Chapter 4

Markov Processes

4.1 Why Study Markov Processes?

As we'll see in this chapter, Markov processes are interesting in more than one respects. On the one hand, they appear as a natural extension of the finite state automata we've discussed in Chapter 3. They constitute an important theoretical concept that is encountered in many different fields. We believe therefore that it is useful for anyone (being in academia, research or industry) to have heard about the terminology of Markov processes and to be able to talk about it.

On the other hand, the study of Markov processes – more precisely *hidden* Markov processes – will lead us to algorithms that find direct application in today's technology (such as in optical character recognition or speech-to-text systems), and which constitutes an essential component within the underlying architecture of several modern devices (such as cell phones).

4.2 Markov Processes

A Markov process[1] is a stochastic extension of a finite state automaton. In a Markov process, state transitions are probabilistic, and there is – in contrast to a finite state automaton – no input to the system. Furthermore, the system is only in one state at each time step. (The nondeterminism of finite state automata should thus not be confused with the stochasticity of Markov processes.)

[1]Named after the Russian mathematician Andrey Markov (1856-1922).

Before coming to the formal definitions, let us introduce the following example, which should clearly illustrate what a Markov process is.

Example. Cheezit[2], a lazy hamster, only knows three places in its cage: (a) the pine wood shaving that offers him a bedding where it sleeps, (b) the feeding trough that supplies him with food, and (c) the wheel where it makes some exercise.

After every minute, the hamster either gets to some other activity, or keeps on doing what he's just been doing. Referring to Cheezit as a process without memory is not exaggerated at all:

- When the hamster sleeps, there are 9 chances out of 10 that it won't wake up the next minute.
- When it wakes up, there is 1 chance out of 2 that it eats and 1 chance out of 2 that it does some exercise.
- The hamster's meal only lasts for one minute, after which it does something else.
- After eating, there are 3 chances out of 10 that the hamster goes into its wheel, but most notably, there are 7 chances out of 10 that it goes back to sleep.
- Running in the wheel is tiring: there is an 80% chance that the hamster gets tired and goes back to sleep. Otherwise, it keeps running, ignoring fatigue.

4.2.1 Process Diagrams

Process diagramas offer a natural way of graphically representing Markov processes – similar to the state diagrams of finite automata (see Section 3.3.2).

For instance, the previous example with our hamster in a cage can be represented with the process diagram shown in Figure 4.1.

[2]This example is inspired by the article found on `http://fr.wikipedia.org/wiki/Chaîne_de_Markov`. The name was generated by a "Computer-Assisted Hamster Naming" technology found on `http://www.coyoteslodge.com/hamform.htm`.

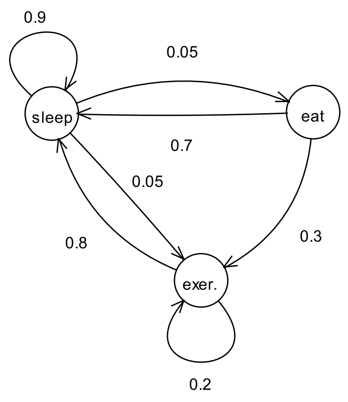

Figure 4.1: Process diagram of a Markov process.

4.2.2 Formal Definitions

Definition 4.1. A *Markov chain* is a sequence of random variables $X_1, X_2, X_3, \ldots$ with the *Markov property*, namely that the probability of any given state X_n only depends on its immediate previous state X_{n-1}. Formally:

$$P(X_n = x \mid X_{n-1} = x_{n-1}, \ldots, X_1 = x_1) = P(X_n = x \mid X_{n-1} = x_{n-1})$$

where $P(A \mid B)$ is the probability of A given B.

