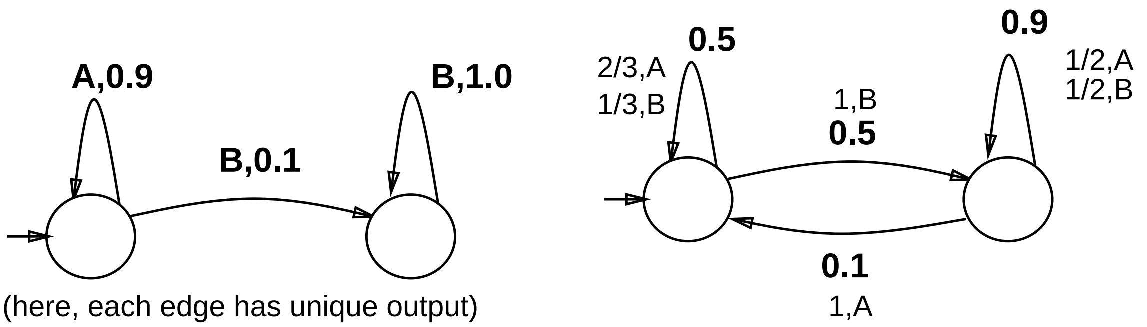
Hidden Markov Models

Process for generating string of characters.



- Set S of states, one marked as initial. (Sometimes also one marked as final)

- Transition probabilities: a_{ij} = prob go to state j given that in state i.

  (For each i, $\sum_j a_{ij} = 1$.)

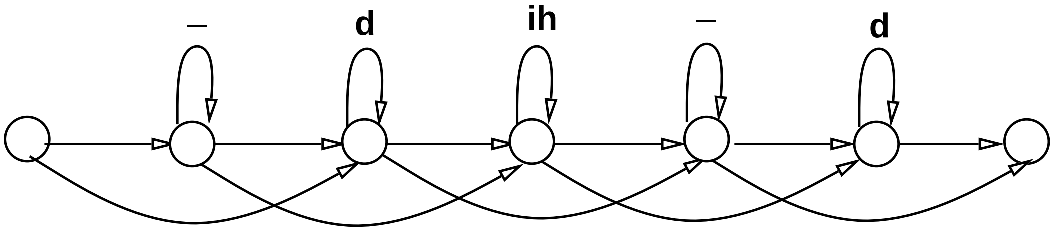
- Output probabilities: $b_{ij}(X)$ = prob of outputting X given that on arc from i to j.

Typical uses:

- model the result of taking an action in an environment.

- model a phoneme, word, or sentence.

Examples



HMM for ‘did’

Typical HMM questions

- "Evaluation Problem": Find P(string | model).

  E.g., say you have HMMs for each of 50 words. You hear some sounds and want the most likely word.

  P(word | sounds) $\propto$ P(sounds | word)$\times$P(word).

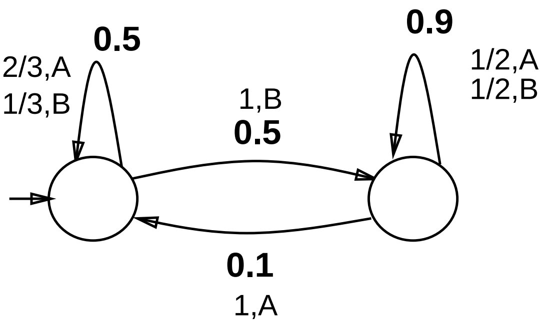
- "Decoding problem": Find most likely path given string, model.

- "Learning Problem": Given a model without the probabilities and an observation, what is the best setting of the probabilities to maximize the likelihood of the observation: P(observation | model).

Evaluation problem: how to calculate P(observation | model)?

One way: for a given model,

$$P(obs) = \sum_{\text{paths } S \text{ of length } |obs|} P(S) \cdot P(obs|S)$$

E.g., "BBA" for model: 

One path is $S =$ stay, stay, stay.

$$P(S) = (0.5)^3, P(obs|S) = (1/3)(1/3)(2/3).$$

Another?

Problem: Could be exponentially many paths.

Evaluation problem: how to calculate P(observation | model)?

Idea: Use dynamic programming approach.

- Calculate P(at time t we're in state i *and* have produced first t characters of observation | model).

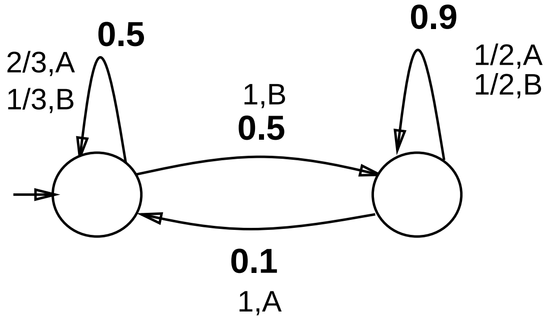
  - called $\alpha_i(t)$.

- $\alpha_i(0)$ is initial distribution.

- $\alpha_j(t) = \sum_i$ [P(in state i at time $t-1$ and have produced first $t-1$ chars) $\times$ P(then go from i to j and output tth char)]

$$= \sum_i \alpha_i(t-1) a_{ij} b_{ij}(obs_t).$$

- Example: "BBA" for



- Final prob $= \sum_i \alpha_i(|obs|)$.

Learning problem (parameter estimation)

Goal: Try to find probabilities that maximize likelihood of observation.

(Could be lots of independent runs, or one big run, or combination)

Method 1: Gradient descent. (Not used much for HMMs)

- Start with some guess and compute P(obs | model).
- Try tweaking values to see if increases or decreases, and hill-climb.
- Problem: slow convergence. Hard to get exact derivatives.

Learning problem contd.

Goal: Try to find probabilities that maximize likelihood of observation.

Method 2: Baum-Welsh/Forward-backward/EM

- Start with some guess of model.

E: Use model & observation to calculate expected number of traversals across each arc.

(and expected # traversals across each arc while producing a given character)

M: Use calculation to update model to model most likely to produce these ratios.

Baum-Welsh, more detail

E: Want to calculate for given model:

* E_{ij} = E[# times cross $i \to j$ | obs].
* $E_{ij}(X)$ = E[# times cross $i \to j$ while producing letter X | obs].

Idea: calculate P[cross $i \to j$ at time t | obs] and add.

(Add over all t for E_{ij}. Add over t such that $obs_t = X$ to get $E_{ij}(X)$.)

Details on next slide...

M: Find what a_{ij}, $b_{ij}(X)$ are most likely to produce these expectations.

* $\bar{a}_{ij} = E_{ij}/\sum_k E_{ik}$.
* $\bar{b}_{ij}(X) = E_{ij}(X)/E_{ij}$.

Baum-Welsh, contd.

Details of E step: (high level idea more important than details)

To calculate P[cross $i \to j$ at time t | obs],

- Rewrite as: $\frac{P\text{[cross } i \to j \text{ at time } t \text{ \& produce obs]}}{P\text{[produce obs]}}$
- Numerator = P[at state i at time t and output first t chars] $\times a_{ij} \times b_{ij}(obs_t) \times$ P[output remaining chars | at state j at time $t+1$].

Theorem: This algorithm will converge to local optimum.