Markov Chains and hidden Markov models

Wouter Duivesteijn

February 28, 2006

Table of contents

What is a Markov Chain?

A stochastic process (family of random variables) $\{X_n, n = 0, 1, 2, \ldots\}$, satisfying

What is a Markov Chain?

A stochastic process (family of random variables) $\{X_n, n = 0, 1, 2, \ldots\}$, satisfying

- it takes on a finite or countable number of possible values. If $X_n = i$,the process is said to be in state i at time n;

What is a Markov Chain?

A stochastic process (family of random variables) $\{X_n, n = 0, 1, 2, \ldots\}$, satisfying

- it takes on a finite or countable number of possible values. If $X_n = i$,the process is said to be in state i at time n;
- whenever the process is in state i, there is a fixed probability P_{ij} that it will next be in state j. Formally:

$$P\{X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \ldots, X_1 = i_1, X_0 = i_0\} = P_{ij}$$

for all states $i_0, i_1, \ldots, i_{n-1}, i_n, i, j$ and all $n \geq 0$.

What is a Markov Chain?

A stochastic process (family of random variables) $\{X_n, n = 0, 1, 2, \ldots\}$, satisfying

- it takes on a finite or countable number of possible values. If $X_n = i$,the process is said to be in state i at time n;
- whenever the process is in state i, there is a fixed probability P_{ij} that it will next be in state j. Formally:

$$P\{X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \ldots, X_1 = i_1, X_0 = i_0\} = P_{ij}$$

for all states $i_0, i_1, \ldots, i_{n-1}, i_n, i, j$ and all $n \geq 0$.

In other words:

Markov Chain property

Given the present, the future is independent of the past.

Representing probabilities in a matrix

We can display all one-step transition probabilities in one matrix:

$$P = \begin{pmatrix} P_{00} & P_{01} & P_{02} & \cdots \\ P_{10} & P_{11} & P_{12} & \cdots \\ P_{20} & P_{21} & P_{22} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Representing probabilities in a matrix

We can display all one-step transition probabilities in one matrix:

$$P = \begin{pmatrix} P_{00} & P_{01} & P_{02} & \cdots \\ P_{10} & P_{11} & P_{12} & \cdots \\ P_{20} & P_{21} & P_{22} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Probabilities are non-negative, and at each step the process must make a transition into some state. Thus, our matrix is *stochastic*;

- $P_{ij} \geq 0$
- $\sum_{j=0}^{\infty} P_{ij} = 1$

Simple weather example

Suppose that if it rains today, it will rain tomorrow with probability α, and if it does not rain today, it will rain tomorrow with probability β. We say that the process is in state 0 when it rains, and state 1 when it does not rain.

Markov Chain

We can model this by a Markov Chain.

Simple weather example

Suppose that if it rains today, it will rain tomorrow with probability α, and if it does not rain today, it will rain tomorrow with probability β. We say that the process is in state 0 when it rains, and state 1 when it does not rain.

Markov Chain

We can model this by a Markov Chain.
The transition probabilities are given by

$$P = \begin{pmatrix} \alpha & 1-\alpha \\ \beta & 1-\beta \end{pmatrix}$$

Slightly less simple weather example

Suppose that if it has rained for the past two days, it will rain tomorrow with probability 0.7; if it rained today but not yesterday, it will rain tomorrow with probability 0.5; if it rained yesterday but not today, it will rain tomorrow with probability 0.4; if it has not rained in the past two days, it will rain tomorrow with probability 0.2.

Slightly less simple weather example

Suppose that if it has rained for the past two days, it will rain tomorrow with probability 0.7; if it rained today but not yesterday, it will rain tomorrow with probability 0.5; if it rained yesterday but not today, it will rain tomorrow with probability 0.4; if it has not rained in the past two days, it will rain tomorrow with probability 0.2.

Not a Markov Chain

Given the weather today, the weather tomorrow depends on the weather yesterday.

Can we fix this? (How?)

Make more states!

