Noisy Pattern Search using Hidden Markov Models

David Barber
University College London

March 13, 2012

1 String search

We have a pattern string `david` that we wish to search for in an observation string, say `fgidavidjj`. More generally, we might have a dictionary of patterns

```
david
anton
fred
jim
barry
```

and wish to check where they may appear in the observation string. There are classical fast algorithms that can do this, for example the Aho-Corasick algorithm [1]. The complexity of the Aho-Corasick algorithm is linear in the length of the observation string and linear in the total length of the dictionary.

2 Noisy pattern search

When the observation string is potentially corrupted with noise, we cannot apply the standard Aho-Corasick algorithm. One approach in this case is to use a Hidden Markov Model (HMM) which defines a distribution over a set of observations $v_1, \ldots, v_T$ (or $v_{1:T}$ for short) and corresponding hidden variables $h_{1:T}$:

$$p(v_{1:T}, h_{1:T}) = p(h_1)p(v_1|h_1)\prod_{t=2}^{T} p(v_t|h_t)p(h_t|h_{t-1}) \tag{1}$$

whose structure as a belief network is represented in figure(1). In this case the hidden variables $h_{1:T}$ represent the 'clean' sequence and the variables $v_{1:T}$ the observed sequence. The 'emission' matrix

$$p(v_t = i|h_t = j) \equiv B_{ij}, \qquad \sum_i B_{ij} = 1$$

describes the probability that the clean state j gets corrupted to the observed state i. The 'transition' matrix

$$p(h_t = i|h_{t-1} = j) \equiv A_{ij}, \qquad \sum_i A_{ij} = 1$$

describes the transition of the latent variables. Finally

$$p(h_1 = i) = \pi_i$$

describes the starting state probability.

The HMM is a generative model which means that we can understand the model by how it generates data. In this case we would sample a hidden state from the distribution $p(h_1)$, for example state 3, and then draw a sample from the distribution $p(v_1|h = 3)$ by drawing from the distribution represented by $B_{:,3}$. Then we may draw h_2 from $A_{:,3}$ and continue in this manner until we've draw a set of states $h_{1:T}$ and $v_{1:T}$.

Whilst the HMM is best understood in the generative sense we can use it to find out something about the hidden states that gave rise to an observation sequence. That is given the sequence $v_{1:T}$ we wish to infer something about the hidden variables. In particular we often wish to infer:

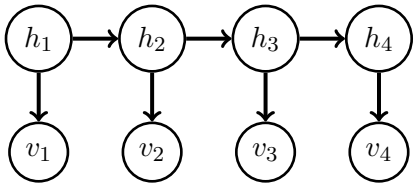

Figure 1: A Hidden Markov Model for 4 timesteps with observations $v_1, \ldots, v_4$ and hidden variables $h_1, \ldots, h_4$.

Most probable explanation What is the most likely hidden sequence that gave rise to the observations?

$$h^*_{1:T} \equiv \arg\max_{h_{1:T}} p(h_{1:T}|v_{1:T})$$

This is also called the Viterbi sequence and has corresponding probability

$$p(h^*_{1:T}|v_{1:T})$$

In practice may wish to list the set of L most probable explanations and their corresponding probabilities.

Most probable marginal explanation For an individual 'time' t, what is the distribution over possible explanations:

$$p(h_t|v_{1:T})$$

Using this we may for example find the most likely marginal explanation $\arg\max_{h_t} p(h_t|v_{1:T})$. Note that this not necessarily the same as the h^*_t.

For an $H \times H$ transition matrix A and $V \times H$ emission matrix B, the computational complexity of finding the (single) most likely explanation and the most probable marginal explanation are both $O\left(TVH^2\right)$. The classical algorithms are described for example in [2].

