UNIVERSITY of HOUSTON

DEPARTMENT OF COMPUTER SCIENCE

Markov Models

Rakesh Verma

Markov Models

A **Markov Process** is a statistical process for which state of the process is strictly based on its previous state and not every state before it.

We can represent this mathematically as a sequence of states *X* and the following properties.

A Markov process has **limited horizon**.

Only the most recent state has impact on the next.

$$P(X_{t+1} = v_k \mid X_1, \ldots, X_t) = P(X_{t+1} = v_k \mid X_t)$$

A Markov process is also **time invariant**.

It doesn't matter when the how many state transitions there were beforehand.

$$P(X_{t+1} = v_k \mid X_t) = P(X_2 = v_k \mid X_1)$$

If a sequence *X* satisfies these properties, then it is a **Markov chain**.

Markov Chain Probability

A wonderful result of the **Markov Properties** is that calculating the probability of a Markov Chain occurring simplifies dramatically.

$$P(X_1, \ldots, X_t) = P(X_1)P(X_2 \mid X_1) \cdots P(X_t \mid X_1, \ldots, X_{t-1})$$
$$P(X_1, \ldots, X_t) = P(X_1)P(X_2 \mid X_1) \cdots P(X_t \mid X_{t-1})$$

As per the chain rule, then as per limited horizon.

We can calculate the probability of the entire sequence using only observations at each transition and the starting state.

Hidden Markov Models

Hidden Markov Models (**HMM**) are Markov Models for which we don't actually observe the states of the process directly.

Instead, we observe some statistical process which depends directly on the current state of the chain.

- As an example, let's say we have dice of every number of sides: our states.
- At each step, we roll our current die twice.
  - The first roll decides which die we use next: this transition has the Markov properties.
  - The second roll decides what value record for the process: these are our **emissions**.
- To an outside observer only presented with the recorded rolls, they cannot know for certain which type of die was rolled at each timestep.
  - Every roll result appears on more than one die.

This is referred to as a **state-emission HMM**, because the observed process depends on the current state.

Hidden Markov Models (Continued)

- **Arc-emission HMMs** are defined based on the transition pair.
  - Referring to the previous example, let's say we have all those dice again.
    - At each step, we roll our current die once.
    - That roll decides which die are going to transition to.
      - This transition is still only dependent on the current state.
    - Roll both die and take their sum.
      - This process is dependent on both the current and next state.
- If for every Y, the transition pair (X → Y) has the same emission distribution, then the model can be equivalently expressed as a state-emission HMM.
  - This means the emission is effectively independent of the future state.

Hidden Markov Models (Continued)

We can formalize both types of HMMs in the same way.

Let an HMM be a 5-tuple (S, K, Π, A, B).

- S is the set of hidden states.
- K is the emission alphabet.
- Π is the set of probabilities of starting on a given state.
- A is the set of transition probabilities.
  - a_{ij} is the probability of transitioning from state *i* to *j*
- B is the set of emission probabilities.
  - b_{ijk} is the probability of outputting the symbol *k* when transitioning from *i* to *j*.

Common notation outside of the HHM machinery

- X is used to denote the sequence of hidden states.
- O is used to denote the sequence of observed emissions.

Fundamental Questions for HMMs

1. Given an HMM, how do we efficiently compute the probability of an output sequence?

$$P(O \mid S, K, \Pi, A, B)$$

2. Given an observation sequence and a model, how do we choose the sequence of hidden states that best explains the observations?

$$P(X \mid O; S, K, \Pi, A, B)$$

3. Given an observation sequence and a space of models, how do we find the best model consistent with our observed data?

$$P(S, K, \Pi, A, B \mid O)$$

Finding the probability of an observation

Given the observation sequence $O=(o_1, \ldots, o_T)$ and a model $\mu= (A, B, \prod)$, we wish to know how to efficiently compute $P(O| \mu)$. This process is called ***decoding***.

For any state sequence X=(X1, …, XT+1), we find: $\boldsymbol{P(O|\mu)=\Sigma_{X1\ldots XT+1}\, \pi_{X1}\, \Pi_{t=1}^{T}\, a_{XtXt+1}\, b_{XtXt+1ot}}$

This is simply the sum of the probability of the observation occurring according to each possible state sequence.

Direct evaluation of this expression, however, is extremely inefficient.

Finding the probability of an observation

In order to avoid this complexity, we can use ***dynamic programming*** or ***memoization*** techniques.

In particular, we use ***treillis*** algorithms.

We make a square array of states versus time and compute the probabilities of being at each state at each time in terms of the probabilities for being in each state at the preceding time.

A treillis can record the probability of all initial subpaths of the HMM that end in a certain state at a certain time. The probability of longer subpaths can then be worked out in terms of the shorter subpaths.

Finding the probability of an observation

A ***forward variable***, $\alpha_i(t) = P(o_1 o_2 \ldots o_{t-1}, X_t = i \mid \mu)$ is stored at (s_i, t)in the trellis and expresses the total probability of ending up in state si at time t.

Forward variables are calculated as follows:

Initialization: $\alpha_i(1) = \pi_i ,\ 1 \le i \le N$

Induction: $\delta_j(t+1) = \Sigma_{i=1}^{N} \alpha_i(t) a_{ij} b_{ijo_t},\ 1 \le t \le T,\ 1 \le j \le N$

Total: $P(O \mid \mu) = \Sigma_{i=1}^{N} \alpha_i(T+1)$

This algorithm requires $2N^2T$ multiplications (much less than the direct method which takes $(2T+1).N^{T+1}$