Hidden Markov Models

Diljeet Jagpal

Spring 2021

Preface

Contents

Chapter 1

Introduction

Chapter 2

Hidden Markov Models

Before we try to apply Hidden Markov Models (HMMs) to rainfall, we must first understand how they work. In this chapter we will build an understanding for the basic principles behind HMMs, along with developing and awnsering a few key questions.

2.1 Definition

In this section we will define a HMM and introduce notation that will be used throughout this paper.

A hidden Markov model is a doubly stochastic process, where the underlying process is Markovian and hidden; Rabiner and Juang, 1986. This comes from the fact that there are two stochastic processes, one determining the transition between states and one determining the output observation. Braum and his colleagues developed this in Leonard E Baum, Eagon, et al., 1967 and Leonard E. Baum and Petrie, 1966.

To define a HMM, we require five things. These are as follows: To define a hidden Markov model, we need five things.

1. N
   - i N is the number of hidden states. Usually based on something in the real world but sometimes can be unknown.
   - ii The states are usually ergodic, i.e. from any given state, we can eventually reach all others, but this is not necessary.
   - iii The states are from the state space $S = \{S_1, S_2, ..., S_N\}$.
2. M
   - i M is the number of observable outputs.
   - ii These make a discrete alphabet of observations called $V = \{v_1, v_2, ..., v_M\}$.
3. A
   - i A is the state transition matrix.
   - ii This is the same as the p matrix for simple Markov chains.
   - iii A = $\{a_{ij}\}$

iv $a_{ij} = \{p(i,j) = \mathbb{P}\{X_n = j | X_{n-1}\}$

4. B

   i B is the observation probability matrix.

   ii B $= \{b_j(k)\}$

   iii $b_j(k) = \mathbb{P}(V_k \ at \ t | q_t = S_j)_{1 \leq j \leq N, 1 \leq k \leq M}$

5. π

   i π is a vector containing all inital state probabilities.

   ii $\pi = \{\pi_i\}$

   iii $\pi_i = \mathbb{P}(q_1 = S_i)$ for $1 \leq i \leq N$

We can now combine the above to provide a formal definition of HMMs.

Definition 2.1. Hidden Markov Model
A Hidden Markov Model is a 5-tuple $\{N, M, A, B, \pi\}$ that is used to represent a doubly stochastic process where the hidden process is Markovian.

Before we continue, we will also provide some more notation, which will be used for the rest of this paper.

i Q represents the Markov model's state sequence, i.e. the hidden process.

ii We will use $O = \{o_1, o_2, ..., o_T\}, \ O_i \in V$ to represent the observation sequence.

iii we will occasionally represent the hidden markov model as $\{N, M, \lambda\}$ where $\lambda = \{A, B, \pi\}$

2.2 Using HMM

As with all models, we can use HMM to simulate processes. In this section we will demonstrate how to do so for a HMM with known parameters as well as propose a few questions one may ask.

2.2.1 Simulation

Simulating a process is usually our reason for building a model. Thus, in this subsection, we will demonstrate how to simulate a HMM to generate a observation sequence.

Let H $= \{N, M, A, B, \pi\}$ be a HMM. To generate a sequence of T observations O we do the following steps:

1. Using π as a probability measure, set $t = 1$ and randomly sample a state as the first $q_1 = s_i$.

2. Using $b_i(k)$ as a probability measure, randomly sample the observation $O_t = v_k$.

3. Using row i of a_{ij} as a probability measures, set $t = t + 1$ and randomly sample a the next state $q_{t+1} = s_j$.

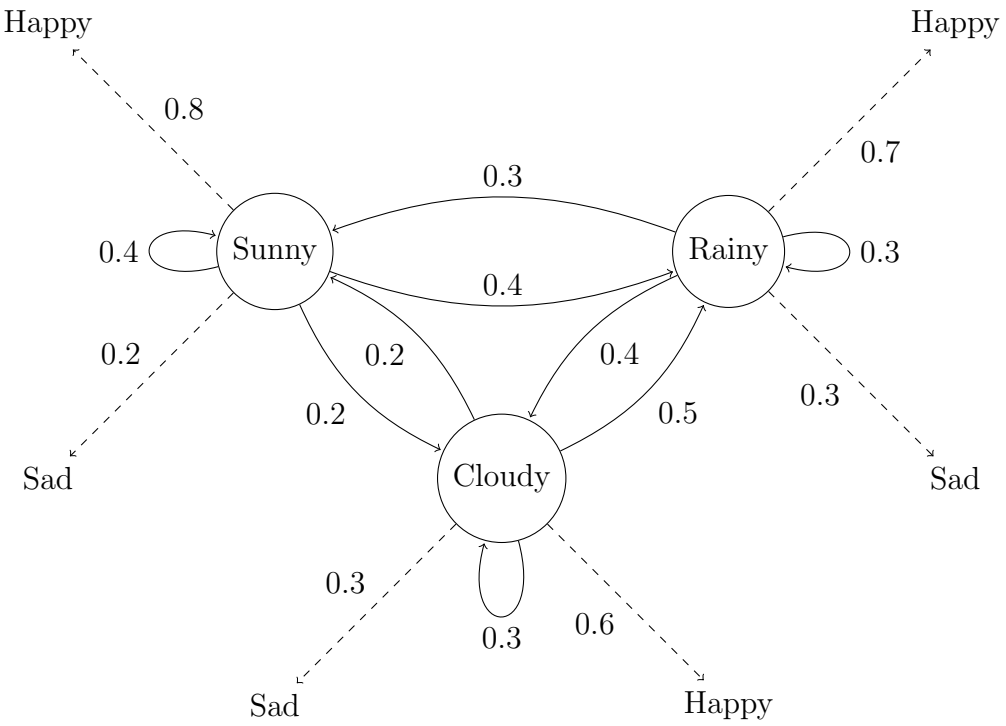

Figure 2.1: A chain representing the HMM with hidden states the weather and observable states mood.The solid lines represent transitions between hidden states, and the dashed represent observable.

4. Repeat steps 2 and 3 until $t = T$

Example 2.2. Suppose Alice is hidden away from the world and has no access to information regarding the weather. She meets Bob every day and knows how weather affects his mood. For simplicity, assume Bob only has two moods, happy and sad. Given matrix A, B and vector π can she predict his mood for three consecutive days?

