Lecture 7

Inf1A: Probabilistic Finite State Machines and Hidden Markov Models

7.1 Introduction

The key difference between deterministic FSMs and non-deterministic FSMs is that D-FSMs have *at most one* trace for any given input string, and N-FSMs may have *many* alternative traces for a given input. It is very easy to think about a D-FSM as being a little machine where you feed in an input and follow its unique path through the machine. However, N-FSMs do not have this interpretation, because there may be many traces for a given input string. In our definition of an N-FSM we do not even have a rule for deciding which of the many paths we should try first. So it is hard to associate a *behaviour* with an N-FSM.

In this lecture we will give a very gentle introduction to *Probabilistic FSMs*, which in a sense lie between D-FSMs and N-FSMs. The essential structure of the set of transitions of a probabilistic FSM is non-deterministic (though throughout this lecture note we will assume that probabilistic FSMs do not have ϵ-transitions). However, the transitions of a probabilistic FSM are also labelled with probabilities. We will see that these probabilities will allow us to describe the behaviour of the FSM as a *random process* which will generate a random trace for any given string. That is, if q is a state of the FSM which has three arrows labelled a, all leading to different states (q, r and s, say), then each of those three transitions has a *probability* associated with it. Each probability is a non-negative real number of value at most 1, and the collection of probabilities for a given state and symbol must sum to 1. Then the likelihood that a particular transition (rather than one of the other transitions) is taken depends on the probability value for that transition. Each (state-symbol) pair of the FSM represents a different experiment defined in terms of the particular transitions and transition probabilities that are available for that (state-symbol).

Therefore, we can view a trace through the FSM for a given input string as being a sequence of trials, with one trial carried out for each state in the trace. That means that we can use basic probability to talk about the behaviour of the probabilistic FSM. For example, we can ask (and answer) questions such as "What is the probability that we

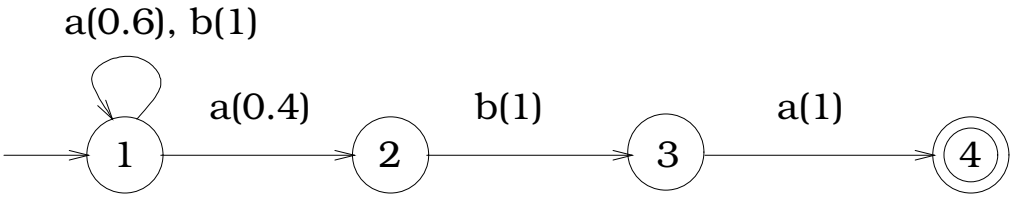

Figure 7.1: An very simple probabilistic FSM

will arrive at an accepting state when we are reading string `10011`"?

The class of probabilistic FSMs seem to lie somewhere between the class of deterministic FSMs and non-deterministic FSMs. Their structure can be as general as that of an N-FSM. However, they are not quite as powerful as N-FSMs because we do not have the freedom to choose any trace we would like - all traces have some probability of being taken, but during one particular computation, the particular trace that is taken is a random trace, whose likelihood is governed by the values of the probabilities labelling the transitions.

In practice, the most common type of probabilistic FSM is something called a Hidden Markov Model (HMM), which is a *transducer-style* FSM in which the outputs, as well as the transitions, are governed by a random process. Hidden Markov Models are important in application areas such as *computational linguistics*, *speech processing* and *gene prediction algorithms*. We will discuss Hidden Markov Models briefly in Section 7.3.

7.2 Basic Probability & Probabilistic FSMs

Some (very) Basic Probability

Probability Theory was developed to give us a firm foundation for talking about random events.

Suppose that we consider an *experiment* of some kind, and that there are a number of possible *outcomes* of that experiment. Usually the sort of things we are interested in are experiments which can be performed repeatedly. In this case we refer to a single execution of the experiment as a *trial*.

For an experiment which has the set of possible outcomes $\{a_1, \ldots, a_k\}$, we can use a *random variable* X to represent the unpredictable outcome of a single trial. We can also use *probabilities* to represent the chance with which the experiment yields a certain outcome: that is, we have certain values $p_1, \ldots, p_k \in [0, 1]$ such that

$$\begin{aligned} p_i &= \Pr[X = a_i] \qquad \text{and} \\ \sum_{i=1}^{k} p_i &= 1. \end{aligned}$$

Whenever we write the term $\Pr[.]$ we take this to mean "the probability of .".

