CSC321: Neural Networks

# Lecture 16: Hidden Markov Models

Geoffrey Hinton

## What does "Markov" mean

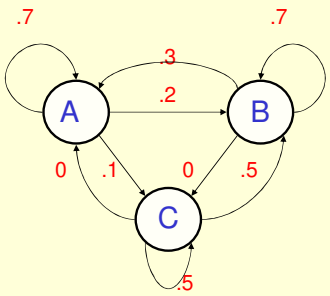
- The next term in a sequence could depend on all the previous terms.
  - But things are much simpler if it doesn't!
  - If it only depends on the previous term it is called "first-order" Markov.
  - If it depends on the two previous terms it is second-order Markov.
- A first order Markov process for discrete symbols is defined by:
  - An initial probability distribution over symbols

  and
  - A transition matrix composed of conditional probabilities

## Two ways to represent the conditional probability table of a first-order Markov process

Current symbol

| Next symbol | A | B | C |
|---|---|---|---|
| A | .7 | .3 | 0 |
| B | .2 | .7 | .5 |
| C | .1 | 0 | .5 |




Typical string: CCBBAAAAABAABACBABAAA

## The probability of generating a string

Product of probabilities, one for each term in the sequence

$$p(\{s_t\}_1^T) = p(s_1)\prod_{t=2}^{T} p(s_t \mid s_{t-1})$$

This comes from the table of initial probabilities

This means a sequence of symbols from time 1 to time T

This is a transition probability

## Learning the conditional probability table

- Naïve: Just observe a lot of strings and set the conditional probabilities equal to observed probabilities
  - But do we really believe it if we get a zero?

$$p(B \mid A) = \frac{\sum_{strings} occurrences\ of\ AB}{\sum_{strings} occurrences\ of\ A}$$

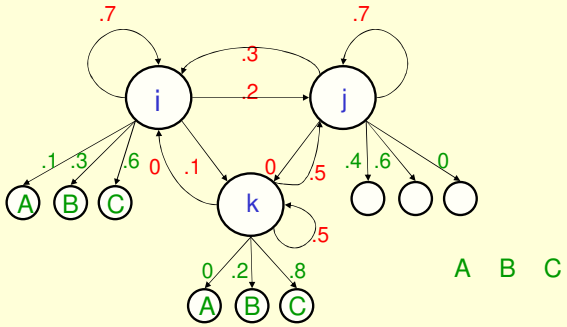
- Better: add 1 to top and number of symbols to bottom. This is like having a weak uniform prior over the transition probabilities.

$$p(B \mid A) = \frac{1 + \sum_{strings} \#AB}{N_{symbols} + \sum_{strings} \#A}$$

## How to have long-term dependencies and still be first order Markov

- We introduce hidden states to get a hidden Markov model:
  - The next hidden state depends only on the current hidden state, but hidden states can carry along information from more than one time-step in the past.
  - The current symbol depends only on the current hidden state.

## A hidden Markov model




A B C

Each hidden node has a vector of transition probabilities and a vector of output probabilities.

## Generating from an HMM

- It is easy to generate strings if we know the parameters of the model. At each time step, make two random choices:
  - Use the transition probabilities from the current hidden node to pick the next hidden node.
  - Use the output probabilities from the current hidden node to pick the current symbol to output.
- We could also generate by first producing a complete hidden sequence and then allowing each hidden node in the sequence to produce one symbol.
  - Hidden nodes only depend on previous hidden nodes
  - So the probability of generating a hidden sequence does not depend on the visible sequence that it generates.

## The probability of generating a hidden sequence

Product of probabilities, one for each term in the sequence

$$p(\{h_t\}_1^T) = p(h_1)\prod_{t=2}^{T} p(h_t \mid h_{t-1})$$

This means a sequence of hidden nodes from time 1 to time T

This comes from the table of initial probabilities of hidden nodes

This is a transition probability between hidden nodes

$$T_{ij} = p(h_t{=}j \mid h_{t-1}{=}i)$$

## The joint probability of generating a hidden sequence and a visible sequence

$$p(\{h_t, s_t\}_1^T) = p(h_1)p(s_1 \mid h_1)\prod_{t=2}^{T} p(h_t \mid h_{t-1})p(s_t \mid h_t)$$

This means a sequence of hidden nodes and symbols too

This is the probability of outputting symbol $s_t$ from node $h_t$

## The probability of generating a visible sequence from an HMM

$$p(\{s_t\}_1^T) = \sum_{H \in \text{hidden paths}} p(\{s_t\}_1^T \mid H)p(H)$$

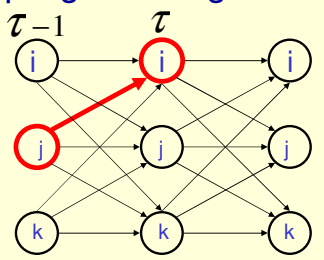
- The same visible sequence can be produced by many different hidden sequences
  - This is just like the fact that the same datapoint could have been produced by many different Gaussians when we are doing clustering.
  - But there are exponentially many possible hidden sequences.
    - It seems hard to figure out $p(\{s_i\}_1^T)$

## The HMM dynamic programming trick

This is an efficient way of computing a sum that has exponentially many terms.

At each time we combine everything we need to know about the paths up to that time into a compact representation:

The joint probability of producing the sequence up to time $\tau$ and using node i at time $\tau$




$$\alpha_i(\tau) = p(\{s_t\}_1^\tau, h_\tau{=}i)$$

This quantity can be computed recursively:

$$\alpha_i(\tau) = p(s_\tau \mid h_\tau{=}i)\sum_j \alpha_j(\tau-1)p(h_\tau{=}i \mid h_{\tau-1}{=}j)$$

## Learning the parameters of an HMM

Its easy to learn the parameters if , for each observed sequence of symbols, we can infer the posterior distribution across the sequences of hidden states

We can infer which hidden state sequence gave rise to an observed sequence by using the dynamic programming trick.