From context we can deduce the following:

$$\begin{gathered}\text{N} = 3, \text{ number of hidden states} \\ \text{M} = 2, \text{ number of possible observations}\end{gathered}$$

$$A = \begin{matrix}(Sunny) \\ (Rainy) \\ (Cloudy)\end{matrix} \begin{bmatrix}0.4 & 0.4 & 0.2 \\ 0.3 & 0.3 & 0.4 \\ 0.2 & 0.5 & 0.3\end{bmatrix} \tag{2.1}$$

$$B = \begin{matrix}(Sunny) \\ (Rainy) \\ (Cloudy)\end{matrix} \begin{bmatrix}0.8 & 0.2 \\ 0.7 & 0.3 \\ 0.6 & 0.4\end{bmatrix} \tag{2.2}$$

$$\pi = \begin{matrix}(Sunny) \\ (Rainy) \\ (Cloudy)\end{matrix} \begin{bmatrix}0.5 \\ 0.3 \\ 0.2\end{bmatrix} \tag{2.3}$$

We can now use 2.2.1 to create an observation sequence $O = \{o_1, o_2, o_3\}$. We will require multiple random variables generated using various probability measures. We will generate uniform random variabes through python using the "rand.py" file. We will then combine these with inversion sampling to select the state.

i We generate a r.v. $= 0.0058$. Using Pi as the probaility distribution, we select Sunny. We can now set $q_1 = s_1$.

ii We generate a r.v. $= 0.1947$. Using $b_1(k)$ as the probability distribution we select Happy as the observation. We can now set $o_1 = v_1$.

iii we generate a r.v. $= 0.7168$. Using a_{1j} as the probability distribution we select Rainy as the next state. We now set $q_2 = s_2$.

iv We generate a r.v. $= 0.1060$. Using $b_2(k)$ as the probability distribution we select Happy as the observation. We can now set $o_2 = v_1$.

v we generate a r.v. $= 0.8977$. Using a_{2j} as the probability distribution we select Cloudy as the next state. We now set $q_3 = s_3$.

vi We generate a r.v. $= 0.1369$. Using $b_3(k)$ as the probability distribution we select Happy as the observation. We can now set $o_2 = v_1$.

Finally, we can look back on our simulated observations O and see that it is equal to $\{v_1, v_1, v_1\}$, i.e. we observe three consecutive Happy days.

2.2.2 Key Problems

To ensure the validity of a model, one may ask questions regarding the probabilities of certain occurances. We discuss these questions in this subsection.

We will focus on three famous questions highlighted in Rabiner and Juang, 1986.

1. Evaluation
   Given model H $= \{N, M, A, B, \pi\}$ what is the probability that it generated the sequence of observations $O = \{o_1, o_2, ..., o_T\}$? i.e. $\mathbb{P}(O \mid H)$

2. Decoding
   What sequence of states $Q = \{q_1, q_2, ..., q_3\}$ best explains a sequence of observations $O = \{o_1, o_2, ..., o_T\}$?

3. Learning
   Given a set of observation $O = \{o_1, o_2, ..., o_T\}$, how can we learn the model H $= \{N, M, A, B, \pi\}$ that would generate them?

In the coming sections, we will be motivating uses and developing solutions for each problem.

2.3 Problem 1: Evaluation

In this section we will tackle the Evaluation problem for HMM; 1. We will first attempt using a simple approach and then find a more efficient implementation; the Forward-Backward Algorithm.

We are looking for the probability that a given model generated a sequence of observations, i.e. $\mathbb{P}(O|\lambda)$. This probability has many useful applications. For example, we may have multiple potential models λ_i and cannot decide which one is most suitable. We can calculate this probability for each λ_i and the select the one with the largest.

To find this probability, we must consider the hidden internal states of the model. Since our probability of observations $\{b_j(k)\}$ is conditioned on the hidden state, we can start by calculating this probability conditioned on these states. Let us assume we have the state sequence $Q = \{q_1, q_2, ..., q_t\}$. For $\mathbb{P}(O|Q, \lambda)$ we can find the product of all the probabilities of an observation given the models state at all times t. In essence, this breaks the $\mathbb{P}(O|Q, \lambda)$ into T parts.

$$\mathbb{P}(O|Q, \lambda) = \prod_{t=1}^{T} \mathbb{P}(O_t|q_t, \lambda) \tag{2.4}$$

One may observe that these probabilities are simply taken from the matrix B.

$$\mathbb{P}(O_t|q_t, \lambda) = b_{q_t}(O_t), \quad \forall t \in [0, T] \tag{2.5}$$

Thus we can rewrite 2.4 as:

$$\mathbb{P}(O|Q, \lambda) = b_{q_1}(O_1)b_{q_2}(O_2)...b_{q_T}(O_T) \tag{2.6}$$

Our next objective is to remove Q from the conditioned part of the probability. To do this, we must first calculate $\mathbb{P}(Q|\lambda)$. This is simply the probability of transitioning from q_1 to q_2, q_2 to q_3 etc. More formally we can use matrix A to find the probability of each of these transitions, and since we are finding the total for the entire sequence, we multiply them all together. We start with π_{q_1} as we also need the probability of starting at q_1.

$$\mathbb{P}(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} ... a_{q_{T-1} q_T} \tag{2.7}$$

We can now successfully remove Q from the condition using 2.6 and 2.7:

$$\begin{aligned} \mathbb{P}(O, Q|\lambda) &= \mathbb{P}(O|Q, \lambda)\mathbb{P}(Q|\lambda) && (2.8) \\ &= \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) a_{q_2 q_3} ... a_{q_{T-1} q_T} b_{q_T}(O_T) && (2.9) \end{aligned}$$

This is the joint probability of observations and the internal states. In other words, it provides the probabilty that given observations O and internal state sequence Q were generated by model λ. To achieve our desired probablity all we need to do is get rid of the Q. Since it is another input, all we must do is sum each value of 2.8. As we have accounted for every possible Q, we no longer need to worry about its particular value. This leaves us with:

$$\begin{aligned} \mathbb{P}(O|\lambda) &= \sum_{allQ} \mathbb{P}(O|Q, \lambda) P(Q|\lambda) && (2.10) \\ &= \sum_{q_1, q_2, ..., q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) a_{q_2 q_3} ... a_{q_{T-1} q_T} b_{q_T}(O_T) && (2.11) \end{aligned}$$

Unforuntately calculating this is infeasible as it requires too many computations. For T timesteps and N states, to find every possible Q, we must sum over N^T state sequences. For each timestep we require a multiplication to $a_{q_{i-1}q_i}$ and $b_{q_i}(O_i)$, except the last where there are no transitions, which leads to $2T-1$ multiplications for each state sequence. Lastly, we require N^T addition operations to sum the result for each state sequence. Our final total number of operations of $(2T-1)N^T + (N^T - 1)$.

