$ and the corresponding control value $\psi_t(x_t)$ backward for $t = T, \dots, 0$,

$$V_t(i) = \max_{u_t} \left[c_t(i, u_t) + V_{t+1}(\phi_t(i, u_t))\right]$$
$$\psi_t(i) = \operatorname*{argmax}_{u_t} \left[c_t(i, u_t) + V_{t+1}(\phi_t(i, u_t))\right]$$

3. Set $u_0^* = \psi_0(x_0)$, and calculate $u_t^* = \psi_t(\phi_{t-1}(x_{t-1}, u_{t-1}^*))$ forward for $t = 1, \dots, T$.

Log likelihood for optimal decoding. Assume that $\pi_0(i)$, P_{ij}, and Q_{jk} for HMM are known. Given the observation y_0, …, y_T, the log likelihood of hidden states $x_0, \dots, x_T$ is viewed as the reward

$$V = k_0(x_0) + c_1(x_0, x_1) + \cdots + c_T(x_{T-1}, x_T)$$

where

$$k_0(x_0) = \log\left(\pi_0(x_0) Q_{x_0, y_0}\right)$$
$$c_t(x_{t-1}, x_t) = \log\left(P_{x_{t-1}, x_t} Q_{x_t, y_t}\right), \quad t = 1, \dots, T$$

The MLE problem is to obtain the optimal decoding x_T^*, …, x_0^* of hidden states backward. It becomes the optimal control problem with control sequence $x_{t-1}^* = u_{t-1} = \phi_t(x_t^*, u_{t-1})$ for $t = T, \dots, 1$.

Viterbi decoding. Starting from the initial cost $k_0(x_0)$, we can work forward and find the optimal value $V_T(x_T^*)$.

1. $V_0(i) = k_0(i)$

2. Compute forward for $t = 1, \dots, T$,

$$V_t(j) = \max_{x_{t-1}} \left[V_{t-1}(x_{t-1}) + c_t(x_{t-1}, j)\right]$$
$$\psi_t(j) = \operatorname*{argmax}_{x_{t-1}} \left[V_{t-1}(x_{t-1}) + c_t(x_{t-1}, j)\right]$$

3. Set $x_T^* = \operatorname*{argmax}_{x_T} V_T(x_T)$, and calculate $x_{t-1}^* = \psi_t(x_t^*)$ backward for $t = T, \dots, 1$.

Viterbi training.

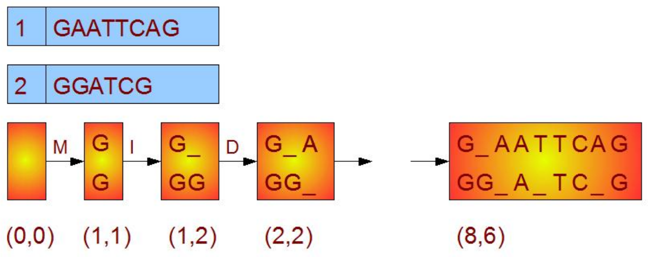
1. Estimate Internal States: Given the current estimate $\pi_0(i; \tilde{\theta})$, $P_{ij}(\tilde{\theta})$, and $Q_{jk}(\tilde{\theta})$, decode x_0^*, …, x_T^* by Viterbi algorithm.

2. Update model parameter θ by MLE:

$$P_{ij}(\tilde{\theta}) \propto \sum_{t=1}^{T} I(x_{t-1}^* = i, x_t^* = j)$$
$$Q_{jk}(\tilde{\theta}) \propto \sum_{t=0}^{T} I(x_t^* = j, y_t^* = k)$$