The possible values of X_i form a countable set S called the *state space* of the chain. If the state space is finite, and the Markov chain time-homogeneous (i.e. the transition probabilities are constant in time), the transition probability distribution can be represented by a matrix $\mathbf{P} = (p_{ij})_{i,j\in S}$, called the *transition matrix*, whose elements are defined as:

$$p_{ij} = P(X_n = j \mid X_{n-1} = i)$$

Let $\mathbf{x}^{(n)}$ be the *probability distribution* at time step n, i.e. a vector whose i-th component describe the probability of the system to be in state i at time state n:

$$\mathbf{x}_i^{(n)} = P(X_n = i)$$

Transition probabilities can be then computed as power of the transition matrix:

$$\begin{aligned}
\mathbf{x}^{(n+1)} &= \mathbf{P} \cdot \mathbf{x}^{(n)} \\
\mathbf{x}^{(n+2)} &= \mathbf{P} \cdot \mathbf{x}^{(n+1)} = \mathbf{P}^2 \cdot \mathbf{x}^{(n)} \\
\mathbf{x}^{(n)} &= \mathbf{P}^n \cdot \mathbf{x}^{(0)}
\end{aligned}$$

Example. The state space of the "hamster in a cage" Markov process is:

$$S = \{\text{sleep}, \text{eat}, \text{exercise}\}$$

and the transition matrix:

$$\mathbf{P} = \begin{pmatrix} 0.9 & 0.7 & 0.8 \\ 0.05 & 0 & 0 \\ 0.05 & 0.3 & 0.2 \end{pmatrix}$$

The transition matrix can be used to predict the probability distribution $\mathbf{x}^{(n)}$ at each time step n. For instance, let us assume that Cheezit is initially sleeping:

$$\mathbf{x}^{(0)} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

After one minute, we can predict:

$$\mathbf{x}^{(1)} = \mathbf{P} \cdot \mathbf{x}^{(0)} = \begin{pmatrix} 0.9 \\ 0.05 \\ 0.05 \end{pmatrix}$$

Thus, after one minute, there is a 90% chance that the hamster is still sleeping, 5% chance that he's eating and 5% that he's running in the wheel.

Similarly, we can predict that after two minutes:

$$\mathbf{x}^{(2)} = \mathbf{P} \cdot \mathbf{x}^{(1)} = \begin{pmatrix} 0.885 \\ 0.045 \\ 0.07 \end{pmatrix}$$

Definition 4.2. The *process diagram* of a Markov chain is a directed graph describing the Markov process. Each node represent a state from the state space. The edges are labeled by the probabilities of going from one state to the other states. Edges with zero transition probability are usually discarded.

4.2.3 Stationary Distribution

The theory shows that – in most practical cases[3] – after a certain time, the probability distribution does not depend on the initial probability distribution $\mathbf{x}^{(0)}$ anymore. In other words, the probability distribution converges towards a *stationary distribution*:

$$\mathbf{x}^* = \lim_{n\to\infty} \mathbf{x}^{(n)}$$

In particular, the stationary distribution $\mathbf{x}^*$ satisfies the following equation:

$$\mathbf{x}^* = \mathbf{P}\cdot\mathbf{x}^* \tag{4.1}$$

Example. The stationary distribution of the hamster

$$\mathbf{x}^* = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

can be obtained using Equation 4.1, as well as the fact that the probabilities add up to $x_1 + x_2 + x_3 = 1$. We obtain:

$$\mathbf{x}^* = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ 1-x_1-x_2 \end{pmatrix} = \begin{pmatrix} 0.9 & 0.7 & 0.8 \\ 0.05 & 0 & 0 \\ 0.05 & 0.3 & 0.2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ 1-x_1-x_2 \end{pmatrix}$$

From the first two components, we get:

$$\begin{aligned} x_1 &= 0.9x_1 + 0.7x_2 + 0.8(1-x_1-x_2) \\ x_2 &= 0.05x_1 \end{aligned}$$

Combining the two equations gives:

$$0.905x_1 = 0.8$$

[3]The Markov chain must be aperiodic and irreductible.

so that:

$$\begin{aligned}
x_1 &= \frac{0.8}{0.905} \approx 0.89 \\
x_2 &= 0.05x_1 \approx 0.044 \\
x_3 &= 1 - x_1 - x_2 \approx 0.072 \\
\mathbf{x}^* &\approx \begin{pmatrix} 0.89 \\ 0.044 \\ 0.072 \end{pmatrix}
\end{aligned}$$

In other words, if we observe the hamster long enough, the probability that it will be sleeping is $x_1 = 89\%$, that it will be eating $x_2 = 4\%$, and that it will be doing some exercise $x_3 = 7\%$.

4.3 Hidden Markov Models

A *hidden Markov model* (HMM) is a statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters. The challenge is to determine the hidden parameters from the observable parameters. Typically, the parameters of the model are given and the challenge is to find the most likely sequence of hidden states that could have generated a given sequence of observed states.