2.1 Looking for a single pattern

When searching for a pattern there are some computational simplifications. For example let's imagine that we wish to search for the single pattern `ACGT` in the observation `GTTAGGTC`. To do this we first describe the generating mechanism for observing noisy versions of the pattern. Here we will use $H = 6$ hidden states and show below their corresponding meaning:

$h = 1$	`A`	
$h = 2$	`C`	
$h = 3$	`G`	
$h = 4$	`T`	
$h = 5$	`notstart`	we have not yet reached the pattern
$h = 6$	`end`	we have finished the pattern

We then define the transition matrix

$$A = \begin{array}{c|cccccc} & [1] & [2] & [3] & [4] & [5] & [6] \\ [1] & 0 & 0 & 0 & 0 & 0.5 & 0 \\ [2] & 1 & 0 & 0 & 0 & 0 & 0 \\ [3] & 0 & 1 & 0 & 0 & 0 & 0 \\ [4] & 0 & 0 & 1 & 0 & 0 & 0 \\ [5] & 0 & 0 & 0 & 0 & 0.5 & 0 \\ [6] & 0 & 0 & 0 & 1 & 0 & 1 \end{array}$$

where $[i]$ indicates state i of the hidden variable. This transition means that if we are in the `notstart` state, we remain in the start state with probability 0.5 or move to the first letter in the pattern. When

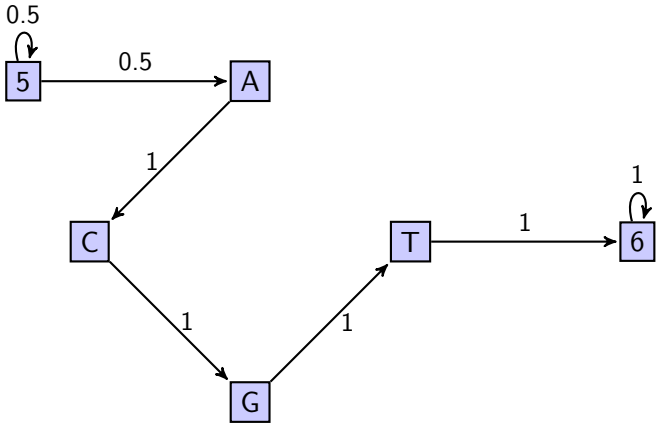

Figure 2: State transition diagram for hidden variable $h_{t-1} \to h_t$ for the `ACGT` pattern.

h is in a pattern letter it deterministically moves to the next letter in the pattern until when it reaches the last letter, after which it deterministically transitions to the `end` state and remains there. This can be represented graphically using a state transition diagram, as in figure(2).

To represent the emission model we can use

$$B = \begin{array}{c} \\ \{1\} \\ \{2\} \\ \{3\} \\ \{4\} \end{array} \begin{array}{c} \begin{array}{cccccc} [1] & [2] & [3] & [4] & [5] & [6] \end{array} \\ \begin{pmatrix} 0.7 & 0.1 & 0.1 & 0.1 & 0.25 & 0.25 \\ 0.1 & 0.7 & 0.1 & 0.1 & 0.25 & 0.25 \\ 0.1 & 0.1 & 0.7 & 0.1 & 0.25 & 0.25 \\ 0.1 & 0.1 & 0.1 & 0.7 & 0.25 & 0.25 \end{pmatrix} \end{array}$$

which says that if we are in the `notstart` state we emit any of `A,C,G,T` with equal probability, and similarly for the `end` state. When we are in a pattern state, we emit the clean pattern with probability 0.7 and one of the other patterns with equal probability 0.1. Here we used the notation $\{i\}$ to indicate state i of the observation variable.

Setting

$$\pi = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

States that with probability 1, the first hidden state is the `notstart` state.

Calling the `HMMviterbi` routine with these settings for the observation `GTTAGGTC`, which corresponds to $v =$ `[3 4 4 1 3 3 4 2]`, we get the most likely explanation is

$$h^* = [5\ 5\ 5\ 1\ 2\ 3\ 4\ 6]$$

which has log probability -10.9972. For $p(h_t|v_{1:T})$ we have the matrix

```
0.0000    0.0108    0.0377    0.9245    0.0013    0.0017    0.0147    0.0026
0.0000    0.0000    0.0108    0.0377    0.9245    0.0013    0.0017    0.0147
0.0000    0.0000    0.0000    0.0108    0.0377    0.9245    0.0013    0.0017
0.0000    0.0000    0.0000    0.0000    0.0108    0.0377    0.9245    0.0013
1.0000    0.9892    0.9515    0.0270    0.0256    0.0240    0.0092    0.0066
0.0000    0.0000    0.0000    0.0000    0.0000    0.0108    0.0485    0.9730
```

in which the t^{th} column represents the probability $p(h_t|v_{1:8})$. Each column represents the distribution over the explanations for time corresponding to that column. In this case, the most likely marginal explanation across time is

$$\bar{h}^* = [5\ 5\ 5\ 1\ 2\ 3\ 4\ 6]$$

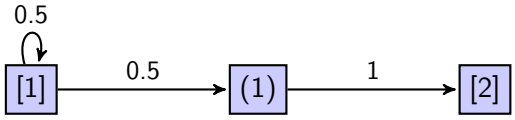

Figure 3: State transition diagram for hidden variable $h_{t-1} \to h_t$ for the pattern state (1) – `ACGT` and two general states [1] – `notstart`, and [2] – `end`.