Example 2.3. Suppose we would like to find the probability of a string of 10 observations using model H, where $N = 3$ and $T = 10$.

$$\begin{aligned} operations &= (2T-1)N^T + (N^T - 1) && (2.12)\\ &= (20-1)3^10 + (3^10 - 1) && (2.13)\\ &= 1180979 && (2.14) \end{aligned}$$

We have a problem as even with a small model we require a considerable number of calculations. For a moderate to large-sized model, we would require an infeasible amount of calculations. To overcome this problem, we look for a more efficient method.

2.3.1 Forward-Backward Algorithm

The Forward-Backward (FB) algorithm offers an alternative, more efficient, method to find $\mathbb{P}(O \mid H)$. In this subsection we will demonstrate its use. Our explanations take motivation from Rabiner and Juang, 1986.

To use FB, we must first introduce two helper functions α and β. We will start by discussing the former.

$$\alpha_t(i) = \mathbb{P}(O_1, O_2, O_3, ..., O_t, q_t = S_i|\lambda) \tag{2.15}$$

α is an extremly powerful tool in reducing the number of calculations. As given by 2.15, it providse the probability that at time t we have seen a sequence of observations and are currently at state $q_t = S_i$. This is not quite $\mathbb{P}(O|\lambda)$ but it represents a part of it. Instead of the probability of the whole sequence, it breaks it into a chunk of size t, commits to end at a particular state and then calculates the same proabbility for this.

We can combine 2.15 with induction to produce an iterative process that can solve our problem.

i Base case
For the base case we require the probability of the q_1 being equal to S_1 and giving us observation O_1. The former is addressed by π_i and the later by $b_i(O_1)$. This gives us:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad i \in [1, N] \tag{2.16}$$

ii Inductive step:

For the inductive step we must consider how to approach the next timestep. We will again be calculating for all $j \in [1, N]$ and as such must take into consideration, for each j, every possible i. This is again the same set of $[1, N]$. Therefore, to account for all possible previous states and their transition to the current state, we must sum over 1 to N the product of $\alpha_t(i)$ and a_{ij}. For the given observation, as before, we compute

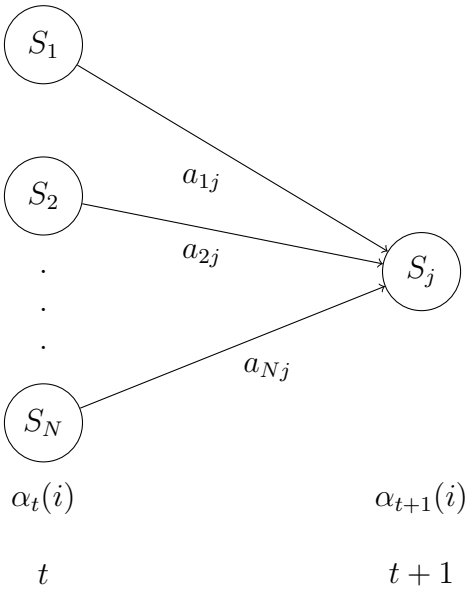

Figure 2.2: For each state j, for each inductive step, we follow the above method. Each$\alpha_t(i)$ is multiplied by each a_{ij} and then summed up. This resultmultipled by $b_j(O_{t+1}$ gives us $\alpha_{t+1}(j)$.

$b_j(O_{t+1})$. Additionally, we must stop before reaching the final step as there is no outward transition and thus this would not be applicable. This gives us:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)a_{ij}\right] b_j(O_{t+1}), \quad j \in [1, N], t \in [1, T-1] \tag{2.17}$$

iii Termination step:
For the termination step, we sum over all alpha at the final time T. Each alpha represents the probability of being in that given state at that particular time, through all possible sequences that it may have followed.

$$\mathbb{P}(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.18}$$

One can see that this method is far more efficient than 2.10. It requires N^2 multiplications for $\alpha_t(i)$ and a_{ij} for T time periods. At each T there are N addition operations for the summation and N multiplication for the $b_j(O_{t+1})$, which is true for all but the first and last timestep, where there are N multiplications and N additions respectively. This gives us $(T-2)(N^2+N+1)+2N$ calculations. Considering the order, we see that the previous method had order $O(TN^T)$ whereas F-B gives us an order of $O(TN^2)$. It is now feasible to compute $\mathbb{P}(O|\lambda)$.

Example 2.4. We attempt to solve the problem stated in 2.3 but this time using the FB algorithm. Suppose we would like to find the probability of a sequence of 10 observations

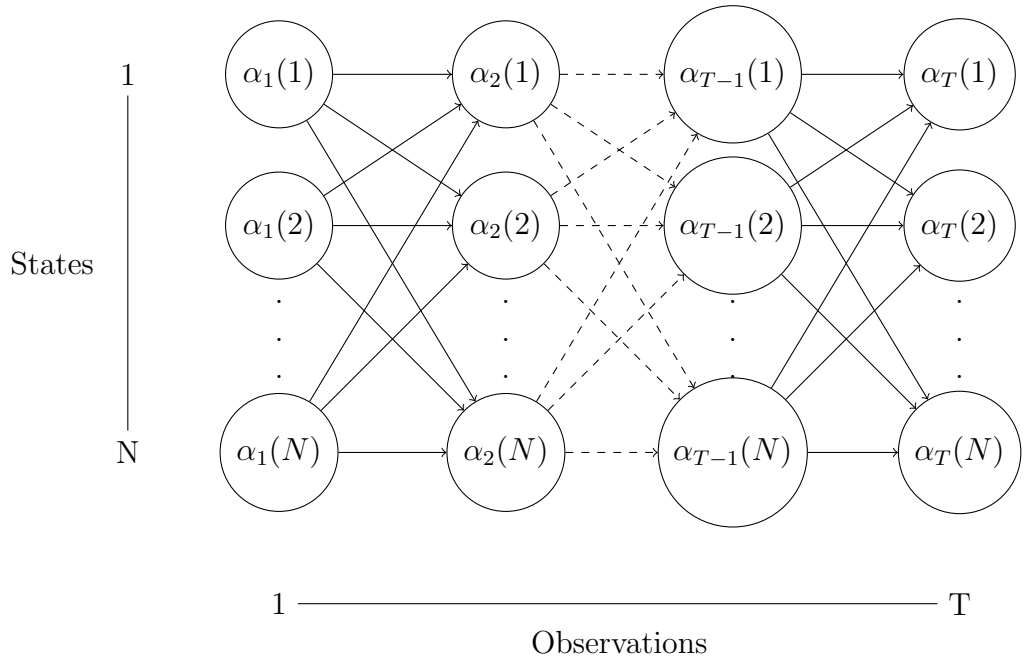

Figure 2.3: For each α at time t, we require all α at time $t-1$. Thus werequire N^2 calculations for each timestep.