3. Repeat the above steps until it converges.

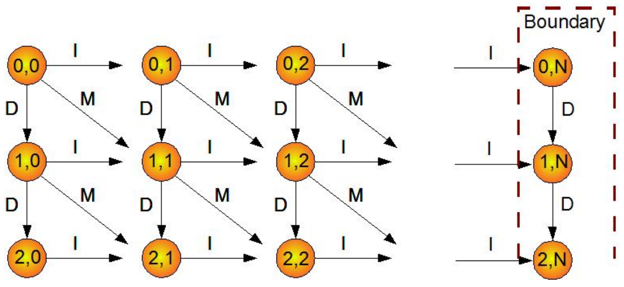
Alignment model for DNA sequence. In an alignment model for DNA sequence an HMM can be further modified. The alignment model introduces an internal state x_t recording a position in the two sequences whose movement is determined by the corresponding hidden state u_t of the t-th instruction. It decides how the two sequences are dynamically aligned: Starting from the empty state $x_0 = (0, 0)$ as the initial state of the aligned sequences, we choose the hidden value $u_t = M$ (Match/mismatch), I (Insert), or D (Delete) to add letters to the aligned sequences, and move the position x_t from x_{t-1} according to the instruction of u_t.



Dynamical system of position update. It introduces a dynamical system: Given the current position $x_t = (i, j)$ it updates with the hidden state $u_t = M$, I, or D

$$x_{t+1} = \phi_t(x_t, u_t) = \begin{cases} (i+1, j+1) & \text{if } u_t = M; \\ (i, j+1) & \text{if } u_t = I; \\ (i+1, j) & \text{if } u_t = D, \end{cases}$$

and terminates when x_T reaches the end position (n_1, n_2) with the respective sizes n_1 and n_2 of sequence.



R code: Viterbi decoding for sequence alignment. A dynamic programming with indefinite time horizon works for Viterbi decoding when it is modified for DNA sequence alignment. The memory usage of algorithm is linearly bounded by the total length of the two sequences which becomes the maximum for the time T of iterations.

```
source("hmm.r")
source("dynamic.r")
```

```
DNA = strsplit(scan("gene57.txt", what="character"), "")
cat(DNA[[1]], "\n", DNA[[2]], "\n", sep="")
th = c(0.1, 0.1, 0.1, 0.1)
out = dynamic(3, length(DNA[[1]])+length(DNA[[2]]), k0, cc)
aligned(out[[1]])
```

R code: Prediction and sequence generation. The hidden state u_{T-1} which produces the terminal position x_T can be used to generate u_T according to the transition probability P_{u_{T-1},u_T}. Then DNA sequences can be extended by the instruction of $u_T = M$, I, or D, and one of A, C, G, T can be appended to either the first, the second sequence, or both. In such a manner the two sequences are iteratively generated by the sequential process of autoregression.

```
for(k in 1:10){
  out = dynamic(3, length(DNA[[1]])+length(DNA[[2]]), k0, cc)
  DNA = prediction(out)
}
cat(DNA[[1]], "\n", DNA[[2]], "\n", sep="")
out = dynamic(3, length(DNA[[1]])+length(DNA[[2]]), k0, cc)
aligned(out[[1]])
```

Needleman-Wunsch algorithm.

At the state $x = (i, j)$ the reward function $c(x, u)$ is given by

$$c(x,u) = \begin{cases} 1 & \text{if } u = M \text{ and the pair at } (i,j) \text{ match;} \\ 0 & \text{otherwise.} \end{cases}$$

When the two sequences have N and L letters, $x = (N, j)$ or (i, L) becomes the boundary state with terminal reward $k(x) = 0$. Then the dynamical programming principle applies with indefinite time horizon.

R code: Needleman-Wunsch decoding. If the reward function is changed, the algorithm can be adjusted for Needleman-Wunsch algorithm. Note that it is not an HMM, and that it does not require the prior estimate of parameter θ. Compare the outputs with Viterbi decoding with various prior estimates for θ.

```
source("needleman.r")
DNA = strsplit(scan("gene57.txt", what="character"), "")
cat(DNA[[1]], "\n", DNA[[2]], "\n", sep="")
out = dynamic(3, length(DNA[[1]])+length(DNA[[2]]), k0, cc)
aligned(out[[1]])
```