In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible. Rather, the observer sees an observable (or output) token. Each hidden state has a probability distribution, called *emission probability*, over the possible observable tokens. Figure 4.2 illustrates a hidden Markov model.

Example 4.1. Assume you have a friend who lives far away and to whom you talk daily over the telephone about what he did that day. Your friend is only interested in three activities: walking in the park, shopping, and cleaning his apartment. The choice of what to do is determined exclusively by the weather on a given day. You have no definite information about the weather where your friend lives, but you know general trends. Based on what he tells you he did each day, you try to guess what the weather must have been like.

You believe that the weather operates as a discrete Markov process (i.e. a Markov chain). There are two states, “Rainy” and “Sunny”, but you cannot observe them directly – they are hidden from you. On each day, there is a certain

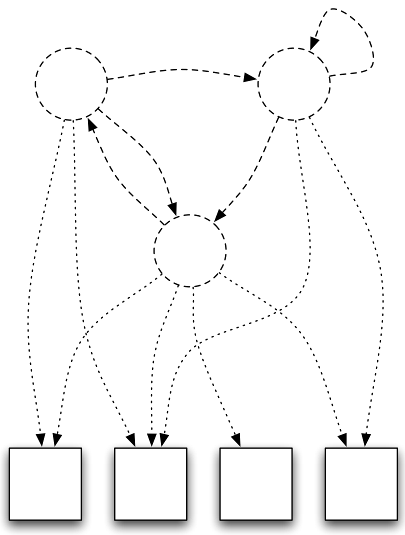

Figure 4.2: Illustration of a hidden Markov model. The upper part (dashed) represents the underlying hidden Markov process. The lower part (shadowed) represents the observable outputs. Dotted arrows represent the emission probabilities.

chance that your friend will perform one of the following activities, depending on the weather: “walk”, “shop’, or “clean”. Since your friend tells you about his activities, those are the observations.

Furthermore, you know the general weather trends in the area, and what your friend likes to do on average.

Concerning the weather, you known that:

- If one day is rainy, there is a 70% chance that the next day will be rainy too.
- If one day is sunny, there is 40% chance that the weather will degrade on the next day.

Your friend’s general habits can be summarized as follows:

- If it is rainy, there is a 50% chance that he is cleaning his apartment, and only 10% chance that he goes out for a walk.
- If it is sunny, there is a 60% chance that he is outside for a walk, 30% chance that he decides to go shopping and 10% that he stays home to clean his apartment.

The entire system is that of a hidden Markov model (HMM). The hidden state space is $S = \{\text{Rainy}, \text{Sunny}\}$, and the possible observable output state $O = \{\text{walk}, \text{shop}, \text{clean}\}$. The transition probability matrix (between hidden states) is:

$$\mathbf{P} = \begin{pmatrix} 0.7 & 0.4 \\ 0.3 & 0.6 \end{pmatrix}$$

Finally, the emission probabilities are:

	Rainy	Sunny
walk	0.1	0.6
shop	0.4	0.3
clean	0.5	0.1

The whole model can also be summarized with the following diagram:

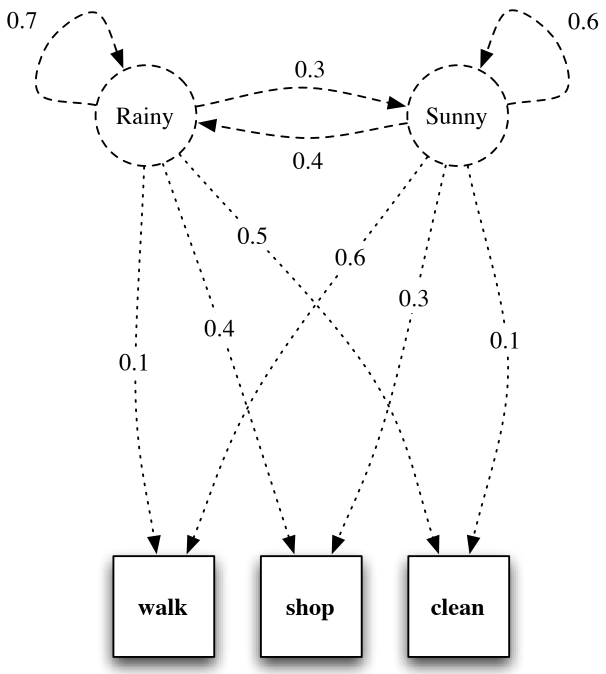

Figure 4.3: Diagram of a hidden Markov model.