using model H. Again, $N = 3$ and $T = 10$.

$$\begin{aligned} operations &= (T-2)(N^2+N+1)+2N && (2.19)\\ &= (10-2)(3^2+10+1)+6 && (2.20)\\ &= 118 && (2.21) \end{aligned}$$

118 is many orders of magnitude smaller than 1180979, which was the requirement without FB algorithm. Thus it is a significant improvement. The improvement against the base method will be even more drastic for larger models, as FB has a fixed N^2 instead of the N^T term.

For question 1 this is sufficient. However, we will later require the β, backward component and as such we will describe it now. Logically it is the same principle as the forward component except this time instead of moving forward step by step we are moving backwards.

$$\beta_t(i) = \mathbb{P}(O_{t+1}, O_{t+2}, , ..., O_T | q_t = S_i, \lambda) \tag{2.22}$$

This represents the probability of seeing the observations O_{t+1} up to O_T given that at time t the model λ is at state S_i. Once again we calculate this using induction.

1. Base case
   For each state we must assume that it is the final state in order to iterate from it. This gives us:

$$\beta_T(i) = 1, \quad i \in [1, N] \tag{2.23}$$

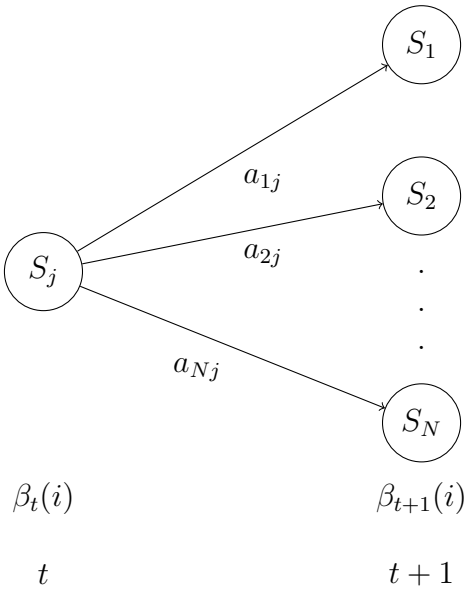

Figure 2.4: Similar to α, we use induction. However, this time we look into the future of the sequence and take steps backwards.

2. Inductive step:

   Each inductive step we move back by one timestep. As before, we will be calculating for all N states except this time i will be varying instead of j. This makes sense as we are stepping backward and we want to see which previous state is the most likely previous state. At each step, the transition probability of having coming from i, a_{ij} to the probability of seeing the given observation at state S_j, $b_j(O_{t+1})$ and the liklihood of being at that state based on future states $\beta_{t+1}(i)$ are multiplied. This gives us:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(i), \quad i \in [1, N], t \in [1, T-1] \tag{2.24}$$

2.4 Problem 2: Decoding

In this section we will address the decoding problem 2. We will use the two helper functions developed in 2.3 togther with the Viterbi Algorithm to do so. We will use ideas presented in Rabiner and Juang, 1986 and Forney, 1973.

The problem searches for the sequence of states that best explains the sequence of observations. However, the definition of 'best' is quite vague and open to interpretations. An obvious appraoch would be to look for each time t, find the most likely state, given all observations and the model. Lets define this probaility as γ.

$$\gamma_t(i) = \mathbb{P}(q_t = S_i | O, \lambda) \tag{2.25}$$

Using 2.15 and 2.22 we can solve for γ quite quickly. If we recall, $\alpha_t(i)$ provides us with the probability of being in state i after seeing all the previous observations and $\beta_t(i)$ gives us the probability of being in state i see all the remaining observations in the future.

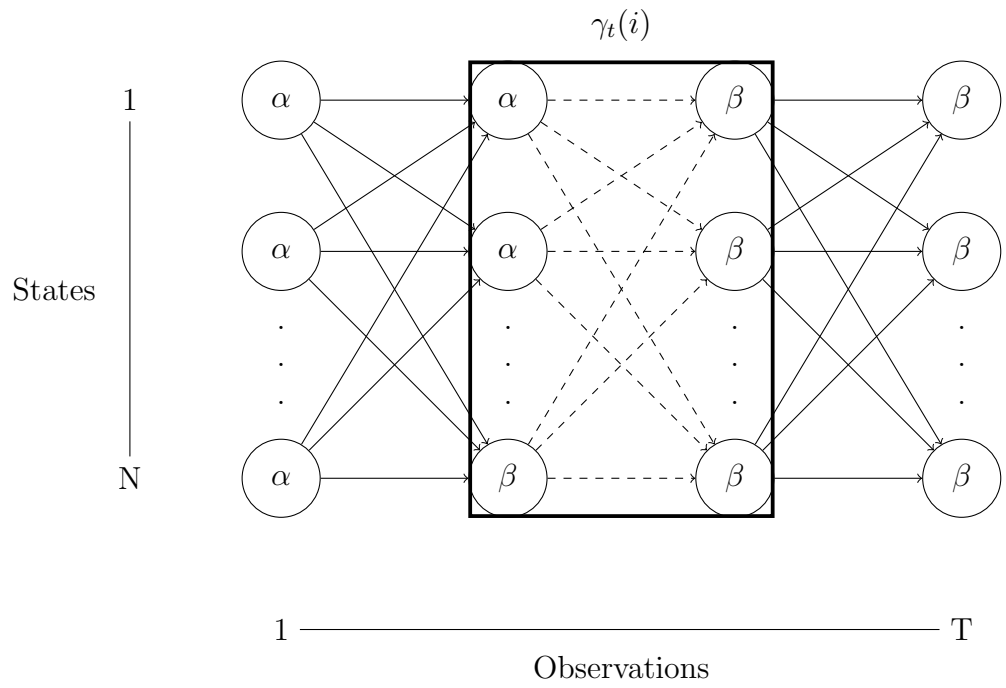

Figure 2.5: Each dashed line represents a calculation for $\gamma_t(i)$. For each time step, we calculate these and then find the maximum. We select the state corresponding to the maximum and add it to the state sequence.