which matches the most likely joint explanation h^* in this case, though this will not generally be so. See `demoACGT.m` for example code.

2.2 General and Pattern States

We can represent the above example more compactly by defining one pattern and two general states. For

(1)	`ACGT`	
[1]	`notstart`	we have not yet reached the pattern
[2]	`end`	we have finished the pattern

where (i) denotes the i^{th} pattern-state. We then define the generalised transition matrix

$$\tilde{A} = \begin{array}{c|ccc} & (1) & [1] & [2] \\ (1) & 0 & 0.5 & 0 \\ [1] & 0 & 0.5 & 0 \\ [2] & 1 & 0 & 1 \end{array}$$

This transition means that if we are in the `notstart` state, we remain in the start state with probability 0.5 or move to the first letter in the pattern. When h is in a pattern letter it deterministically moves to the next letter in the pattern until when it reaches the last letter, after which it deterministically transitions to the `end` state and remains there. This can be represented graphically using the generalised state transition diagram, as in figure(3).

To represent the emission model we first specify the emission for the generalstates:

$$B_{\text{general}} = \begin{array}{c|cc} & [1] & [2] \\ \{1\} & 0.25 & 0.25 \\ \{2\} & 0.25 & 0.25 \\ \{3\} & 0.25 & 0.25 \\ \{4\} & 0.25 & 0.25 \end{array}$$

A simple way to complete the emission for the patterns is to specify a corruption matrix, for example

$$B_{\text{pattern}} = \begin{array}{c|cccc} & A & C & G & T \\ A & 0.7 & 0 & 0.2 & 0.1 \\ C & 0.1 & 0.9 & 0 & 0.1 \\ G & 0.1 & 0.1 & 0.7 & 0 \\ T & 0.1 & 0 & 0.1 & 0.8 \end{array}$$

A more complex but more powerful alternative is to specify for example

```
Bpattern{1,4}=ones(4,1)/4;
```

which would say that for pattern 1, the 4^{th} position of the pattern has uniform emission distribution. This allows for full flexibility meaning that the emission distribution can be pattern dependent. This requires creating a cell array with entries `Bpattern{pattern,patternposition}` that are probability vectors describing the emission distribution for each of the patterns and the position in that pattern.

If the routine `patternsearchsetup.m` is called with `Bpattern` as a matrix, it recognises this as a corruption probability matrix; otherwise it assumes `Bpattern` is a cell array of probability vectors. See `demoFirstname.m` for an example of both approaches.

2.2.1 Standard states

The software converts the pattern and general states transition $\tilde{A}$ to a full (sparse) transition A based on the following. Beginning with patternstate(1) it defines L_1 states to this pattern, and then moves to patternstate(2), defining states $L_1 + 1, \ldots, L_1 + L_2$, *etc.* until all patterns have been considered. Then the generalstates are appended, making a total of

$$H = \sum_i L_i + G$$

standard hidden states.

The routine `patternsearchsetup.m` returns the standard states and the standard transition A, emission B and prior π.

```
[A,B,standardprior,startpatternIDX,endpatternIDX,generalstateIDX]...
      =patternsearchsetup(pattern,patternstate,generalstate,Atilde,Bpattern,Bgeneral,prior);
```

Conversely,

```
[patternstate generalstate]=getpattern(standardstate,startpatternIDX,generalstateIDX);
```

returns the index of the patternstate or generalstate that the standardstate corresponds to. See below and `demoFirstname.m` for an example.

