1 Review from last week

Last week we discussed Markov-Models and Hidden-Markov-Models, and also the Forward algorithm. Let's review the basic setting:

1.1 HMM

We have a finite hidden Markov-chain of states: $S_i$, which we are not exposed to, and each state emits some observable output: $X_i$. We assume the following dependencies (or actually, independencies): If we are given $S_{i-1}$, then $S_i$ does not depend on any previous history before $S_{i-1}$ (that's the Markov property) and if we are given $S_i$ then $X_i$ is independent of everything else.

It can be visualized in a dependency graph:

![Figure 1: Dependency graph of all random variables](image)

So we have the following set of parameters which define our model:

$$\tau_{k,l} = P(S_i = l | S_{i-1} = k)$$
$$e_l(x) = P(X_i = x | S_i = l)$$

Here $\tau$ stands for 'transitions', and $e$ stands for 'emissions'.

If we denote the number of possible states that each $S_i$ can be in, by $N$ (in the CpG Island example $N = 2$, because either we are in a CpG Island or we are not, and in the Croupier example we also have $N = 2$, because either
the current dice is \textit{Fair} or it is \textit{Loaded}) then \( \tau \) is a \( N \times N \) matrix, where each row sums up to 1, and \( \tau_{k,l} \) is the probability that the next state will be \( l \) given that the current state is \( k \).

And if we denote the size of the alphabet of possible emissions by \( M \) (in the CpG Island example \( M = 4 \), because the possible emissions belong to \{\text{A, C, G, T}\}, and in the Croupier example \( M = 6 \), because the possible emissions belong to \{1, 2, 3, 4, 5, 6\}), then \( e \) is a matrix of size \( N \times M \), where again each row sums to 1, and \( e_i(x) \) is the probability that we will observe an output \( x \) given that we are in state \( l \).

### 1.2 The Forward Algorithm

The motivation is the following:

We are given the emitted sequence and the state sequence. We are given the parameters: emissions and transitions, and we are asking - what is the probability of a certain specific sequence of states and emissions?

In other words - what is their joint distribution:

\[
P(\vec{s} = \vec{s}, \vec{X} = \vec{x}) = p_0(s_1) \cdot e_{s_1}(x_1) \prod_{i=2}^{T} \tau_{s_{i-1},s_i} e_{s_i}(x_i)
\]

(1)

The formula above follows from the chain rule and from our assumptions about the dependencies between the states and the emissions. Here is the derivation:

\[
P(\vec{s} = \vec{s}, \vec{X} = \vec{x}) =
\]

\[
P(S_1 = s_1) \cdot P(S_2 = s_2 | S_1 = s_1) \cdot \ldots \cdot P(S_i = s_i | S_{i-1} = s_{i-1}) \cdot \ldots \cdot P(S_T = s_T | S_{T-1} = s_{T-1}) \cdot
\]

\[
P(x_1 = x_1 | \vec{S} = \vec{s}) \cdot P(x_2 = x_2 | x_1 = s_1, \vec{s}) \ldots \cdot P(x_i = x_i | x_{i-1} = \vec{x}, \vec{S} = \vec{s}) \cdot \ldots
\]

\[
P(x_T = x_T | x_{1:T-1} = \vec{x}, \vec{S} = \vec{s})
\]

From the dependencies assumptions we have:

\[
= p_0(s_1) \cdot P(S_2 = s_2 | S_1 = s_1) \cdot \ldots \cdot P(S_i = s_i | S_{i-1} = s_{i-1}) \cdot \ldots \cdot P(S_T = s_T | S_{T-1} = s_{T-1}) \cdot
\]

\[
P(x_1 = x_1 | s_1) \cdot P(x_2 = x_2 | s_2) \cdot \ldots \cdot P(x_i = x_i | s_i) \cdot \ldots \cdot P(x_T = x_T | s_T)
\]

Let’s denote the initial probability to be in state \( k \) by \( p_0(k) \). Now, we can write this long product in the following form:

\[
= p_0(s_1) \cdot P(S_2 = s_2 | S_1 = s_1) \cdot \ldots \cdot P(S_i = s_i | S_{i-1} = s_{i-1}) \cdot \ldots \cdot P(S_T = s_T | S_{T-1} = s_{T-1}) \cdot
\]

\[
P(x_1 = x_1 | s_1) \cdot P(x_2 = x_2 | s_2) \cdot \ldots \cdot P(x_i = x_i | s_i) \cdot \ldots \cdot P(x_T = x_T | s_T)
\]

And this exactly the right hand side of Eq. (1).