A nice example of an experiment is the example of throwing a dice. In this case the outcome can be any of the six values $1, 2, 3, 4, 5, 6$. We can use a random variable X to represent one trial of this experiment, where $p_i = \Pr[X = i] = 1/6$ for every $i = 1, \ldots, 6$.

If we consider the probabilistic Finite State Machine in Figure 7.1, then notice that if the current state is the start state 1, and the current symbol is a, then there are two outgoing transitions for that symbol. The transition returning back to the state 1 has probability 0.6, and the transition to the state 2 has probability 0.4. We can write this as

$$\begin{array}{rcrcl} \Pr[1 \xrightarrow{\mathsf{a}} 1] & = & 0.6 & \quad \Pr[1 \xrightarrow{\mathsf{a}} 2] & = & 0.4 \\ \Pr[1 \xrightarrow{\mathsf{a}} 3] & = & 0 & \quad \Pr[1 \xrightarrow{\mathsf{a}} 4] & = & 0 \end{array}$$

Let Q, as always, denote the set of states, in this case $\{1, 2, 3, 4\}$. Then, because we insist that $\sum_{q \in Q} [1 \xrightarrow{\mathsf{a}} q] = 1$, therefore the transition for the state 1 and symbol a is a well-defined random variable. We imagine that whenever our probabilistic FSM is reading a string, and we arrive at state q and our next symbol is a, the next state q' is chosen randomly according to the probabilities for the transitions leaving q labelled with a. In the case when there is *no* transition leaving q labelled by a, then we reject the string and the random process is terminated.

Probabilistic FSMs

We will not give a *formal* definition of probabilistic FSMs in this Lecture Note. Essentially the definition of a probabilistic FSM is the same as Definition 3.1 of a N-FSM (with no ϵ-transitions), except that the definition must include probabilities for all the transitions of the FSM. Specifically, we require that for every state $q \in Q$ and every symbol $\mathsf{a} \in \Sigma$, that one of the following holds:

$$\begin{array}{rl} \text{either} & \Delta(q, \mathsf{a}) = \emptyset \\ \text{or} & \displaystyle\sum_{q' \in \Delta(q,\mathsf{a})} \Pr[q \xrightarrow{\mathsf{a}} q'] = 1. \end{array}$$

For any input string $x \in \Sigma^*$ of length k and every *trace* t (as defined in Lecture Note 3) of the form

$$t = s_0, x_1, q_1, \ldots, q_{k-1}, x_k, q_k$$

of the Finite State Machine, we can calculate the probability that this trace represents the path taken through the FSM on reading x. The probability that this is the path taken is

$$\Pr[t] \;=\; \Pr[s_0 \xrightarrow{x_1} q_1] \Pr[q_1 \xrightarrow{x_2} q_2] \ldots \Pr[q_{k-1} \xrightarrow{x_k} q_k].$$

We adopt the very reasonable convention that $\Pr[q \xrightarrow{\mathsf{a}} q'] = 0$ whenever $q' \notin \Delta(q, \mathsf{a})$.

We can also evaluate the probability that a string x is accepted by the probabilistic FSM. We consider the set $T(x)$ of all traces for x such that the final state q_k of the trace is an accept state, and we sum $\Pr[t]$ over all such traces t. That is

$$\Pr[x \text{ is accepted}] \;=\; \sum_{t \in T(x), q_k \in F} \Pr[t].$$

Example 7.1: Consider the probabilistic FSM of Figure 7.1. Consider the input string aaba. Notice that there is actually an accepting trace for this string, so if we treated our machine as an N-FSM, the string would be accepted. What is the probability that the string

is accepted? Well, the three possible traces for the string are

$$\begin{array}{lcll} t_1 & = & 1, \mathsf{a}, 1, \mathsf{a}, 1, \mathsf{b}, 1, \mathsf{a}, 1 & \\ t_2 & = & 1, \mathsf{a}, 1, \mathsf{a}, 2, \mathsf{b}, 3, \mathsf{a}, 4 & \text{and} \\ t_3 & = & 1, \mathsf{a}, 1, \mathsf{a}, 1, \mathsf{b}, 1, \mathsf{a}, 2. & \end{array}$$

Only the second of these three traces leads to an accepting state. Therefore the probability that the computation for `aaba` ends in an accepting state is exactly the probability $\Pr[t_2]$ that the random trace for x will be t_2. This probability is

$$\Pr[t_2] \;=\; \Pr[1 \xrightarrow{\mathsf{a}} 1]\Pr[1 \xrightarrow{\mathsf{a}} 2]\Pr[2 \xrightarrow{\mathsf{b}} 3]\Pr[3 \xrightarrow{\mathsf{a}} 4] = 0.6 * 0.4 * 1 * 1 = 0.24.$$