4.3.1 Viterbi Algorithm

You talk to your friend three days in a row and discover that on the first day he went for a walk, on the second day he went shopping, and on the third day he cleaned his apartment.

There are many questions that can be asked from this sequence of observations, such as the overall probability of such a sequence to be observed, etc. We will however focus on the following one:

> What is the most likely sequence of rainy/sunny days that would explain these observations?

One could obviously enumerate all possible sequences of hidden states, calculate for each the probability of observing the given output sequence, and eventually pick up the most probable one.

Yet, the assumption that the underlying process (the weather) has the Markov property (i.e. the probability of each state only depends on the previous one) allows us to solve this question using a much more efficient technique: the *Viterbi algorithm*.

The algorithm

The Viterbi algorithm (also called "forward-Viterbi") works as follows. For each successive observable output and each possible hidden state, keep track of:

- the relative probability,
- and the most probable sequences of hidden states *so far* (with their corresponding probability).

The updating process from one observable output to the next includes multiplying the probabilities obtained so far with the corresponding transition probability *and* emission probability.

The algorithm is best illustrated with a concrete example. Figures 4.4 to 4.7 illustrate, for the above example (where your friend is observed to walk, then shop and finally clean his apartment), how the most probable sequence of hidden events is obtained. The most likely sequence of hidden states for the given observations turns out to be (Sunny, Rainy, Rainy).

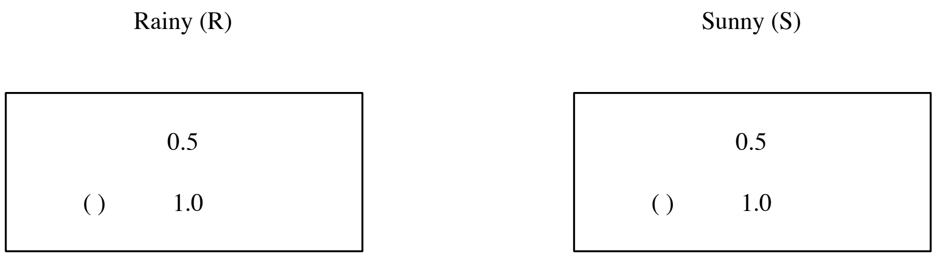

Figure 4.4: Viterbi algorithm – step 0 (initialization). All hidden states are assumed to have the same initial probability.

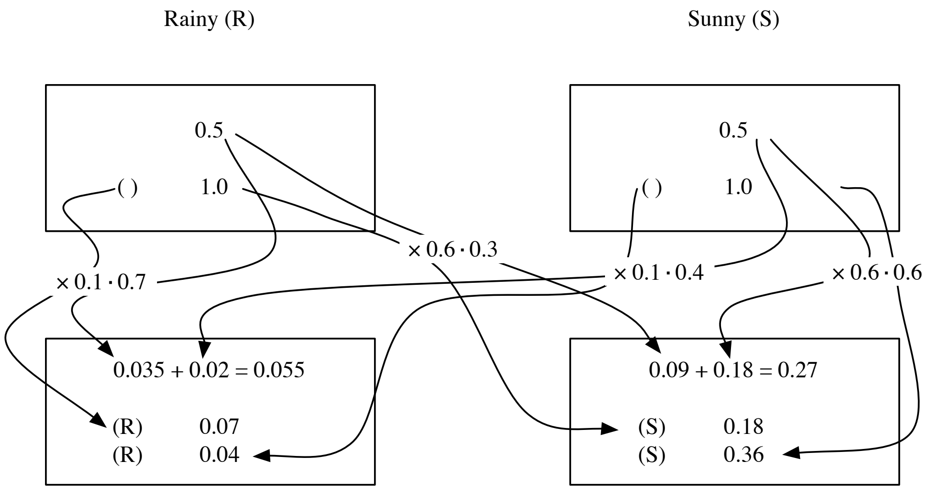

Figure 4.5: Viterbi algorithm – step 1 (observe “walk”). The new states are updated using both emission and transition probabilities (see Figure 4.3).