We say the process is in state

0 if it rained both today and yesterday
1 if it rained today but not yesterday
2 if it rained yesterday but not today
3 if it did not rain either yesterday or today

The story on the previous slide now represents a four-state Markov Chain with transition probability matrix

$$P = \begin{pmatrix} 0.7 & 0 & 0.3 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.4 & 0 & 0.6 \\ 0 & 0.2 & 0 & 0.8 \end{pmatrix}$$

Application in Biology

Page 47 of the book: CpG islands example. In this example, dinucleotides are important. We need a model that generates sequences in which the probability of a symbol depends on the previous symbol. Therefore: a Markov Chain.

States

For DNA, we need states A, C, G, and T.

Application in Biology

Page 47 of the book: CpG islands example. In this example, dinucleotides are important. We need a model that generates sequences in which the probability of a symbol depends on the previous symbol. Therefore: a Markov Chain.

States

For DNA, we need states A, C, G, and T. To model the begin and the end of the sequence, we add two special states $\mathcal{B}$ and $\mathcal{E}$.

Application in Biology

Page 47 of the book: CpG islands example. In this example, dinucleotides are important. We need a model that generates sequences in which the probability of a symbol depends on the previous symbol. Therefore: a Markov Chain.

States

For DNA, we need states A, C, G, and T. To model the begin and the end of the sequence, we add two special states $\mathcal{B}$ and $\mathcal{E}$.

By adding an end state, we model a distribution of lengths of the sequence. This distribution decays exponentially.

Why extending the model?

CpG islands example: given a long piece of sequence, how can we find regions with many more C and G nucleotides than elsewhere (these are called CpG islands), if there are any?

We can use the Markov Chain model, by calculating the log-odds score for a window of, say, 100 nucleotides around every nucleotide. Biologists believe that CpG islands have sharp boundaries, and are of variable length.

Again, we have a problem with a Markov Chain. How do we solve this?

What is an HMM?

Call the state sequence π; this follows a simple Markov Chain. Just as before, we introduce a begin state and an end state, both denoted as state 0.

New are the *emission probabilities*; the probability that symbol b is seen when in state k (denoted $e_k(b)$).
For the CpG model, all $e_k(b)$ are 1 or 0.

Given the symbol sequence, the state sequence is hidden.

Probability of a sequence

Joint probability of an observed sequence x and a state sequence π:

$$P(x, \pi) = P_{0\pi_1} \prod_{i=1}^{L} e_{\pi_i}(x_i) P_{\pi_i \pi_{i+1}}$$

where $\pi_{L+1} = 0$.

This formula is not useful in practice; in general we do not know the path.

What algorithms?

- Find the most likely sequence of (hidden) states which could have generated a given output sequence. Solved by the Viterbi algorithm.
- Compute the probability of a particular output sequence. Solved by the forward algorithm.
- Compute the posterior probability of state k at time i given the emitted sequence. Solved by the backward algorithm.
- What it's all about: algorithms to translate the emitted sequence into a sequence of states (decoding).

Viterbi algorithm (1/2)

It is no longer possible to tell what state the system is in by looking at the corresponding symbol. Often we are interested in the sequence of underlying states.

A predicted path through the HMM will tell us which part of the sequence is predicted to be a CpG island. For our prediction, we want to use the path with the highest probability:

$$\pi' = \text{argmax}_\pi P(x, \pi)$$

This path can be found recursively.

Viterbi algorithm (2/2)

All sequences have to start in the begin state; state 0. Denote by $v_k(i)$ the probability of the most probable path ending in state k with observation i. We assume there is an end state.

Suppose the $v_k(i)$ are known for all states k. Then the $v_\ell(i+1)$ for observation x_{i+1} and for all states ℓ are

$$v_\ell(i+1) = e_\ell(x_{i+1}) \max_k (v_k(i) P_{k\ell})$$

By saving back pointers, the actual state sequence can be found.

Implementational issue

Both for the Viterbi algorithm and the algorithms described later, we have to multiply very many probabilities $\rightsquigarrow$ underflow.