Notice that for the probabilistic FSM of Figure 7.1, it is quite likely that the random process for generating a path through the FSM will reject the string `aaba` for the simple reason that it ends up in a state where there is no transition at all for the next symbol. Therefore it is interesting for this example to evaluate the probability that the random process generates an entire trace for string `aaba` (this is equal to $\Pr[t_1] + \Pr[t_2] + \Pr[t_3]$). It is also interesting to compare this with the probability that the process generates an entire trace for `bbbb`.

7.3 Hidden Markov Models

As we mentioned earlier, most applications which model data using probabilistic FSMs actually work with a variant of this model called *Hidden Markov Models (HMMs)*. Hidden Markov Models are essentially *Transducer-style* probabilistic FSMs with one twist: the outputs generated by the HMM are generated at states of the FSM (rather than along transitions), and also, the *outputs* generated at a state are *randomly-generated*, according to some random variable modelling what happens at that state.

The term *Hidden Markov Model* is used to reflect the fact that in most applications which use HMMs to model behaviour, the states or transitions corresponding to that behaviour will not be visible in the data generated by the process. For example, if we consider the problem of modelling speech patterns, it is reasonable (and common) to assume that certain rhythms occur with more frequency than others - for example, it may be likely that certain sounds (or *phonemes*) may follow one particular *phoneme*. This can sometimes by modelled by assuming that after hearing this particular phoneme, we transition to a state which has an associated random process which outputs the "next phoneme" with the appropriate probability (favouring certain likely phonemes).

With this in mind, a Hidden Markov Model will include an output alphabet denoted Λ. For each state $q \in Q$ of the HMM, we will have associated with that state a random variable taking values in Λ according to certain (state-dependent) probabilities. Whenever we perform a computation on a HMM, we will obtain a sequence of outputs, or *observations*, which are the outputs generated by the states which are visited during that computation.

It is sometimes the case that a HMM does not have an explicit input alphabet, when the phenomenon being modelled is a phenomenon with well-defined time steps.

In Figure 7.2, we present a Hidden Markov Model to model the typical weather patterns from day-to-day in Edinburgh during winter time. There are three states in this HMM,

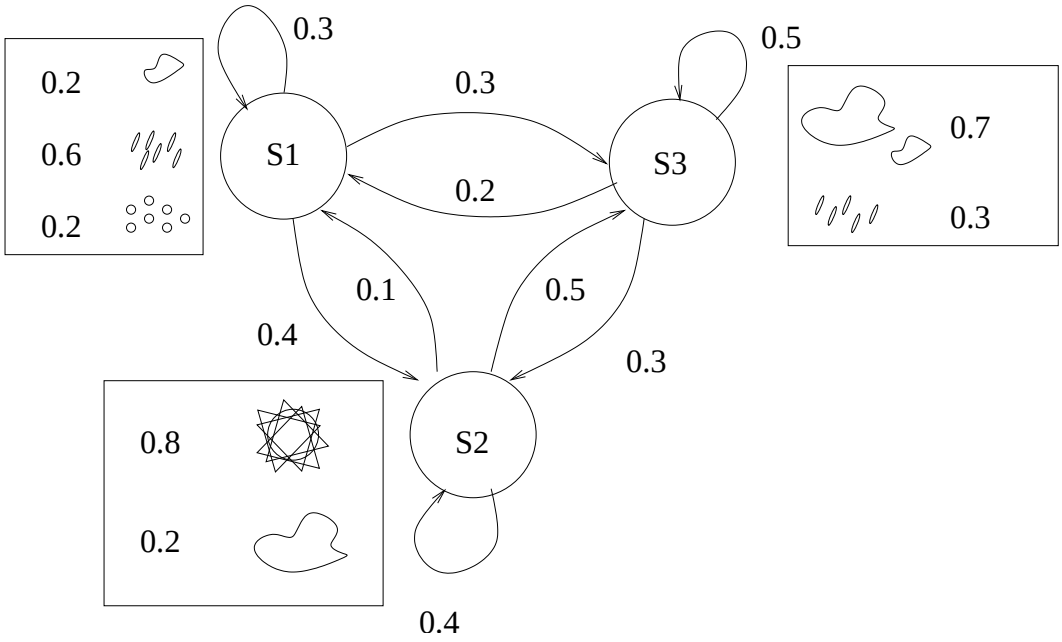

Figure 7.2: A Hidden Markov Model modelling the weather changes from day-to-day during winter in Edinburgh