This suggests γ is equal to $\alpha_t(i)\beta_t(i)$. We now normalise this to get a probability, which is possible by dividing by the probability of getting this particular observation given all possible observation sequences $\mathbb{P}(O|\lambda)$. This is equivalent to dividing by the sum of $\alpha_t(j)\beta_t(j)$ over all possible states j.

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\mathbb{P}(O|\lambda)} \tag{2.26}$$

$$= \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)} \tag{2.27}$$

This ensures that:

$$\sum_{i=1}^{N}\gamma_t(i) = 1 \tag{2.28}$$

As such this ensures all $\gamma_t(i)$ are probabilities. We can now go through each timestep t calculating $\gamma_t(i)$ and selecting state i corresponding to the largest $\gamma_t(i)$.

This method will maximise the number of correct states and give the correct answer if the model is completely connected, i.e. each state can reach every other state. If this is not the case, we may get a state sequence that is not possible. e.g. At time t the model is in state i and at time $t+1$ it is in j, but $a_{ij} = 0$, thus is not possible.

In the scenario given, we must redefine 'best' to the path that maximises $\mathbb{P}(Q|O,\lambda)$. Since we are maximising this, we can equivalently maximise $\mathbb{P}(Q,O|\lambda)$. To solve this problem, we require the Viterbi Algorithm.

2.4.1 Viterbi Algorithm

In this subsection we will demonstrate the application of the Viterbi Algorithm.

The Viterbi Algorithm was developed in Viterbi, 1967 and explored in further depth by in Forney, 1973. It follows a similar lattice structure of the helper functions in the FB algorithm 2.3.1. Similar to how we broke 1 into smaller pieces, we do the same to $\mathbb{P}(Q|O,\lambda)$.

$$\delta_t(i) = max_{q_1,q_2,...,q_t-1}\mathbb{P}(\{q_1,q_2,...q_t=i\},O_1,O_2,...,O_t|\lambda) \tag{2.29}$$

An important point to note is $\delta_t(i)$ does not store the sequence. Thus we must introduce a new variable that will be responsible for doing so. We can call this $\psi_t(i)$, where it is equal to the state we have come from given we are at time t and state i. We will once again use induction. The process is as follows.

1. Base case
   For each state we must calculate $\pi_i b_i(O_1)$ to determine which intial state is most likely. We also must set $\psi_1(i)=0$ as there have not been any states until now.

$$\begin{aligned}\delta_1(i) &= \pi_i b_i(O_1) &(2.30)\\ \psi_1(i) &= 0 &(2.31)\end{aligned}$$

2. Inductive step:
   For each state we calculate the $\delta_{t-1}(i)$ times a_{ij} to find the most likely next state based on previous state liklihood and the transition probability. We maximize this term and the multiply by $b_j(O_t)$ to include a bias based on the observation probabilities. As before we store the argument of the maximum in ψ. This gives us:

$$\begin{aligned}\delta_t(j) &= max_{1\le i\le N}[\delta_{t-1}(i)a_{ij}]b_j(O_t), t\in[2,T] &(2.32)\\ \psi_t(j) &= argmax_{1\le i\le N}[\delta_{t-1}(i)a_{ij}], j\in[1,N] &(2.33)\end{aligned}$$

3. Termination step:
   To terminate this recursion, we need to find the maximum $\delta_T(i)$. Here we maximise, as the values we already calculated the values in the induction. We will store this probability in $\mathbb{P}^*$. Similarly, we need to find the argument corresponding to this max for the final state, and we set this to the final state q_T^*.

$$\begin{aligned}\mathbb{P}^* &= max_{1\le i\le N}[\delta_T(i)] &(2.34)\\ q_T^* &= argmax_{1\le i\le N}[\delta_T(i)] &(2.35)\end{aligned}$$

   To find any particular states that we may be interseted in q_t^*, we use $q_t^* = \psi_{t+1}(q_{t+1}^*)$.

2.5 Problem 3: Learning

HMMs have a large number of parameters. Obtaining these is a high-dimensional optimisation problem. In this section we utilise the Baum-Welch Algorithm to simulatenously solve for all HMM parameters.

The learning problem 3 addresses how can we learn a model from given sequence of observations? Given the sequence of observations $O=\{O_1,O_2,...,O_T\}$ we want to find the most suitable $\lambda=(A,B,\pi)$.