2.3 A dictionary of patterns

We can use the idea of pattern and general states to do more complex things. If we have a dictionary of patterns we can assign a pattern state to each:

(1)	`david`	
(2)	`anton`	
(3)	`fred`	
(4)	`jim`	
(5)	`barry`	
[1]	`notstart`	we have not yet reached a pattern
[2]	`end`	we have finished all patterns

Running `demoFirstname.m` an example output is

```
observed sequence: barezwztrq
Viterbi sequence corresponds to
observation: [patternstate generalstate]

b:  [0 1]
a:  [3 0] fred
r:  [3 0] fred
e:  [3 0] fred
z:  [3 0] fred
w:  [0 2]
z:  [0 2]
t:  [0 2]
r:  [0 2]
q:  [0 2]
```

This says that the most likely explanation is that pattern 3 `fred` starts at position 2. In the above printout, the generalstate being 0 indicates that a pattern state is more probable, and vice versa.

In `demoFirstname.m` this is achieved by defining the generalised transition $\tilde{A}$.

The emission distribution for the g^{th} generalstate is specified using `Bgeneral(:,g)`.

There are two alternatives described for the `Bpattern` emission. The simplest defines a corruption matrix that states that with 90% probability the correct letter is emitted, otherwise another letter is emitted

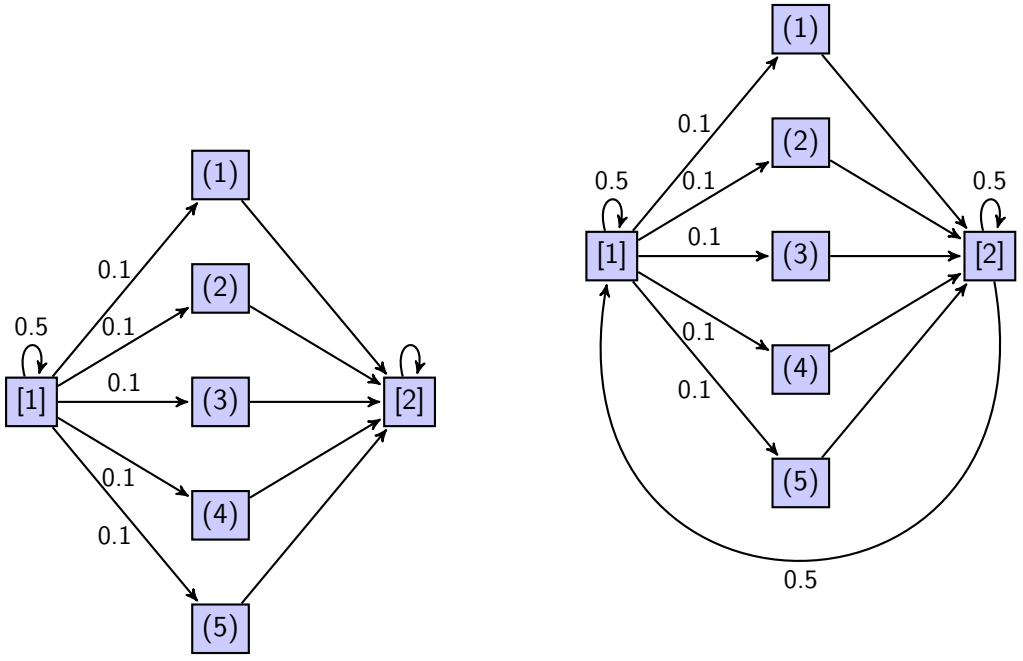

Figure 4: State transition diagram for hidden variable $h_{t-1} \rightarrow h_t$ for a set of 5 pattern states and two general states `notstart, end`. For deterministic transition (probability 1), no probability is marked. (a) Only a single pattern from the dictionary will be matched once. (b) Possibly more than one pattern will be matched.

uniformly at random. The more complex construction is to use `Bpattern{pt,l}` which contains the emission distribution

$$p(v|h = \{\text{pattern is pt and we are in the position l in the pattern}\})$$

This allows one to have different emission probabilities depending on the position in the pattern. For example, it might be that for the pattern `fred` we wish to have a different emission distribution $p(v|\texttt{f})$ than for $p(v|\texttt{d})$. This allows some flexibility in defining the emission and follows the encoding scheme in section(2.2.1).

The prior is defined one of either

`prior`	contains the distribution over all H states.
`prior.patternstate`	contains the distribution over which patternstates one should begin in.
`prior.generalstate`	contains the distribution over which generalstates one should begin in.

