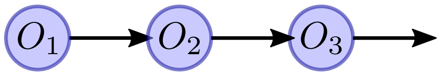
Hidden Markov Models

Justus Piater

Table of Contents

1. Sequential Models

1.1. Markov Chain



$$\begin{aligned} p(O_1, \ldots, O_N) &= p(O_1) \prod_{n=2}^{N} p(O_n \mid O_{n-1}) \\ p(O_n \mid O_{n-1}, \ldots, O_1) &= p(O_n \mid O_{n-1}) \end{aligned}$$

The second equation is an instance of the ***Markov property*** (probabilistic dependencies confined to finite neighborhoods) and here follows from *d-separation*.

To model a ***stationary*** sequence, use a ***homogeneous*** Markov chain where $p(O_n \mid O_{n-1})$ is the same for all n.

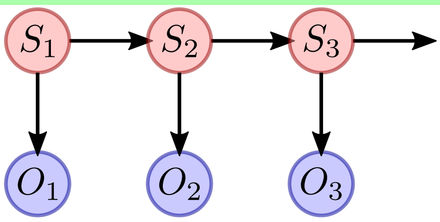
Due to their strong conditional-independence assumptions, such sequential models are severely limited in their expressive power.

We can also define *higher-order* Markov chains. For example, in a second-order Markov chain, $p(O_n \mid O_{n-1}, \ldots, O_1) = p(O_n \mid O_{n-1}, O_{n-2})$.

The number of parameters of an Mth-order Markov chain is exponential in M.

1.2. State-Space Models

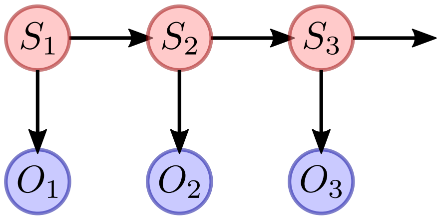
To add expressive power while retaining computational tractability, introduce latent variables:



$$\begin{aligned} p(O_1, \ldots, O_N, S_1, \ldots, S_N) &= p(S_1) \left(\prod_{n=2}^{N} p(S_n \mid S_{n-1}) \right) \prod_{n=1}^{N} p(O_n \mid S_n) \\ p(S_n \mid S_{n-1}, \ldots, S_1) &= p(S_n \mid S_{n-1}) \\ p(O_n \mid O_{n-1}, \ldots, O_1) &\neq p(O_n \mid O_{n-1}) \end{aligned}$$

The ***latent*** (never observed) variables S exhibit the Markov property, but – as long as none of the latents is observed – $p(O_n \mid O_{n-1}, \ldots, O_1)$ depends on its entire history of ***observations***.

1.3. Hidden Markov Model



A ***Hidden Markov Model*** is a state-space model of the above structure with ***discrete*** latent (***state***) variables S and (discrete or continuous) observation variables O.

Applications:

All kinds of sequential (e.g., temporal) data, e.g.:

- speech recognition

  sub-words, words, syntax; possibly combined in stacked HMMs [Rabiner 1989 Sec. VIII]

- handwritten character recognition

  sequences of strokes at different orientations [Bishop 2006 Sec. 13.2]

- genetics

Where large amounts of training data are available, HMMs have mostly fallen out of fashion in favor of neural nets. However, the underlying principle of state-space models is still extremely popular, in neural networks as well as in purely-probabilistic models.

Rabiner [Rabiner 1989] defines three *basic problems* for HMM:

1. Compute the probability $p(\mathbf{o})$ of an observation sequence $\mathbf{o} = o_1, \ldots, o_N$ (given the model parameters θ).

2. Given an observation sequence $\mathbf{o}$ (and θ), compute the state sequence that maximizes $p(s_1, \ldots, s_N \mid \mathbf{o})$.

3. Given one or more observation sequences $\mathbf{o}$, compute the model parameters that maximize $p(\mathbf{o} \mid \theta)$.

