# Hidden Markov Models
# Baum Welch Algorithm

Introduction to Natural Language Processing
CS 585
Andrew McCallum

March 9, 2004

# Administration

- If you give me your quiz #2, I will give you feedback.
- I'm now giving you quiz #3. Hand it in next class, and we'll give you feedback before the midterm.
- I'm now giving you homework #3. Due one week after Spring Break ends: March 30th; assignment gives many hints about implementation.

  Everyone should be subscribed to class mailing list.
  To: majordomo@cs.umass.edu
  subscribe cs585

# Standard HMM formalism

- $(X, O, A, B)$, $\mu = (A, B)$

- $X$ is hidden state sequence, $O$ is observation sequence
  Probability of starting in some state is folded into $A$, let $x_0$ always be the starting state
  $A$ is matrix of transition probabilities
  $B$ is matrix of output probabilities

$$P(X, O|\mu) = \prod_{t=1}^{N} a[x_{t-1}, x_t] b[o_t, x_t]$$

- HMM is a probabilistic finite state automaton, with probabilistic outputs (from vertices, not arcs, in the simple case; book describes more complex "outputs on arcs".)

# Probabilistic Inference in an HMM

Three fundamental questions for an HMM:

- Compute the probability of a given observation sequence, when the tag sequence is hidden (language modeling)
- Given an observation sequence, find the most likely hidden state sequence (tagging)
- Given observation sequence(s) and a set of states, find the parameters that would make the observations most likely (parameter estimation)

## Calculating the probability of an observation sequence

Given a model $\mu = (A, B)$
we want to find $P(O|\mu)$

$$P(X, O|\mu) = \prod_{t=1}^{N} a[x_{t-1}, x_t] b[o_t, x_t]$$

$$P(O|\mu) = \sum_{X} P(O, X|\mu)$$

Problem: sum is exponential in sequence length!

# Finding probability of observation sequence using dynamic programming

Efficient computation of total probability: forward procedure

Intuition: Probability of the first $t$ observations is the same for all possible $t+1$ length sequences

Define forward probability

$$\alpha_i(t) = P(o_1 o_2 ... o_t, x_t | \mu)$$

$$\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a[x_i, x_j] b[x_j, o_{t+1}]$$

Compute it recursively from the beginning.
(This is a version of variable elimination algorithm for Bayes Net inference.)

# Forward Procedure Recipe

Initialization

$$\alpha_i(1) = a[x_0, x_i]b[x_i, o_1]$$

Induction

$$\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t)a[x_i, x_j]b[x_j, o_{t+1}]$$

Termination
(Note that $\alpha_i(T) = P(o_1...o_T, x_T = i|\mu)$

$$P(o_1...o_T|\mu) = \sum_{i=1}^{N} \alpha_i(T)$$

This is the solution to Problem #1

# Problem #3: Parameter Estimation

We want to find the most likely model parameters given the data (using MLE):

$$\arg\max_{\mu} P(O_{\text{training}}|\mu)$$

This would let us learn model probabilities from raw data

Can't determine these probabilities analytically.

Use iterative hill-climbing algorithm to try to find good model

# HMM training: Baum-Welch reestimation

Used to automatically estimate parameters of an HMM
a.k.a. the Forward-Backward algorithm

A special case of the Expectation Maximization (EM) algorithm

1. Start with initial probability estimates
2. Compute expectations of how often each transition/emission is used
3. Re-estimate the probabilities based on those expectations

...and repeat until convergence

# HMM training: Baum-Welch reestimation

Needed because the state paths are hidden, and the equations cannot be solved analytically.

Provides a maximum likelihood estimates: attempts to find the model that assigns the training data the highest likelihood.

Hill-climbing algorithm that can get stuck in local maxima

Not so effective for inductive POS tagging (the ML re-estimation procedure doesn't know the meaning we have given to the hidden states) But good in many other tasks (speech...)

We need "expected counts" for the E-step!

## Calculating the probability of the observations and a state $i$ at time $t$

Given model $\mu = (A, B)$
we want to find $P(x_t = i, O|\mu)$

$$P(P(x_t = i, O|\mu) = P(o_1 o_2 ... o_t, x_t = i|\mu) P(o_{t+1} o_{t+2} ... o_T | x_t = i, \mu)$$

(Why is this true?)

Remember we have the first part $\alpha_i(t) = P(o_1 o_2 ... o_t, x_t = i|\mu)$.

We need something for the second part: mirror image of the "forward procedure", called "backward procedure."

# Backward procedure recipe

Definition

$$\beta_i(t) = P(o_{t+1}o_{t+2}...o_T|x_t = i, \mu)$$

Initialization

$$\beta_i(T) = 1$$

Induction

$$\beta_i(t) = \sum_{j=1}^{N} a[x_i, x_j]b[x_j, o_{t+1})\beta_j(t+1)$$

## Probability of a state $i$ at time $t$

$$\begin{aligned} P(x_t = i, O|\mu) &= P(o_1 o_2 ... o_t, x_t = i|\mu) P(o_{t+1} o_{t+2} ... o_T | x_t = i, \mu) \\ &= \alpha_i(t) \beta_i(t) \end{aligned}$$

$$P(x_t = i|O, \mu) = \frac{P(x_t = i, O|\mu)}{P(O|\mu)} = \gamma_i(t)$$

## Probability of a transition from state $i$ to state $j$ at time $t$

The probability of a trajectory being in state $x_i$ at time $t$ and making the transition to $s_j$ at $t+1$ given the observation sequence and model.

$$\xi_t(i,j) = P(x_t = i, x_{t+1} = j | O, \mu)$$

We compute these probabilities using the forward and backward variables.

$$\xi_t(i,j) = \frac{\alpha_i(t) a[x_i x_j] b[x_j, o_{t+1}] \beta_j(t+1)}{Pr(O|\mu)}$$

# Expected transition and emission counts

Note that (E-step)

$$\sum_{t=1}^{T} \gamma_i(t) \quad = \quad \text{expected number of transitions from } x_i$$

$$\sum_{t=1}^{T} \xi_t(i,j) \quad = \quad \text{expected number of transitions from } x_i \text{ to } x_j$$

Then we can estimate parameters by ratio of expected counts (M-step)

$$\bar{a}[x_i, x_j] = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1} T - 1 \gamma_j(t)}$$

$$\bar{b}[x_i, o_k] \frac{\sum_{t=1}^{T-1} \gamma_j(t)\ 1(o_t = k)}{\sum_{t=1}^{T-1} \gamma_j(t)}$$

# Baum-Welch training algorithm

- Begin with some model $\mu$ (perhaps random, perhaps preselected)
- Run $O$ through the current model to estimate the expectations of each model parameter.
- Change the model to maximize the values of the paths that are used a lot (while still repsecting the stochastic constraints).
- Repeat, hoping to converge on optimal values for the model parameters, $\mu$.

# Baum-Welch tips and tricks: normalization

$\alpha$ and $\beta$ values can get very small. On-the-fly re-normalization badly needed.

Normalize $\alpha, \beta$ using the same normalization factor

$$Z(t) = \sum_{i=1}^{N} \alpha_i(t)$$

Then adjust the $\alpha, \beta$ across all states after each time step

$$\alpha_i(t)^* = \alpha_i(t)/Z(t)$$

$$\beta_i(t)^* = \beta_i(t)/Z(t)$$

# HMM final remarks

- Parameter "tying" (keep just one counter and parameter across several states or transitions.

  Any combination possible.

  Reduces capacity, and thus over-fitting

- Real number output: Emissions represented by a Gaussian distribution.

- Empty (epsilon) transitions, do not generate output.