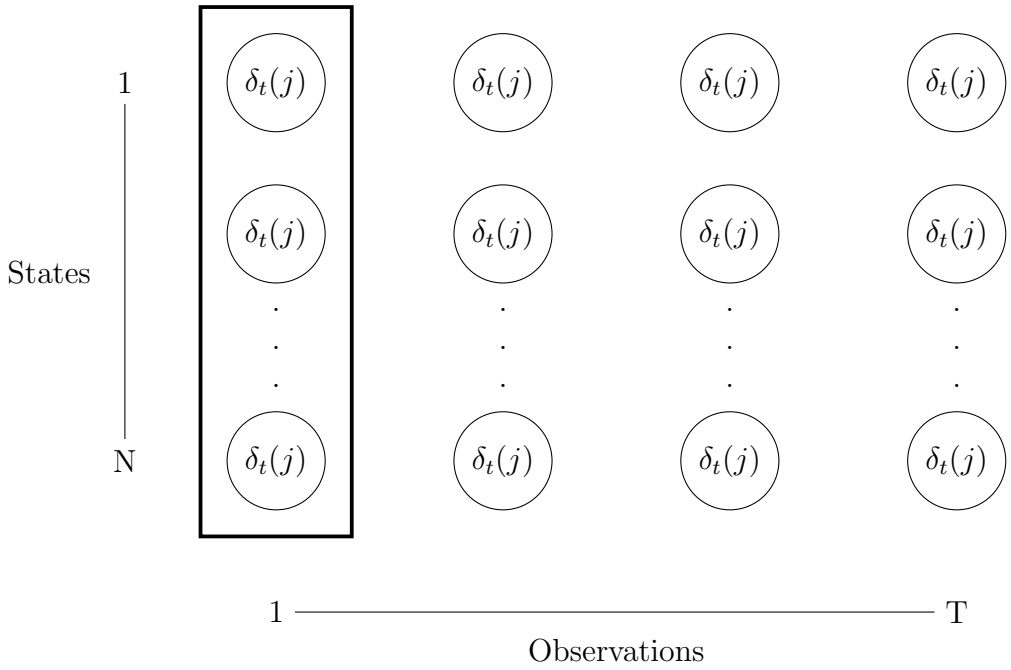

Figure 2.6: Once $\delta_t(j)$ is calculated for a particular t, we find the maximum, which is the state we add to our state sequence. Since $\delta_t(j)$ accounts for transition probability a_{ij}, impossible transitions will always be equal to 0. Thus unless all transitions are equal to 0, they cannot be the maximum.

2.5.1 Baum-Welch Algorithm

In this subsection we will demonstrate the Baum-Welch Algorithm. Explanations have been inspired by Leonard E Baum et al., n.d.

For the Baum-Welch algorithm we will need all three parameters intrudced earlier; α 2.15, β 2.22 and γ 2.25. We will also need a new parameter $\xi_t(i,j)$. This will capture the probability of being in some state S_i at time t and then S_j at time $t+1$ given the observations and the model.

$$\xi_t(i,j) = \mathbb{P}(q_t = S_i, q_{t+1} = S_j | O, \lambda) \tag{2.36}$$

To solve this problem we can refer back to 2.15 for the left hand side (including q_t) and 2.22 for the right hand side (indlucing q_{t+1}). To get from state i to j we use $a_{ij}b_j(O_{t+1})$. Putting this all togther we get a liklihood. To normalize this into a probabiltiy we once more use $\mathbb{P}(O|\lambda)$. This gives us:

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\mathbb{P}(O|\lambda)} \tag{2.37}$$

Since we are interested in the probability of going from one i to one j, for the denominator we use the sum of all i and all j.

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \tag{2.38}$$

One may find it helpful to compare $\xi_t(i,j)$ with $\gamma_t(i)$. $\gamma_t(i)$ represents the probability of being at state i at timestep t. Summing over j for $\xi_t(i,j)$ leaves us with $\xi_t(i)$ which

represents the probability of being at state i at timestep t. This is identical to $\gamma_t(i)$. We can now state:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{2.39}$$

For $\gamma_t(i)$, if we sum over t we can get a number that can be used as the expected number of times S_i is visited. Thus the expected number of transitions from S_i can be calculated by:

$$\sum_{t=1}^{T-1} \gamma_t(i) \tag{2.40}$$

Similarly for $\xi_t(i)$, if we sum over t we can get a number that can be used as the expected number of times S_i transitions to S_j.Thus the expected number of transitions from S_i to S_j can be calculated by:

$$\sum_{t=1}^{T-1} \xi_t(i,j) \tag{2.41}$$

Now that we have created a set of tools, we will need to tackle the learning problem itself. To build a improved λ, $\bar{\lambda}$, we need to find $\bar{\pi}$, $\bar{A}$ and $\bar{B}$.

i $\bar{\pi}$

Since $\gamma_t(i)$ represents the expected frequency in S_i at time t, if we let t=1 this now is equvialent to $\bar{\pi}$

$$\bar{\pi} = \gamma_1(i) \tag{2.42}$$

ii $\bar{a_{ij}}$

Here we can use the expected number of transitions from state i to j divided by expected number of transitions from i in total. This will provide the appropriate transition probability.

$$\bar{a_{ij}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{2.43}$$

iii $\bar{b_j(k)}$

For the B matrix, we will need to divide the expected number of times the model is in state j and observes v_k by the expected number of times it is j.

$$\bar{b_j(k)} = \frac{\sum_{t=1,s.t.O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T-1} \gamma_t(j)} \tag{2.44}$$

We have used our λ and O to produce our parameters: $\alpha_t(i)$, $\beta_t(i)$, $\gamma_t(i)$ and $\xi_t(i,j)$. This is called parameter re-estimation. We have then used these parameters to produce our $\bar{\lambda}$ = $\{\bar{\pi}, \bar{a_{ij}}, \bar{b_j(k)}\}$. This is called expectation maximization. This method can be iterated and each iteration should provide an improvement for lambda or worst case senario be equvialent. Proof of this can be found in Leonard E Baum et al., n.d. Once lambda stabalizes and is no longer changing, we can assume a local optimumum has been reached.

Example 2.5. Let us continue from Example 2.2. Alice is unsure of her model, thus records Bob's mood for five days. Her Observation sequence is {Happy, Sad, Happy, Sad, Sad}. She wants to use one iteration of Baum-Welch to improve her model. Code files are available in the "Example" Folder under "MyCode".

To apply the algorithm we need our $\alpha_t(i)$, $\beta_t(i)$, $\gamma_t(i)$ and $\xi_t(i,j)$.

We begin with $\alpha_t(i)$. We are expecting a lattice of size 3x5. We calculate using 2.15 and store the values in the following matrix:

$$\begin{bmatrix} 0.4 & 0.0494 & 0.048984 & 0.00912654 & 0.00250527 \\ 0.21 & 0.0849 & 0.059661 & 0.0173532 & 0.00540931 \\ 0.12 & 0.08 & 0.040704 & 0.018349 & 0.00570851 \end{bmatrix}$$

Similarly for $\beta_t(i)$ using 2.22,

$$\begin{bmatrix} 0.0178115 & 0.0648 & 0.0844 & 0.28 & 1 \\ 0.0197177 & 0.062835 & 0.0943 & 0.31 & 1 \\ 0.0196482 & 0.063591 & 0.0949 & 0.31 & 1 \end{bmatrix}$$

Using 2.25 to calculate $\gamma_t(i)$,

$$\begin{bmatrix} 0.522979 & 0.234978 & 0.303474 & 0.187581 & 0.183899 \\ 0.303949 & 0.391592 & 0.412978 & 0.39488 & 0.397069 \\ 0.173072 & 0.373431 & 0.283549 & 0.417539 & 0.419032 \end{bmatrix}$$

$\xi_t(i,j)$ produces a result which ressembles a transition matrix for each timestep. We will have 5-1 = 4 of these.

