# NLP

# Introduction to NLP

## *Hidden Markov Models (cont'd)*

# Observation Likelihood

- Given multiple HMMs
  - e.g., for different languages
  - Which one is the most likely to have generated the observation sequence
- Naïve solution
  - try all possible state sequences

# Forward Algorithm

- Dynamic programming method
  - Computing a forward trellis that encodes all possible state paths.
  - Based on the Markov assumption that the probability of being in any state at a given time point only depends on the probabilities of being in all states at the previous time point

# HMM Learning

- Supervised
  - Training sequences are labeled
- Unsupervised
  - Training sequences are unlabeled
  - Known number of states
- Semi-supervised
  - Some training sequences are labeled

# Supervised HMM Learning

- Estimate the static transition probabilities using MLE

$$a_{ij} = \frac{Count(q_t = s_i, \mathrm{q}_{\mathrm{t+1}} = s_j)}{Count(q_t = s_i)}$$

- Estimate the observation probabilities using MLE

$$b_j(k) = \frac{Count(q_i = s_j, o_i = v_k)}{Count(q_i = s_j)}$$

- Use smoothing

# Unsupervised HMM Training

- Given:
  - observation sequences
- Goal:
  - build the HMM
- Use EM (Expectation Maximization) methods
  - forward-backward (Baum-Welch) algorithm
  - Baum-Welch finds an approximate solution for $P(O|\mu)$

# Outline of Baum-Welch

- Algorithm
  - Randomly set the parameters of the HMM
  - Until the parameters converge repeat:
    - E step – determine the probability of the various state sequences for generating the observations
    - M step – reestimate the parameters based on these probabilities
- Notes
  - the algorithm guarantees that at each iteration the likelihood of the data $P(O|\mu)$ increases
  - it can be stopped at any point and give a partial solution
  - it converges to a local maximum

# NLP