So we can easily compute in linear time the joint probability.

The problem is that the \( s_i \) are hidden. We are not exposed to them, and we have only the emissions output. So we would like to compute the probability of an output sequence: \( P(\vec{X} = \vec{x}) \). Now, according to the law of total probability we have:

\[
P(\vec{X} = \vec{x}) = \sum_{\vec{s}} P(\vec{s} = \vec{s}, \vec{X} = \vec{x})
\]

(2)

But now another problem occurs - we have exponential number of terms in this sum! And here comes the Forward algorithm which helps us to compute this stuff efficiently. Last week we defined:

\[
F_k(i) = P(X_1 = x_1, \ldots, X_i = x_i, S_i = k)
\]

(3)

And according to the law of total probability this is the sum over all possible ‘prefix-histories’ \( s_1, \ldots, s_{i-1} \) which end with \( s_i = k \) of the probabilities that they emitted \( x_1, \ldots, x_i \) multiplied by the probabilities of the transitions.

The 'Forward' table is of size \((N + 1) \times (T + 1)\), and the table is dynamically filled according to the formula:

\[
F_k(i) = \left( \sum_l F_l(i-1) \tau_{l,k} \right) e_k(x_i)
\]

(4)
There are a few technical details - we added a special state - the '0 state' which comes before the Markov chain, and from which the Markov chain must start. That’s why in the initialization we filled the first column with $F_0(0) := 1$ and $\forall k > 0, F_k(0) := 0$.

We then use the same formula (4), for the initialization of the column $i = 1$, because it actually means:

$$F_k(1) = \left( \sum_{j=0}^{N} F_l(0) \tau_{l,k} \right) e_k(x_1) = (F_0(0) \tau_{0,k}) e_k(x_1) = \tau_{0,k} e_k(x_1)$$ (5)

Here $\tau_{0,k}$ (the probability to pass from the initial '0' state to $k$), is actually the probability $P(S_1 = k)$, which we called $p_0(k)$ before we added this initial state '0'. (So that’s why in the pseudo-code from last week, we used a $(N + 1) \times (T + 1)$ Forward table: We added additional state, and it increased the number of states from $N$ to $N + 1$, and the length of the whole sequence increased from $T$ to $T + 1$, because we forced our sequence to start from this state)

You can see the pseudo-code of the algorithm in Algorithm (1). The time-complexity of this algorithm is $O(N^2T)$ since we are filling a table of size $O(NT)$, and for the computation of each cell we need to do $O(N)$ work.

The Markov property enables us to sum over an exponential space of variants, using dynamic programing. The Markov property is also reflected in the fact that when filling the $i$-th column in the table - we are looking only on the previous $i - 1$ column. I.e. - We don’t need to remember what happened earlier.

---

**Algorithm 1**

Forward: Calculate $P(X_1 = x_1, ..., X_T = x_T)$

1: **Initialize:** $F_0(0) := 1$, $\forall k > 0, F_k(0) := 0$

2: **for** $i = 1...T$ **do**

3:  **for** $k = 0...N$ **do**

4:  $F_k(i) = \left( \sum_{l=0}^{N} F_l(i - 1) \tau_{l,k} \right) e_k(x_i)$

5:  **end for**

6: **end for**

7: $P(\vec{X} = \vec{x}) = \sum_{k=0}^{N} F_k(T)$

---

![Forward Dynamic Table](image)

**Figure 2:** The Forward Dynamic Table - when computing $F_k(j)$ we need to sum other all possible histories of states. And the table does that for us, So we just need to look at the previous column.
It’s worth mentioning that if our dependency graph looked differently, then it is possible that we had to look more into the past - i.e. to look at more than 1 column before the current column.