- $$\begin{bmatrix} 0.152212 & 0.221395 & 0.149372 \\ 0.0599335 & 0.0871742 & 0.156841 \\ 0.0228318 & 0.083023 & 0.0672175 \end{bmatrix}$$
- $$\begin{bmatrix} 0.0979363 & 0.0957461 & 0.0412951 \\ 0.126237 & 0.123414 & 0.141942 \\ 0.0793006 & 0.193818 & 0.100312 \end{bmatrix}$$
- $$\begin{bmatrix} 0.0805428 & 0.133759 & 0.0891724 \\ 0.073574 & 0.122185 & 0.217218 \\ 0.0334641 & 0.138936 & 0.111149 \end{bmatrix}$$
- $$\begin{bmatrix} 0.0535945 & 0.0803918 & 0.0535945 \\ 0.0764283 & 0.114643 & 0.203809 \\ 0.0538761 & 0.202035 & 0.161628 \end{bmatrix}$$

Using these along with 2.5.1 will allow us to re-estimate our A,B and π parameters. π is the first column of the γ matrix. Thus our new inital distribution is

$$\bar{\pi} = \{0.522979, 0.303949, 0.173072\} \tag{2.45}$$

The number of transtions from state i to j can be taken as the sum of a particular position over all η matrices. These can be divded by the sum of the corresponding γ row, disregarding the last timestep.

$$\bar{a_{ij}} = \begin{bmatrix} 0.307672 & 0.425369 & 0.266959 \\ 0.223609 & 0.297603 & 0.478789 \\ 0.151871 & 0.495204 & 0.352925 \end{bmatrix} \tag{2.46}$$

For each row of γ, we sum the values where the observation is Happy and where it is Sad and then divide both by the sum of the row. This gives us a row of our new observation matrix.

$$b_j\bar{(k)} = \begin{bmatrix} 0.576765 & 0.423235 \\ 0.377237 & 0.622763 \\ 0.27398 & 0.72602 \end{bmatrix} \tag{2.47}$$

This method can be repeated until convergence to retrieve the optimal parameters (local optimum).

Chapter 3

Model Selection

Selecting the best model to represent a process is often not a trivial task. In this chapter we discuss methods of comparing models against data, approximating parameters and testing whether a model fits.

3.1 Maximum Likelihood Estimators

We wish to find a model that can explain a given dataset. Assume we have two such models, how do we decide which is superior at this task? In this chapter we will build on this idea.

3.1.1 Likelihood

In this subsection we define likelihood and motivate the need for Maximum likelihood estimators.

To decide between models we must awnser which model explains the dataset better. This can be interpreted as the probability a given model produces a sequence of observations. This probability is known as the Likelihood. We can now define this; Ross, 2004.

Definition 3.1. Likelihood Function
Given model θ and the observations $x_1, x_2, ..., x_n$, the likelihood function is given by

$$L(\theta|x_1, x_2, ..., x_n) = \mathbb{P}(x_1, x_2, ..., x_n|\theta) \tag{3.1}$$

In other words it is given by the probability that the given model produced the given observations.

This function allows us to directly compare which model has a higher probability of fitting the data. A model with a higher Likelihood would have a superior fit to a model with a lower likelihood. As such, it is natural to maximise the Likelihood, in search of the ideal model.

3.1.2 Maximum Liklihood Estimators

Through maximizing the likelihood function 3.1, we can find the model that best fits the given observations. This process is called Maximum Likelihood estimation. In this subsection we will demonstrate its methodology.

We often use $\log(L(\theta|x_1, x_2, ..., x_n))$ instead of $L(\theta|x_1, x_2, ..., x_n)$ as this often leads to easier differenciation. This is possible as the log function is monotonically increassing thus has the same maximum at the same value of θ.

The method can be described as follows:

i Calculate the joint probability density of observeriving your data $X_1 = x_1, ..., X_n = x_n$ given your model θ, i.e.

$$\mathbb{P}(X_1 = x_1, ..., X_n = x_n|\theta) \tag{3.2}$$

ii Take the Natural logarithm of both sides.

$$\ln(\mathbb{P}(X_1 = x_1, ..., X_n = x_n|\theta)) \tag{3.3}$$

iii Take the partial derivative with respect to each parameter.

iv For each paramter, set the derivative equal to 0 and rearrange for its value. The given value will be the maximum likelihood estimation of that parameter.

Example 3.2. To demonstrate 3.1.2 we present the following example.

Let $\{1, 2, 4, 4, 4, 9\}$ be our Observed data. We are convived that this data follows an expoential(λ) distribution but we are unsure of what the λ parameter is. We begin by finding the joint probability density. According to our model all observations are independent thus the joint distribution is:

$$\mathbb{P}(X_1 = x_1, ..., X_6 = x_6|\theta) = \mathbb{P}(X_1 = x_1)...\mathbb{P}(X_6 = x_6) \tag{3.4}$$

We can now substitute in the PDF of an expoential distribtution at the different observations.

$$\mathbb{P}(X_1 = x_1, ..., X_6 = x_6|\theta) = \lambda e^{-\lambda}\lambda e^{-2\lambda}\lambda e^{-4\lambda}\lambda e^{-4\lambda}\lambda e^{-4\lambda}\lambda e^{-9\lambda} \tag{3.5}$$

We now simplify and take the natural log of both sides.

$$\begin{aligned} \ln(\mathbb{P}(X_1 = x_1, ..., X_6 = x_6|\theta)) &= \ln(\lambda^6 e^{-24\lambda}) && (3.6)\\ &= ln(\lambda^6) - 24\lambda && (3.7) \end{aligned}$$

We now differenciate with respect to our parameter, λ.

$$\frac{d}{d\lambda} = \frac{6}{\lambda} - 24 \tag{3.8}$$

Finally, to find the maximum we set the differential equal to 0 and solve for λ.

$$\begin{aligned} 0 &= \frac{6}{\lambda} - 24 && (3.9)\\ 24 &= \frac{6}{\lambda} && (3.10)\\ \lambda &= \frac{6}{24} && (3.11)\\ &= \frac{1}{4} && (3.12) \end{aligned}$$

Through MLE our parameter estimate for λ is 0.25 thus our model is expoential(0.25). Logically this seems to make sense as the expected value of this model is 4, which seems to be the most popular and central value from the dataset.