2. Problem 1: Probability of an Observation Sequence

2.1. The Forward-Backward Algorithm

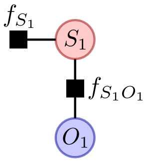
- In principle, we need to marginalize $p(\mathbf{o} \mid S_1, \ldots, S_N)$ over all possible state sequences, whose number is exponential in N.

- The ***Forward-Backward*** algorihm for HMM solves this in time proportional in N.

- It is equivalent to the *Sum-Product* algorithm in that $p(\mathbf{o}) = Z$, which is the normalization constant that turns message products into marginal probabilities.

  This is shown in the following sections.

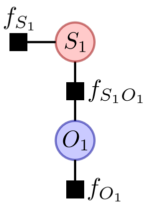
2.2. First Time Slice, Without Observation



$$
\begin{aligned}
f_{S_1} &= p(S_1) \\
\mu_{f_{S_1} \to S_1} &= p(S_1) \\
f_{S_1 O_1} &= p(O_1 \mid S_1) \\
\mu_{f_{S_1 O_1} \to S_1} &= \sum_o p(O_1 = o \mid S_1) \;=\; \mathbf{1} \\
\mu_{f_{S_1} \to S_1} \mu_{f_{S_1 O_1} \to S_1} &= p(S_1) \\
Z &= 1
\end{aligned}
$$

The (element-wise) product of the incoming (vector) messages at S_1 yields the marginal probability distribution $p(S_1)$, with a normalization factor $Z = \sum \mu_{f_{S_1} \to S_1} \mu_{f_{S_1 O_1} \to S_1} = 1$.

2.3. First Time Slice, With Observation

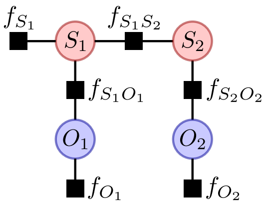


$$
\begin{aligned}
f_{S_1} &= p(S_1) \\
\mu_{f_{S_1} \to S_1} &= p(S_1) \\
f_{S_1 O_1} &= p(O_1 \mid S_1) \\
f_{O_1} &= \text{vector whose element at } o_1 = 1 \text{ and 0 everywhere else} \\
\mu_{f_{S_1 O_1} \to S_1} &= \sum_o p(O_1 = o \mid S_1) f_{O_1} \;=\; p(O_1 = o_1 \mid S_1) \\
\mu_{f_{S_1} \to S_1} \mu_{f_{S_1 O_1} \to S_1} &= p(O_1 = o_1, S_1) \\
Z &= p(O_1 = o_1)
\end{aligned}
$$

The message product at S_1 yields a vector whose elements are the joint probabilities $p(O_1 = o_1, S_1 = s)$ for all values s that S_1 can take. To recover $p(S_1 \mid O_1 = o_1)$ this vector has to be normalized by dividing it by Z, the sum of its elements (as prescribed by the Sum-Product algorithm). Here, this sum is a marginalization of $p(O_1 = o_1, S_1)$ over S_1 such that Z ***equals the marginal probability of the observation***:

$$
p(S_1 \mid O_1 = o_1) \;=\; \frac{p(S_1, O_1 = o_1)}{p(O_1 = o_1)}
$$

2.4. Second Time Slice, With Observations



$$
\begin{aligned}
f_{S_1S_2} &= p(S_2 \mid S_1) \\
\mu_{f_{S_1S_2}\to S_2} &= \sum_s p(O_1 = o_1, S_1 = s)p(S_2 \mid S_1 = s) = p(S_2, O_1 = o_1) \\
\mu_{f_{S_2O_2}\to S_2} &= p(O_2 = o_2 \mid S_2) \\
\mu_{f_{S_1S_2}\to S_2}\mu_{f_{S_2O_2}\to S_2} &= p(O_2 = o_2 \mid S_2)p(O_1 = o_1 \mid S_2)p(S_2) \\
&= p(O_1 = o_1, O_2 = o_2 \mid S_2)p(S_2) \\
&= p(O_1 = o_1, O_2 = o_2, S_2) \\
Z &= p(O_1 = o_1, O_2 = o_2)
\end{aligned}
$$

For the first *emphasized* equality see its derivation.