each corresponding to a different type of meteorological situation which is common in winter. In this particular HMM, we do not concern ourselves with an input alphabet, because we assume that transitions are being made on a day-to-day basis. The output generated by any state consists of a label stating the dominant "weather condition" for the current day, and is generated according to the *probabilistic distribution* (or *random variable* - essentially a "random rule") for outputs from that state. For example, whenever the current state is $S1$, the most likely output is "rain" (with probability 0.6), and "cloud" (probability 0.2) and "snow" (probability 0.2) are also possible weather conditions in this meteorological state.

For any given Hidden Markov Model, we can work out the probabilities of seeing a certain sequence of observations. For example, suppose for now that we initially start at the state $S1$. What is the probability of seeing the following sequence?

"cloud", "rain", "cloud", "sun"

Since we are given the initial state in advance, we know the probability that we see "cloud" on the initial day. To examine what happens on subsequent days, we need to consider all possible traces which have the possibility of generating the sequence "rain", "cloud", "sun". These traces are:

$$
\begin{aligned}
t_1 &= S1(\text{“}cloud''), S1(\text{“}rain''), S1(\text{“}cloud''), S2(\text{“}sun'') \\
t_2 &= S1(\text{“}cloud''), S1(\text{“}rain''), S3(\text{“}cloud''), S2(\text{“}sun'') \\
t_3 &= S1(\text{“}cloud''), S1(\text{“}rain''), S2(\text{“}cloud''), S2(\text{“}sun'') \\
t_4 &= S1(\text{“}cloud''), S3(\text{“}rain''), S3(\text{“}cloud''), S2(\text{“}sun'') \\
t_5 &= S1(\text{“}cloud''), S3(\text{“}rain''), S2(\text{“}cloud''), S2(\text{“}sun'') \\
t_6 &= S1(\text{“}cloud''), S3(\text{“}rain''), S1(\text{“}cloud''), S2(\text{“}sun'')
\end{aligned}
$$

The probability of a given trace can be calculated in terms of the transition probabilities *and* the output probabilities along the path. For example, we compute $\Pr[t_1]$ as follows:

$$\begin{aligned}\Pr[t_1] &= \Pr[\text{S1="cloud"}]\Pr[S1 \to S1]\Pr[\text{S1="rain"}]\Pr[S1 \to S1] \\ &\quad \Pr[\text{S1="cloud"}]\Pr[S1 \to S2]\Pr[\text{S2="sun"}] \\ &= (0.2) \times (0.3) \times (0.6) \times (0.3) \times (0.2) \times (0.4) \times (0.8) \\ &= 0.0006912.\end{aligned}$$

It is a nice exercise to compute the probabilities of all the other traces, and therefore work out the overall probability of the sequence of observations "cloud", "rain", "cloud", "sun" (starting in $S1$).

In the applications that use Hidden Markov Models, it is typically the case that the detailed structure of the HMM will not be known, and that the probabilities labelling the transitions and states will certainly not be known. Therefore people who work with HMMs often have as their goal the hope of using data from the real world to find out the structure and probabilities of the underlying HMM (assuming that a HMM can be a good model for the particular phenomenon being modelled). Therefore it is very important to have precise ways of computing the likelihood of certain observations in a given HMM, so that we will be able to evaluate whether a given HMM is a good match for certain data.

7.4 Learning a HMM from Data

A very common problem that arises in *Artificial Intelligence* is the problem of *inferring* a Hidden Markov Model from real world data. There are many many examples of this sort of work being carried out, for application areas as diverse as *speech prediction*, *gene recognition* and *astronomy*.

Moray Allan and Chris Williams (of the School of Informatics, University of Edinburgh) consider an especially nice application, which is the problem of learning harmonic structures from chorale harmonies. For this particular application, it is the case that certain aspects of the HMM structure are reflected in the musical data, even though the HMM probabilities are not. We do not have the time (or the technical background) to study this paper in the course, but it is available online if you are interested in reading it.

- "Harmonising chorales by probabilistic inference", by Moray Allan and Chris Williams; to appear in *Advances in Neural Information processing Systems*, **17**, 2005. Available online from:
  `http://www.tardis.ed.ac.uk/~moray/papers/harmony.pdf`