As we can see, even for a simple model the derivate is not trivial. Unfortunately in real world applications this is often the case, particurally true for Hidden Markov based models. Thus we often require alternative methods to compare models.

3.2 Approximate Bayesian Computation

Maximum Likelihood Estimators are not always easy to calculate. In some such cases Approximate Bayesian Computation (ABC) can provide an alternative method of parameter estimation. In this section we will discuss the principles of Bayesian Statistics, ABC and provide a demonstration.

ABC uses Bayesian inference. It involves using the prior distribution, our assumption of the model, combined with the likelihood to infer a posterior distribtution, the expected distribtution given our data and model. This can be more formally described using Bayes Theorem. Given x is the data and θ is the model,

$$\mathbb{P}(\theta|x) = \frac{\mathbb{P}(x|\theta)\mathbb{P}(\theta)}{\mathbb{P}(x)} \tag{3.13}$$

$$\mathbb{P}(\theta|x) \propto \mathbb{P}(x|\theta)\mathbb{P}(\theta) \tag{3.14}$$

$$posterior \propto likelihood * prior \tag{3.15}$$

ABC infers posterior distribtutions by replacing the likelihood calculatation with a comparison between observed and simulated data. Toni et al., 2009. While there are many variations of ABC, we will focus on the simpliest, ABC Rejection Sampling. The method can be described as follows:

1. Sample θ' from the prior distribution $\mathbb{P}(\theta)$.
2. Simulate dataset x' using the sampled θ'.
3. Compare simulated dataset with observed dataset using a distance function. If distance is smaller than a preset ϵ then accept θ' otherwise reject. Distance function can use summary statistics.
4. Repeat previous steps.

Given a small enough tolerence (ϵ) and an appropriate statistic we have:

$$\mathbb{P}_\epsilon(\theta|x) = \int \mathbb{P}_\epsilon(\theta, x'|x)dx' \approx \mathbb{P}(\theta|x) \tag{3.16}$$

Example 3.3.

3.3 Kolmogorov-Smirnov Test

Chapter 4

Replicating Existing Rainfall Models

4.1 Cowpertwait

4.2 Grando

4.2.1 Model

4.2.2 Fitting

4.3 Replicating Grando

4.3.1 Code

4.3.2 Results

Nine Parameter Estimation

Three Parameter Estimation

Adjusted Algorithm

Chapter 5

Extending HMM based Rainfall Models

5.1 Disc Structure

5.2 Seasonal Effects

5.3 High Dimensionality

Chapter 6

Simple Rainfall HMM

6.1 Baum-Welch

6.1.1 Building the Observation Matrix

6.1.2 Local vs Global Optimum

6.2 Analysis

6.2.1 Convergence

6.2.2 Selecting an Optimum

6.2.3 Results

Chapter 7

Generalising the Observation Matrix

7.1 Generalised Model

7.2 Function for Observation Matrix

7.3 Parameter Estimation

7.3.1 Methodology

7.3.2 Results

Chapter 8

Results

8.1 Datasets

8.2 Analysis

8.2.1 CDFs

8.2.2 Kolmogorov–Smirnov tests

Chapter 9

Future Research

Bibliography

Baum, Leonard E et al. (n.d.). "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes". In: ().

Baum, Leonard E, John Alonzo Eagon, et al. (1967). "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology". In: *Bulletin of the American Mathematical Society* 73.3, pp. 360–363.

Baum, Leonard E. and Ted Petrie (Dec. 1966). "Statistical Inference for Probabilistic Functions of Finite State Markov Chains". In: *Ann. Math. Statist.* 37.6, pp. 1554–1563. DOI: 10.1214/aoms/1177699147. URL: https://doi.org/10.1214/aoms/1177699147.

Cowpertwait, Paul (2009). "A Neyman-Scott model with continuous distributions of storm types". In: *ANZIAM Journal* 51, pp. C97–C108.

Cowpertwait, Paul, Valerie Isham, and Christian Onof (2007). "Point process models of rainfall: developments for fine-scale structure". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 463.2086, pp. 2569–2587.

Cowpertwait, Paul, Jim Salinger, and Brett Mullan (2009). "A spatial-temporal stochastic rainfall model for Auckland City: Scenarios for current and future climates". In: *Journal of Hydrology (New Zealand)*, pp. 95–109.

Cowpertwait, Paul SP (1994). "A generalized point process model for rainfall". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 447.1929, pp. 23–37.

— (1995). "A generalized spatial-temporal model of rainfall based on a clustered point process". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 450.1938, pp. 163–175.

— (1998). "A Poisson-cluster model of rainfall: some high-order moments and extreme values". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1971, pp. 885–898.

— (2006). "A spatial–temporal point process model of rainfall for the Thames catchment, UK". In: *Journal of Hydrology* 330.3-4, pp. 586–595.

— (2010). "A spatial-temporal point process model with a continuous distribution of storm types". In: *Water Resources Research* 46.12.

Cowpertwait, PSP, CG Kilsby, and PE O'Connell (2002). "A space-time Neyman-Scott model of rainfall: Empirical analysis of extremes". In: *Water Resources Research* 38.8, pp. 6–1.

Daley, Daryl J and David Vere-Jones (2007). *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media.

Dempster, Arthur P, Nan M Laird, and Donald B Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1, pp. 1–22.

Forney, G David (1973). "The viterbi algorithm". In: *Proceedings of the IEEE* 61.3, pp. 268–278.

Grando, Debora (2019). "Data-driven models and random generators of rainfall". MA thesis.

Markov, Andre (n.d.). "An example of statistical investigation of the text Eugene Onegin concerning the connection of samples in chains". In: ().

Rabiner, L. and B. Juang (1986). "An introduction to hidden Markov models". In: *IEEE ASSP Magazine* 3.1, pp. 4–16. DOI: 10.1109/MASSP.1986.1165342.

Ross, Sheldon M (2004). *Introduction to probability and statistics for engineers and scientists*. Elsevier.

Sharma, Jay (2020). "Hidden Markov Models And Their Applications In Stock Market Prediction". MA thesis.

Toni, Tina et al. (2009). "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems". In: *Journal of the Royal Society Interface* 6.31, pp. 187–202.

Viterbi, Andrew (1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE transactions on Information Theory* 13.2, pp. 260–269.