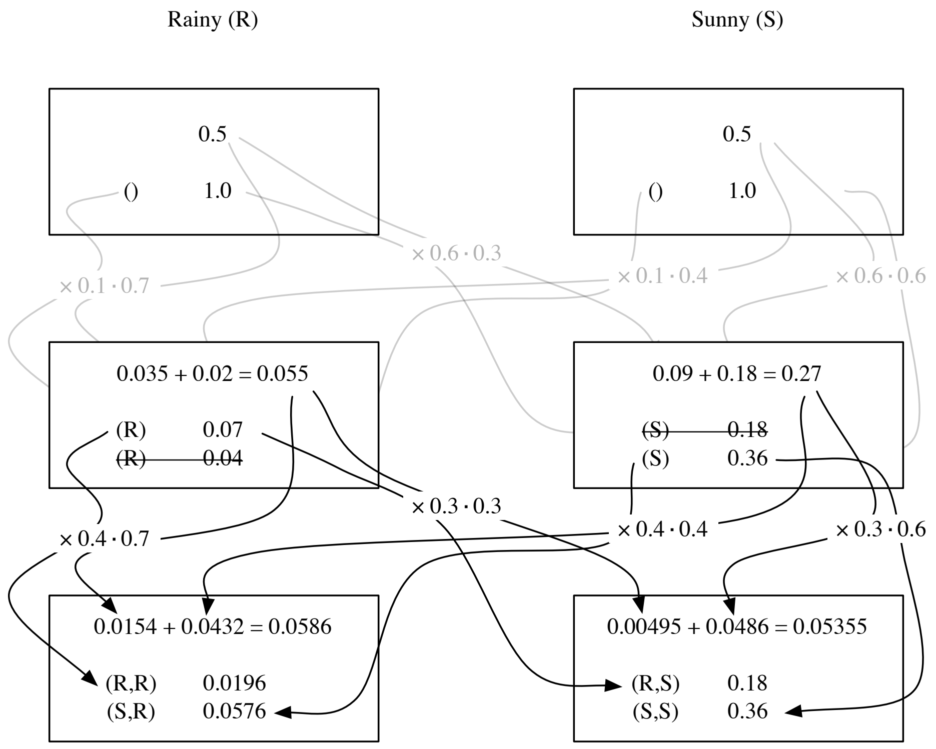

Figure 4.6: Viterbi algorithm – step 2 (observe “shop”). For each hidden state, only the most probable sequence so far is kept – all other sequences are discarded. The updating process then repeats.

