CSC321: Neural Networks

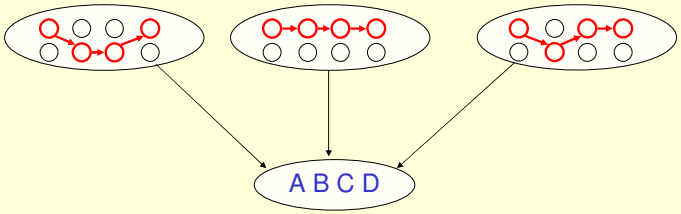
Lecture 17: Learning Hidden Markov Models (with Poritz's notation)

Geoffrey Hinton

The probability of generating a visible sequence from an HMM

$$p(\mathbf{O}) = \sum_{\mathbf{s} \in hidden\ paths} p(\mathbf{O}|\mathbf{s}) p(\mathbf{s})$$

The same visible sequence can be produced by many different hidden sequences



The posterior probability of a hidden path given a visible sequence

$$p(\mathbf{s}|\mathbf{O}) = \frac{p(\mathbf{s}) p(\mathbf{O}|\mathbf{s})}{\sum_{\mathbf{r} \in hidden\ paths} p(\mathbf{r}) p(\mathbf{O}|\mathbf{r})}$$

a hidden path

The sum in the denominator could be computed efficiently using the dynamic programming trick.

But for learning do not need to know about entire hidden paths.

Learning the parameters of an HMM

Its easy to learn the parameters if , for each observed sequence of symbols, we can infer the posterior probability for each hidden node at each time step.

We can infer these posterior probabilities by using the dynamic programming trick.

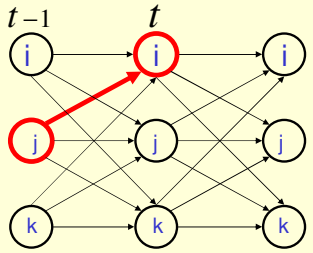
A note on notation:

Poritz uses the notation $s_t = s$

This is confusing because s is being used in two ways. I therefore use $s_t = i$ instead.

Reminder: 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 t and using state i at time t.

$$\alpha_t(i) = p(O_1 \ldots O_t, s_t = i)$$

This quantity can be computed recursively:

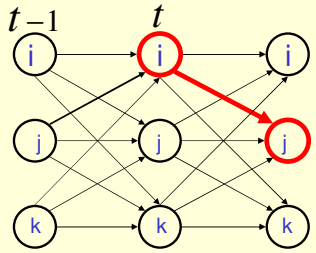
$$\alpha_t(i) = \sum_{h \in \text{hidden states}} \alpha_{t-1}(h)\, a_{hi}\, b_i(O_t)$$

transition prob, output prob

The dynamic programming trick again

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

The conditional probability of producing all of the sequence beyond time t given state i at time t.



$$\beta_t(i) = p(O_{t+1} \ldots O_T \mid s_t = i)$$

This quantity can be computed recursively:

$$\beta_t(i) = \sum_{j \in \text{hidden states}} a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)$$

The forward-backward algorithm (also known as the Baum-Welch algorithm)

- We do a forward pass along the observed string to compute the alpha's at each time step for each node.
- We do a backward pass along the observed string to compute the beta's at each time step for each node.
  - Its very like back-propagation.
- Once we have the alpha's and beta's at each time step, its "easy" to re-estimate the output probabilities and transition probabilities.

The statistics needed to re-estimate the transition probabilities and the output probabilities

- To learn the transition matrix
  - we need to know the expected number of times that each transition between two hidden nodes was used when generating the observed sequence.
- To learn the output probabilities
  - we need to know the expected number of times each node was used to generate each symbol.

The re-estimation equations (the M-step of the EM procedure)

For the transition probability from node i to node j:

$$a_{ij}^{new} = \frac{< i \text{ to } j \text{ transitions when generating the data} >}{\sum_{k \in \text{hidden states}} < i \text{ to } k \text{ transitions when generating the data} >}$$

For the probability that node i generates symbol A:

$$b_i(A) = \frac{< \text{state } i \text{ produces symbol } A \text{ when generating the data} >}{\sum_{B \in \text{symbols}} < \text{state } i \text{ produces symbol } B \text{ when generating the data} >}$$

Summing the expectations over time

- The expected number of times that node i produces symbol A requires a summation over all the different times in the sequence when there was an A.

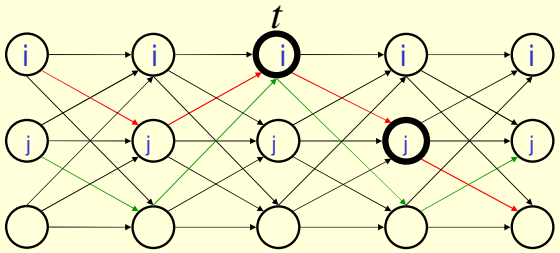
$$< \text{times } i \text{ produces } A >= \sum_{t:O_t=A} p(s_t = i \,|\, \mathbf{O})$$

- The expected number of times that the transition from i to j occurred requires a summation over all pairs of adjacent times in the sequence

$$< \text{transitions from } i \text{ to } j > \; = \sum_{t=1}^{T-1} p(s_t{=}i, s_{t+1}{=}j \,|\, \mathbf{O})$$

Combining the influences of the past and the future to get the full posterior

- To re-estimate the output probabilities, we need to compute the posterior probability of being at a particular hidden node at a particular time.
  - This requires a summation of the posterior probabilities of all the paths that go through that node at that time.



Combining past and future influences efficiently

$$p(\mathbf{O}, s_t = i) \;=\; \alpha_t(i)\, \beta_t(i)$$

$$p(s_t{=}i \,|\, \mathbf{O}) \;=\; \frac{p(\mathbf{O}, s_t{=}i)}{p(\mathbf{O})} \;=\; \frac{\alpha_t(i)\, \beta_t(i)}{p(\mathbf{O})}$$

$$p(s_t{=}i, s_{t+1}{=}j \,|\, \mathbf{O}) \;=\; \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1}) \beta_{t+1}(j)}{p(\mathbf{O})}$$