This allows for some flexibility in defining the prior.

If we wish to search for the occurrence of more than one pattern in the observation sequence, we can add a link back to the general `notstart` state, see figure(4b). In this case we can find search for the existence of patterns (possibly recurring) in a observation sequence. For the emission matrix B, we can define for example a 26×26 matrix that is diagonally dominant in that the probability of emitting state i when the hidden state is i is higher than emitting any other probability.

2.4 More complex search

Consider two dictionaries, one containing first names as above and the other surnames

```
barber
ilsung
fox
chain
fitzwilliam
quinceadams
grafvonunterhosen
```

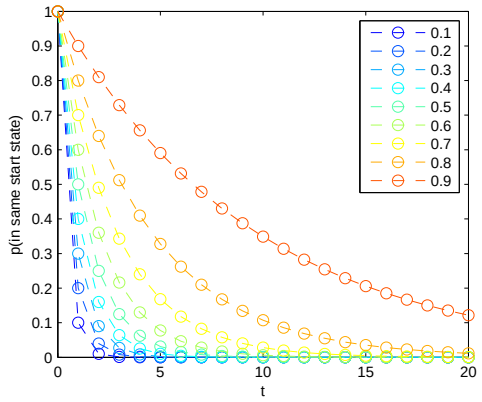

Figure 5: The probability of remaining in the same state p^t for t timesteps, where p is the probability of remaining in the same state for a single timestep. This probability decreases exponentially (geometrically) with time so that, for example, for $p = 0.5$, the probability that we remain in the same state after 4 timesteps is about 6%.

and that we wish to search for patterns of the form `firstname*surname` where $*$ represents any length≥ 1 sequence. One way to do this is to consider

(1)	`david`	
(2)	`anton`	
(3)	`fred`	
(4)	`jim`	
(5)	`barry`	
(6)	`barber`	
(7)	`ilsung`	
(8)	`fox`	
(9)	`chain`	
(10)	`fitzwilliam`	
(11)	`quinceadams`	
(12)	`grafvonunterhosen`	
[1]	`notstartfirstname`	we have not yet reached a firstname
[2]	`notstartsurname`	we have finished a firstname but not started surname
[3]	`end`	we have finished all patterns

and define appropriate transition probabilities, see `demoFirstnameSurname.m`. For example from $p([2] \rightarrow [2]) = 0.5$ and $p([2] \rightarrow \{(6-12)\}) = 1/7$ would represent remaining in the `notstartsurname` with probability 0.5 or transitioning to a surname with probability 0.5. Note that this means that the number of observations between the end of the firstname and the start of the surname is geometric distribution, so that the generating mechanism would place very low probability to long after firstname – before surname periods. See the discussion also below.

2.5 Self-transition probability

Consider a two state Markov chain with probability of remaining in the same state given by p. Then in t timesteps, the probability that we are still in the same state is p^t which decays exponentially quickly with t, see figure(5). One can address this by using an explicit duration model in which we define a set of duration patterns, each of different length and define transitioning into a duration pattern with probability p_i. In this way, at the expense of additional states, one can define durations between states of any length and probability.

2.6 Computational Complexity

For a length K pattern, this can be represented by a single pattern state, which itself corresponds to K actual states. The fact that these actual states transition deterministically to the next internal pattern state means that the computational expense of inference in the HMM is reduced. For a set of G general patternstates , V observation states and dictionary of P patterns with pattern i of length L_i, the complexity of inferring the most likely explanation or the marginal explanation is order

$$TV\left(G^2 + GP + \sum_{i=1}^{P} L_i\right)$$

In practice, typically the transition matrix will be even sparser, so that this is an upper bound on the order of complexity. More generally, the computational complexity is of order

$$T\left(\sum_{ij}\mathbb{I}\left[B_{ij}\neq 0\right]+\sum_{ij}\mathbb{I}\left[A_{ij}\neq 0\right]\right)$$

2.6.1 Speeding things up

Inference in the HMM corresponds to passing non-negative messages along a chain [2]. A simple approximation is to only retain messages that have a numerical value above some threshold. This makes the messages sparse, at the potential cost of not computing the inference exactly.

References

[1] A. V. Aho and M. J. Corasick. Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[2] D. Barber. *Bayesian Reasoning and Machine Learning.* Cambridge University Press, 2012.