For instance if in our hidden Markov chain, each state was depending on its 2 preceeding states (i.e. $S_i$ depends both on $S_{i-1}$ and on $S_{i-2}$) then we will have to look more than 1 column backward, and we will need to modify the subproblems of the dynamic algorithm correspondingly.

But if we don’t change the dependencies of the states, and for instance change the dependencies of the emissions, the same dynamic algorithm will still work, but now $e_k(x)$ will just depend on more things than before. And we will need to update the formula correspondingly.

It’s also worth mentioning that in the 80’s - Judea Pearl defined a very important concept- Markov Blanket: The Markov Blanket of a given random variable is the minimal set of other variables that our variable depends on, and that given the values of those variables - our variable will be independent of anything else.

So now we finished the review of the material from last week - and we will talk about the Backward algorithm.

Figure 3: Variation1: Each emission depends on the current and the previous state, so we will have the flowing formula: $F_k(i) = \sum_l F_l(i-1) \tau_{l,k} e_{l,k}(x_i)$. Variation2: Each emission depends on the current state and the previous emission, so we will have the following formula $F_k(i) = (\sum_l F_l(i-1) \tau_{l,k}) e_{k,x_{i-1}}(x_i)$

2 The Backward Algorithm

Lets return to the Croupier example. We are given the sequence of the dice emissions, and we want to estimate the probability that the croupier has been cheating in the 38-th dice rolling. So we want to answer the question - did he use the Fair dice or the Loaded dice in the 38-th dice rolling?

Or in other words we want to estimate the posterior probability that the 38-th dice rolling was Fair, vs. the posterior
probability that this dice rolling was loaded:

\[ P(S_i = \text{Fair} | \vec{X} = \vec{x}) \quad \text{vs.} \quad P(S_i = \text{Loaded} | \vec{X} = \vec{x}) \]

For this purpose we need to be able to compute the probability of a given sequence of emissions, i.e. we need to be able to compute: \( P(\vec{X} = \vec{x}) \). And we already know how to do it! According to the law of total probability we have:

\[ P(\vec{X} = \vec{x}) = \sum_k P(\vec{X} = \vec{x}, S_T = k) = \sum_k F_k(T) \quad (6) \]

(Here \( T \) is the length of the sequence.)

And now we will use the Backward algorithm to compute the rest of the information. In this algorithm we will compute the probabilities of future emissions, conditioned on the current state. Let’s define:

\[ B_k(i) := P(X_{i+1} = x_{i+1}, ..., X_T = x_T | S_i = k) \quad (7) \]

\( B_k(i) \) is the probability of the rest of future observations, given that we are now in state \( k \).

It is important to note - that the \( F_k(i) \) was the joint probability of \( X_1, ..., X_i, S_i \), and now \( B_k(i) \) is NOT the joint of \( X_{i+1}, ..., X_T, S_i \), but rather the conditional probability of \( X_{i+1}, ..., X_T \) given \( S_i \).

So now - before we get into the algorithm, let’s answer the question about the 38-th dice-rolling. How do we know if it was Fair or Loaded?

\[ P(S_i = k | \vec{X} = \vec{x}) = \frac{P(S_i = k, \vec{X} = \vec{x})}{P(\vec{X} = \vec{x})} \quad (8) \]

We need to find: \( P(S_i = k, \vec{X} = \vec{x}) \).

Now, according to the chain rule, we have:

\[ P(S_i = k, \vec{X} = \vec{x}) = P(X_{i+1} = x_{i+1}, ..., X_T = x_T | S_i = k, X_1 = x_1, ..., X_i = x_i) \cdot P(S_i = k, X_1 = x_1, ..., X_i = x_i) \]

And since we know that given \( S_i \), all the later emissions are independent of the previous emissions, we get the following:

\[ = P(X_{i+1} = x_{i+1}, ..., X_T = x_T | S_i = k) \cdot P(S_i = k, X_1 = x_1, ..., X_i = x_i) = B_k(i) \cdot F_k(i) \]

In other words:

\[ P(S_i = k, \vec{X} = \vec{x}) = B_k(i) \cdot F_k(i) \quad (9) \]