Implementational issue

Both for the Viterbi algorithm and the algorithms described later, we have to multiply very many probabilities $\rightsquigarrow$ underflow.

Preventing underflow

Always perform the Viterbi algorithm in log space. This will be discussed on Thursday.

Forward algorithm (1/2)

For Markov Chains we knew how to calculate the probability of a sequence, $P(x)$. We want to calculate this quantity for an HMM.

In an HMM, many different state paths can lead to the same sequence x, so

$$P(x) = \sum_{\pi} P(x, \pi)$$

The number of possible paths increases exponentially with the length of the sequence, so we do not want to brute force evaluate this. Again, we shall use a dynamic programming approach.

Forward algorithm (2/2)

In the Viterbi algorithm, replace the maximisation steps with sums. This is called the *forward algorithm*.

Instead of $v_k(i)$ we have the probability of the observed sequence up to and including x_i, requiring that $\pi_i = k$;

$$f_k(i) = P(x_1 \ldots x_i, \pi_i = k)$$

The recursion equation becomes

$$f_\ell(i+1) = e_\ell(x_{i+1}) \sum_k f_k(i) P_{k\ell}$$

for all states ℓ. Now $P(x) = \sum_k f_k(L) P_{k0}$.

Backward algorithm (1/2)

Viterbi algorithm finds the most probable path through the model. But, we might want to know the probability that observation x_i came from state k given the observed sequence, i.e. $P(\pi_i = k|x)$.

First calculate the probability of producing the entire sequence with the ith symbol being produced by state k:

$$\begin{aligned} P(x, \pi_i = k) &= P(x_1 \dots x_i, \pi_i = k) P(x_{i+1} \dots x_L | x_1 \dots x_i, \pi_i = k) \\ &= P(x_1 \dots x_i, \pi_i = k) P(x_{i+1} \dots x_L | \pi_i = k) \end{aligned}$$

The first term is the $f_k(i)$ from the forward algorithm. The second term is called $b_k(i)$, so we can write

$$P(x, \pi_i = k) = f_k(i) b_k(i)$$

Backward algorithm (2/2)

Recall: $b_k(i) = P(x_{i+1} \dots x_L | \pi_i = k)$. We start at the end of the sequence, setting $b_k(L) = P_{k0}$ for all states k. Then, we obtain the other $b_k(i)$ by a backward recursion:

$$b_k(i) = \sum_\ell P_{k\ell} e_\ell(x_{i+1}) b_\ell(i+1)$$

Now, by conditioning, we can calculate

$$P(\pi_i = k|x) = \frac{P(x, \pi_i = k)}{P(x)} = \frac{f_k(i) b_k(i)}{P(x)}$$

where $P(x)$ is the result of the forward calculation.

In fact, in the termination of the backward algorithm, we can calculate

$$P(x) = \sum_\ell P_{0\ell} e_\ell(x_1) b_\ell(1)$$

New decoding algorithms (1/2)

Viterbi decoding: select the most probable path. Forget about the others. Not very good if many paths have nearly the same probability as the most probable one.

New decoding algorithms (1/2)

Viterbi decoding: select the most probable path. Forget about the others. Not very good if many paths have nearly the same probability as the most probable one.

Another approach:

$$\hat{\pi}_i = \text{argmax}_k P(\pi_i = k|x)$$

More appropriate when interested in the state at a particular point i, rather than the complete path. The sequence defined by $\hat{\pi}$ may even not be legitimate.

New decoding algorithms (2/2)

Another, more sophisticated approach: assume we have a function $g(k)$ defined on the states. We then consider

$$G(i|x) = \sum_k P(\pi_k|x)g(k)$$

Now let $g(k)$ take the value 1 for a subset of the states and 0 for the rest. Then $G(i|x)$ is the posterior probability of the symbol i coming from a state in the specified set.

For the CpG island model, let $g(k) = 1$ for $k \in \{A^+, C^+, G^+, T^+\}$, and 0 for all other states. Then $G(i|x)$ is the posterior probability according to the model that base i is in a CpG island.