The second *emphasized* equality holds since $O_1 \perp\!\!\!\perp O_2 \mid S_2$.

To recover $p(S_2 \mid O_1 = o_1, O_2 = o_2)$ from the message product $p(O_1 = o_1, O_2 = o_2, S_2)$, this probability vector has to be divided by its marginalization Z over S_2, i.e., Z ***equals the joint probability of the observation***:

$$
p(S_2 \mid O_1 = o_1, O_2 = o_2) = \frac{p(S_2, O_1 = o_1, O_2 = o_2)}{p(O_1 = o_1, O_2 = o_2)}
$$

2.5. $\mu_{f_{S_1S_2}\to S_2}$

$$
\begin{aligned}
\mu_{f_{S_1S_2}\to S_2} &= \sum_s p(O_1 = o_1, S_1 = s)p(S_2 \mid S_1 = s) \\
&= \sum_s p(O_1 = o_1 \mid S_1 = s)p(S_1 = s)p(S_2 \mid S_1 = s) \\
&= \sum_s p(O_1 = o_1, S_2 \mid S_1 = s)p(S_1 = s) \\
&= \sum_s p(S_1 = s, S_2, O_1 = o_1) \\
&= p(S_2, O_1 = o_1)
\end{aligned}
$$

The third equation holds because $S_2 \perp\!\!\!\perp O_1 \mid S_1$.

2.6. Remark

For the purpose of solving Problem 1, the backward pass is not required. Once S_N has received its messages, Z is available.

3. Problem 2: Compute a MAP State Sequence

3.1. The Viterbi Algorithm

- The *Max-Sum* algorithm finds a most probable state sequence given an observation sequence.
- Its formulation for HMM is known as the ***Viterbi*** algorithm.

4. Problem 3: Compute Model Parameters From Observation Sequences

4.1. The Baum-Welch Algorithm

Chicken-and-egg problem:

- If we know the model parameters $\boldsymbol{\theta}$, we can determine the marginal posterior distributions $p(S_n)$ that maximize $p(\mathbf{o})$.

- If we know the $p(S_n)$, we can determine the parameters $\boldsymbol{\theta}$ that maximize $p(\mathbf{o})$.

Unfortunately we have neither.

Solution: an ***Expectation-Maximization*** (EM) algorithm

1. Initialize $\boldsymbol{\theta}$ suitably.

2. Iterate until convergence:

   a. **E Step**

      Run the Forward-Backward algorithm to determine posterior probabilities.

   b. **M Step**

      Maximize $p(\mathbf{o})$ over $\boldsymbol{\theta}$.

Over such a sequence of E and M steps the data likelihood will never decrease. While this guarantees convergence, it does not guarantee convergence to a global maximum.

4.2. Notation

$$
\begin{aligned}
\gamma(S_n) &= p(S_n \mid \mathbf{o}, \theta_{\text{old}}) \\
\xi(S_{n-1}, S_n) &= p(S_{n-1}, S_n \mid \mathbf{o}, \theta_{\text{old}})
\end{aligned}
$$

4.3. E Step

Compute using the Forward-Backward algorithm and $\boldsymbol{\theta}_{\text{old}}$:

$$
\begin{aligned}
& p(\mathbf{o}) \\
\alpha(S_n) &= p(o_1, \ldots, o_n, S_n) = \mu_{f_{S_{n-1}S_n} \to S_n} \mu_{f_{S_n O_n} \to S_n} \\
\beta(S_n) &= p(o_{n+1}, \ldots, o_N \mid S_n) = \mu_{f_{S_n S_{n+1}} \to S_n} \\
\gamma(S_n) &= \frac{\alpha(S_n)\beta(S_n)}{p(\mathbf{o})} \\
\xi(S_{n-1}, S_n) &= \frac{\alpha(S_{n-1}) p(o_n \mid S_n) p(S_n \mid S_{n-1}) \beta(S_n)}{p(\mathbf{o})}
\end{aligned}
$$

- The so-called *forward variable* $\boldsymbol{\alpha}$ and *backward variable* $\boldsymbol{\beta}$ are introduced here in keeping with notational conventions of the Forward-Backward algorithm.