And if we want this formula to hold in the edge-case where \( i = T \), then we need to define \( \forall k, B_k(T) = 1 \). (And actually it makes sense, since \( B_k(i) \) is the conditional probability of some intersection of events conditioned on \( S_i \), we will get that \( B_k(T) \) will be the conditional probability of an EMPTY intersection of events, conditioned on \( S_T \). And there is a convention in mathematics that an empty intersection equals by definition to the whole space. So it makes sense to define \( B_k(T) \) to be 1, and it will perfectly match into our recursive formula)

It’s worth mentioning, that the initialization of the last column can be different - in case the croupier knows for example - that in the end the dice will be checked by the inspector. Thus some places in the last column may be 0 in this case.

So we have got the following result:

\[ P(S_i = k | \vec{X} = \vec{x}) = \frac{P(S_i = k, \vec{X} = \vec{x})}{P(\vec{X} = \vec{x})} = \frac{F_k(i) \cdot B_k(i)}{\sum_l F_l(T)} \quad (10) \]

With this we can answer the question about the probability that in a specific place in the sequence the croupier used the Loaded dice.

Another question we can ask- is what is the probability that the croupier has swiched the dice from Fair to Loaded exactly between the \( i \)-th rolling and the \( i + 1 \)-th rolling. In other words we want to estimate

\[ P(S_i = k, S_{i+1} = l | \vec{X} = \vec{x}) \]
And actually we already know all what we need:

$$P(S_i = k, S_{i+1} = l | \vec{X} = \vec{x}) = \frac{P(S_i = k, S_{i+1} = l, \vec{X} = \vec{x})}{P(\vec{X} = \vec{x})}$$  \hspace{1cm} (11)$$

And from the chain rule we have that

$$P(S_i = k, S_{i+1} = l, \vec{X} = \vec{x}) = \prod_{i=1}^{T} P(X_i = x_i, ..., X_T = x_T | X_1 = x_1, ..., X_{i+1} = x_{i+1}, S_i = k, S_{i+1} = l)$$

And again - because of our HMM assumptions - the above equals to:

$$= \prod_{i=1}^{T} P(X_i = x_i, ..., X_T = x_T | X_1 = x_1, ..., X_{i+1} = x_{i+1}, S_i = k, S_{i+1} = l) = F_k(i) \cdot \tau_{k,l} \cdot e_l(x_{i+1}) \cdot B_l(i+1)$$

Hence we have:

$$P(S_i = k, S_{i+1} = l | \vec{X} = \vec{x}) = \frac{F_k(i) \cdot \tau_{k,l} \cdot e_l(x_{i+1}) \cdot B_l(i+1)}{\sum_j F_j(T)}$$  \hspace{1cm} (12)$$

In this way we can answer the question about the probability that at some specific moment - between \(i\) and \(i+1\) the croupier has switched the dice from Fair to Loaded.

Now we need to show how can we efficiently compute the values of \(B_k(i)\) \((1 \leq k \leq N, 1 \leq i \leq T)\). As before - we will use the law of total probability, and we will sum over all possible future paths \(S_{i+1}, ..., S_T\) which start from \(S_i = k\), and which emit \(x_{i+1}, ..., x_T\). For each such future path, we will sum the product of the transition probabilities multiplied by the emission probabilities. The problem: we need to sum over an exponential number of variants. The solution: as before we will use dynamic programing, and the Markov property will help us. But now we are going to fill the table in the opposite direction!