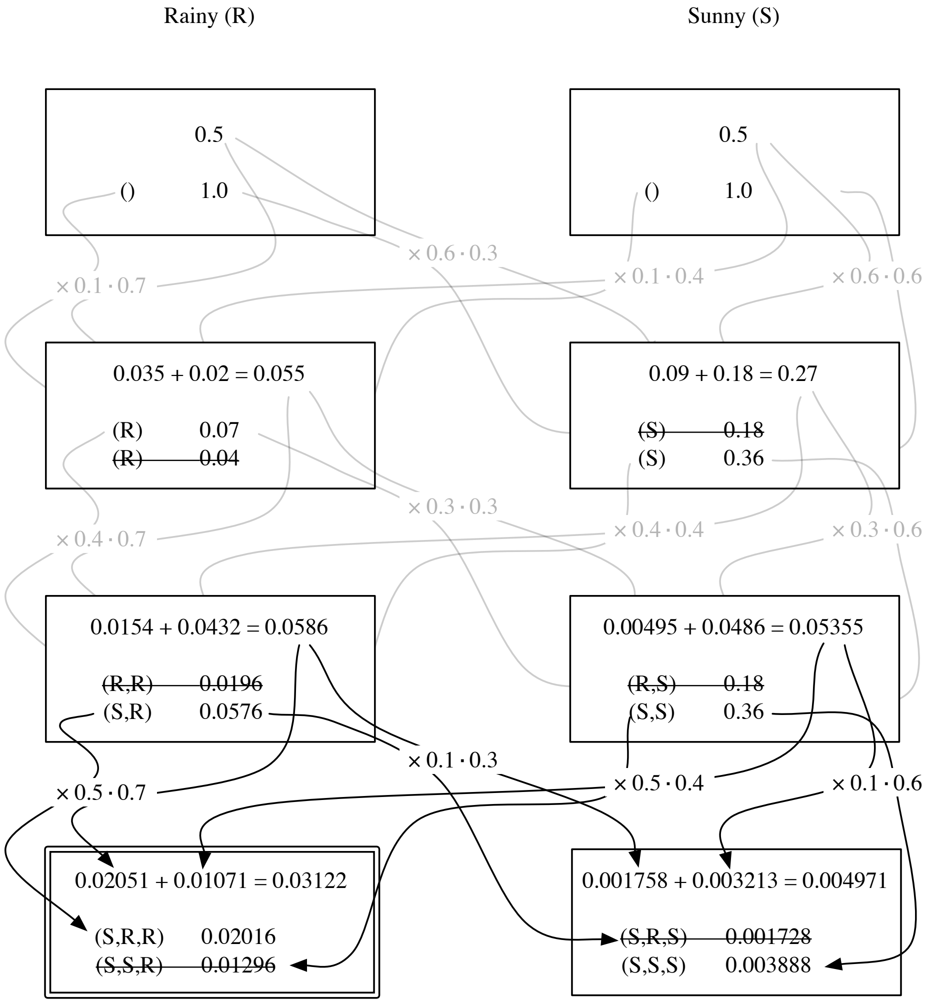

Figure 4.7: Viterbi algorithm – step 3 (observe "clean"). After the final step is processed, we only consider the most probable hidden state (here, "Rainy"). The most likely sequence of hidden states for the given observation is thus $\{$S,R,R$\}$, i.e. $\{$Sunny, Rainy, Rain$\}$.

4.3.2 Complexity of the Viterbi Algorithm

The computational complexity of the Viterbi algorithm is $\mathcal{O}(M^2 \cdot N)$, where $M = |S|$ is the number of possible hidden states, and N the number of observed outputs. In other words, the number of steps required by the Viterbi algorithm scales linearly with the length of the observed sequence.

This dramatically contrasts with a brute-foce search, which is $\mathcal{O}(M^N)$, i.e. exponential. The following Table 4.1 compares the time possibly required by the Viterbi algorithm and a brute-force algorithm, for our particular example model ($M = 2$), with observed sequences of different lengths. We assume that both algorithms take 1ms to compute the above example with $N = 3$.

	Viterbi	Brute Force
$N = 3$	1ms	1ms
$N = 10$	3.3ms	0.13s
$N = 20$	6.7ms	2.2min (!)

Table 4.1: Comparison between Viterbi and brute foce algorithms.

4.3.3 Applications

Hidden Markov models (HMM) have applications in many domains, such as temporal pattern recognition. Note however that HMM often don't offer *per se* the whole solution to a problem, but that they constitute a part of a more complete solutions, typically used to drastically speed up the processing of input sequences.

Here are some example of applications (which will be either demonstrated during the lecture or studied in some exercise sheet):

- Text recognition, optical character recognition (OCR)
  - Hidden variable: letters of the alphabet
  - Observable: simple features extracted from pixels of character maps (such as strokes and arcs)
- Predictive text input systems for portable devices such as cell phones (maybe even the iPhone?).
  - Hidden variable: letters of the alphabet

  - Observable: the sequence of keys pressed by the user

- Speech-to-Text Systems (such as IBM ViaVoice)
  - Hidden variable: phonemes, syllables of the words
  - Observable: features of frequency spectrum (such as formants, peaks, etc.)

4.4 Chapter Summary

- A *Markov process* is a stochastic extension of a finite state automaton.
- A system with the *Markov property* is a system with no memory: the transition probabilities only depend on the current state of the system.
- In a *hidden Markov model*, the underlying system is assumed to be a Markov process. The actual state of the underlying system is not directly visible. Rather, the observer sees an observable output, whose *emission probability* depends on the hidden state of the system.
- The *Viterbi algorithm* is an efficient algorithm that finds the most likely sequence of hidden states given a sequence of observations.
- *Applications* of Hidden Markov models include text recognition, predictive text input systems of portable devices and speech-to-text software.