- The calculation of $\boldsymbol{\gamma}$ arises from the Sum-Product message product followed by the product rule of probability; see also Exercise 7.

- For $\boldsymbol{\xi}$ see Exercise 8.

4.4. M Step

These hold for *homogeneous* HMM with *discrete* observation probabilities:

$$
\begin{aligned}
p(S_1) &= \gamma(S_1) \\
p(S_n = t \mid S_{n-1} = s) &= \frac{\sum_{n=2}^{N} \xi(S_{n-1} = s, S_n = t)}{\sum_{t'} \sum_{n=2}^{N} \xi(S_{n-1} = s, S_n = t')} \\
p(O_n = o \mid S_n = s) &= \frac{\sum_{n=1}^{N} \gamma(S_n = s) 1_{O_n = o}}{\sum_{n=1}^{N} \gamma(S_n = s)}
\end{aligned}
$$

The left-hand sides we collectively called θ above.

The second and third equations compute, respectively, the desired conditional by dividing the joint by the marginal.

5. Exercises

5.1. Markov Models and Parameters

Determine the number of parameters of

1. N i.i.d. variables
2. a length-N Markov chain
3. a length-N, Mth-order Markov chain
4. a length-N HMM
5. a fully-connected model with N variables

where all random variables take K distinct values.

6. Show that any higher-order Markov model can be converted into a first-order Markov model.

5.2. Baum-Welch

7. In the E step of the Baum-Welch algorithm, verify the result for γ, without referring to the Sum-Product message product.

8. Ibidem, verify the result for ξ.

   The numerator of ξ equals $p(S_{n-1}, S_n, \mathbf{o})$.

$$
\begin{aligned}
& p(\mathbf{o}_{<n}, S_{n-1}) p(o_n \mid S_n) p(S_n \mid S_{n-1}) p(\mathbf{o}_{>n} \mid S_n) & \\
= \; & p(\mathbf{o}_{<n}, S_{n-1}) p(o_n, \mathbf{o}_{>n} \mid S_n) p(S_n \mid S_{n-1}) & O_n \perp\!\!\!\perp O_{>n} \mid S_n \\
= \; & p(\mathbf{o}_{<n}, S_{n-1}) p(o_n, \mathbf{o}_{>n} \mid S_n, S_{n-1}) p(S_n \mid S_{n-1}) & O_n, O_{>n} \perp\!\!\!\perp S_{n-1} \mid S_n \\
= \; & p(\mathbf{o}_{<n}, S_{n-1}) p(o_n, \mathbf{o}_{>n}, S_n \mid S_{n-1}) & \\
= \; & p(\mathbf{o}_{<n} \mid S_{n-1}) p(o_n, \mathbf{o}_{>n}, S_n \mid S_{n-1}) p(S_{n-1}) & \\
= \; & p(\mathbf{o}_{<n}, o_n, \mathbf{o}_{>n}, S_n \mid S_{n-1}) p(S_{n-1}) & O_{<n} \perp\!\!\!\perp O_n, O_{>n}, S_n \mid S_{n-1} \\
= \; & p(S_{n-1}, S_n, \mathbf{o}_{<n}, o_n, \mathbf{o}_{>n}) &
\end{aligned}
$$

9. Show that $\mu_{f_{S_n S_{n+1}} \to S_n} = p(o_{n+1}, \ldots, o_N \mid S_n)$ (used at the E step). Proceed similarly to our derivation for the forward pass.

10. We defined the Baum-Welch algorithm for training a HMM on a single observation sequence. Think about how to extend it for training on multiple observation sequences. (See Ex. 13.12 [Bishop 2006] for more information.)

6. References

6.1. References

C. Bishop, *Pattern Recognition and Machine Learning* [1], Springer 2006.

L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition [2]". *Proceedings of the IEEE* 77(2), pp. 257–286, 1989.

[1] https://www.microsoft.com/en-us/research/people/cmbishop/#!prml-book/
[2] http://dx.doi.org/10.1109/5.18626