![Backward Dynamic Table](image)

Figure 4: The Backward Dynamic Table - when computing \(B_k(j)\) we need to sum other all possible future paths of states. And the table does that for us, so we just need to look at the next column.
So now - we want to somehow use $B_k(i)$ for the computation of $B_l(i - 1)$. And again we will use the law of total probability:

$$B_l(i - 1) = P(X_i = x_i, ..., X_T = x_T|S_{i-1} = l) = \sum_k P(S_i = k, X_i = x_i, ..., X_T = x_T|S_{i-1} = l) =$$

By the chain rule:

$$= \sum_k P(X_i = x_i|S_{i-1} = l, S_i = k, X_{i+1} = x_{i+1}, ..., X_T = x_T) \cdot P(X_{i+1} = x_{i+1}, ..., X_T = x_T|S_{i-1} = l, S_i = k) \cdot P(S_i = k|S_{i-1} = l)$$

Which due to our HMM assumptions, equals to:

$$= \sum_k P(X_i = x_i|S_i = k) \cdot P(X_{i+1} = x_{i+1}, ..., X_T = x_T|S_i = k) \cdot P(S_i = k|S_{i-1} = l) =$$

$$= \sum_k e_k(x_i) \cdot B_k(i) \cdot \tau_{l,k} \quad \text{(13)}$$

**Algorithm 2 Backward:**

1: **Initialize:** $\forall 1 \leq k \leq N, B_k(T) := 1$
2: **for** $i = T...2$ **do**
3:  **for** $l = 1...N$ **do**
4:     $B_l(i - 1) = \sum_{k=1}^{N} e_k(x_i) \cdot B_k(i) \cdot \tau_{l,k}$
5:  **end for**
6: **end for**

That’s how the Markov property helps us to compute the $B_k(i)$ efficiently.

Now, we saw above in Eq. (10) how to compute the posterior probability of $S_i = k$:

$$P(S_i = k|\vec{X} = \vec{x}) = \frac{P(S_i = k, \vec{X} = \vec{x})}{P(\vec{X} = \vec{x})} = \frac{F_k(i) \cdot B_k(i)}{\sum_l F_l(T)}$$

But what can we say about the prior of $S_i = k$?

If $i = 1$, then this is by definition one of the parameters of our model: it’s just $P_0(k)$, the initial probability to start from state $k$. How do we compute this for $i \geq 2$?

Well, according to the law of total probability, we have:

$$P(S_i = k) = \sum_{s_1, ..., s_{i-1}} P(S_1 = s_1, ..., S_{i-1} = s_{i-1}, S_i = k) =$$

By the chain rule:

$$= \sum_{s_1, ..., s_{i-1}} P(S_1 = s_1) P(S_2 = s_2|S_1 = s_1) \cdots P(S_i = k|S_1 = s_1, ..., S_{i-1} = s_{i-1}) =$$

By the Markov assumption:

$$= \sum_{s_1, ..., s_{i-1}} P(S_1 = s_1) P(S_2 = s_2|S_1 = s_1) \cdots P(S_i = k|S_{i-1} = s_{i-1}) =$$
In our notations: $P_0(a) := P(S_1 = a)$, and $\tau_{a,b} := P(S_i = b | S_{i-1} = a)$, and we can write the above, the following way:

$$P(S_i = k) = \sum_{s_1, \ldots, s_i} P_0(s_1) \tau_{s_1, s_2} \cdots \tau_{s_{i-1}, k}$$

So to summarize - we showed the following:

$$P(S_i = k) = \sum_{s_1, \ldots, s_i} P_0(s_1) \tau_{s_1, s_2} \cdots \tau_{s_{i-1}, k}$$

Now, if we think of $\tau$ as a matrix of size $N \times N$, where $\tau_{a,b}$ is the probability of the transition from state $a$ to state $b$, and if we think of $p_0$ as a row vector, where $p_0[a] = P(S_1 = a)$ Then we can see by induction on $i$, that the following holds:

$$\sum_{s_1, \ldots, s_i} P_0(s_1) \tau_{s_1, s_2} \cdots \tau_{s_{i-1}, k} = (p_0 \tau^{i-1}) [k]$$

Let's prove it by induction on $i$:

**Base case:** $i = 2$: In this case the equation Eq. (15) looks like this:

$$\sum_{s_1} P_0(s_1) \tau_{s_1, s_2} = (P_0(1) \ldots P_0(N)) \begin{pmatrix} \tau_{1,1} \\ \vdots \\ \tau_{N,1} \end{pmatrix} = \begin{pmatrix} P_0(1) & \ldots & P_0(N) \end{pmatrix} \begin{pmatrix} \tau_{1,1} & \cdots & \tau_{1,N} \\ \vdots & \ddots & \vdots \\ \tau_{N,1} & \cdots & \tau_{N,N} \end{pmatrix} \begin{pmatrix} \tau_{1,1} \\ \vdots \\ \tau_{N,1} \end{pmatrix} = (p_0 \tau) [k]$$

**Inductive step:**
Assume that we already know Eq. (15) holds for some $i$, then for $i + 1$ we have:

$$\sum_{s_1, \ldots, s_i} P_0(s_1) \tau_{s_1, s_2} \cdots \tau_{s_{i-1}, k} \tau_{s_{i-1}, s_i} = \sum_{s_i} \tau_{s_i, k} \sum_{s_1, \ldots, s_{i-1}} P_0(s_1) \tau_{s_1, s_2} \cdots \tau_{s_{i-1}, s_i}$$

And according to the inductive assumption for $i$, we get:

$$= \sum_{s_i} \tau_{s_i, k} \cdot (p_0 \tau^{i-1}) [s_i] =$$

$$= (p_0 \tau^{i-1}) [1] \ldots (p_0 \tau^{i-1}) [N] \begin{pmatrix} \tau_{1,1} \\ \vdots \\ \tau_{N,1} \end{pmatrix} =$$

$$= \begin{pmatrix} (p_0 \tau^{i-1}) [1] & \ldots & (p_0 \tau^{i-1}) [N] \end{pmatrix} \begin{pmatrix} \tau_{1,1} & \cdots & \tau_{1,N} \\ \vdots & \ddots & \vdots \\ \tau_{N,1} & \cdots & \tau_{N,N} \end{pmatrix} [k] = ((p_0 \tau^{i-1}) \tau) [k] = (p_0 \tau^i) [k]$$

As we wanted.

So to summarize, we have the following result:

$$P(S_i = k) = (p_0 \tau^{i-1}) [k]$$

In other words- according to the Markov property, the prior of $S_i = k$ can be computed by raising the $\tau$ matrix to the $i - 1$-th power, and then multiplying the initial probability $p_0$ row vector, by this matrix from the right, and then taking the $k$-th place in the resulting row vector.
Figure 5: The TBP protein. It’s binding region is of length 6.

3 Motif Finding using HMMS

So where do we use HMMs? One example is Motif finding in the DNA:

TBP - TATA Binding Protein: This protein helps in DNA transcription, and it has some statistical preferences in its binding to the DNA. There are sequences in the DNA which are more likely to bind with it, and there are sequences which are less likely.

Here we can use the HMM model: we have an observable sequence - the DNA itself, and we have some hidden information, because we don’t know if the current position in the DNA is a TBP binding site or not. And since the binding region of the TBP is of length 6, we can model the problem using an HMM with 7 hidden states: 0,1,...,6.

6 of them will enumerate positions within the binding region, and the 7-th will denote a position which is outside the binding region.

Hence our model will have the following parameters: PWM - probabilistic weight matrix - that’s the emissions matrix. It is of size $6 \times 4$, and for each $1 \leq i \leq 6$ and for each $Letter \in \{A, C, G, T\}$.

$P_i(Letter) =$ the probability to see $Letter$ in the current place, given that this is the $i$-th place of the binding.

And we also have $P_0(Letter) = P(Letter|\text{TBP doesn't bind in the region of the DNA})$

And since there are so many sequences which the TBP doesn’t bind to, that the last distribution is almost more or less equal to the genome distribution.

And we have an additional parameter $\alpha$ - which corresponds to the transition probability from the 0-state ("outside binding sequence") to the 1-st state ("1-st place in binding sequence"). This can be seen in Figure (6). And this is
the only parameter of the hidden Markov chain, because after the chain has entered state 1 - it must proceed to states 2, 3, 4, 5, 6 and then 0, with probability 1.

The following is an empirical estimation of the emission probabilities:

<table>
<thead>
<tr>
<th>PWM</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>0.8</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>0.9</td>
</tr>
</tbody>
</table>

Figure 7: The emission probabilities.

It can be seen that this protein really likes the TATAAT sequence, but it can bind with other sequences too.