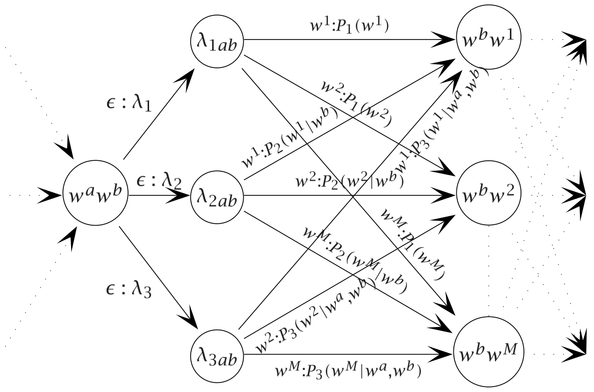
## Some of an HMM for an interpolated language model






## The third problem: Parameter estimation = Parameter learning

- 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 case of the Expectation Maximization (EM) algorithm
- One starts with initial probability estimates
- One computes expectations of how often each transition/emission is used
- One re-estimates 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 estimate: 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 reestimation procedure doesn't know the meaning we have given to the hidden states)
- But good in many tasks (speech, including information extraction)



## Probability of an observation sequence

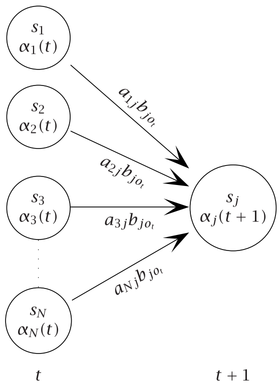
- Given $O = (o_1, \ldots, o_T)$ and model $\mu = (A, B, \Pi)$
- $P(O|X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \cdots b_{x_T o_T}$
- $P(X|\mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \cdots a_{X_{T-1} X_T}$
- $P(O, X|\mu) = P(O|X, \mu) P(X|\mu)$
- $P(O|\mu) = \sum_X P(O|X, \mu) P(X|\mu)$ [Marginalization]
- $P(O|\mu) = \sum_{X_1 \cdots X_T} \prod_{t=1}^{T} a_{X_{t-1} X_t} b_{x_t o_t}$
- Again, difficult to compute like this!



## Dynamic Programming

- Efficient computation of this quantity: forward procedure
- Intuition: Probability of the first $t$ observations is the same for all possible $t + 1$ length state sequences.
- Define forward probability
$\alpha_i(t) = P(o_1 o_2 \cdots o_{t-1}, X_t = i|\mu)$
- $\alpha_j(t + 1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{i o_t}$
- Compute it recursively from beginning
- A version of the variable elimination algorithm for Bayes Net inference

## Closeup of the computation at one node




$$\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{jo_t}$$



## Dynamic Programming (2)

- Similarly, calculate backward probability $\beta_i(t)$ from end
- The probability of the rest of the observations given a middle state
- $\beta_i(t) = P(o_t \cdots o_T | X_t = i)$
- $\beta_i(T+1) = 1$
- $\beta_i(t) = \sum_{j=1,\ldots,N} a_{ij} b_{io_t} \beta_j(t+1)$
- Forward: $P(O|\mu) = \sum_{i=1}^{N} \alpha_i(T)$
- Backward: $P(O|\mu) = \sum_{i=1}^{N} \pi_i \beta_i(1)$
- Combination: $P(O|\mu) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t), \quad$ for any $t$



## EM Algorithm Intuition

- We don't know what the model is.
- But we can work out the probability of the observation sequence using some (perhaps random) model
- Looking at that calculation, we can see which state transitions and symbol emissions were probably used the most
- By increasing the probability of those, we can choose a revised model which gives a higher probability to the observation sequence



## The likelihood of being in state $i$ at time $t$

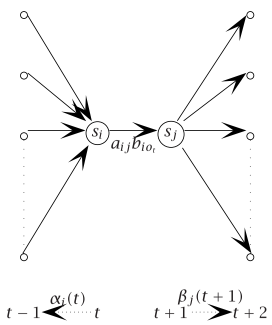
Using forward and backward variables, we can calculate $\gamma_i(t)$, the probability of being in state $i$ at time $t$:

$$\begin{aligned}\gamma_i(t) &= P(X_t = i | O, \mu) \\ &= \frac{P(X_t = i, O|\mu)}{P(O|\mu)} \\ &= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}\end{aligned}$$



## Chance of moving from state $i$ to $j$ at time $t$

Define $p_t(i,j)$, the probability of traversing an arc $i - j$ at time $t$ given observations $O$.






## Computing probability of traversing arc

$$\begin{aligned}p_t(i,j) &= P(X_t = i, X_{t+1} = j | O, \mu) \\ &= \frac{P(X_t=i, X_{t+1}=j, O|\mu)}{P(O|\mu)} \\ &= \frac{\alpha_i(t) a_{ij} b_{io_t} \beta_j(t+1)}{\sum_{m=1}^{N} \alpha_m(t)\beta_m(t)} \\ &= \frac{\alpha_i(t) a_{ij} b_{io_t} \beta_j(t+1)}{\sum_{m=1}^{N}\sum_{n=1}^{N} \alpha_m(t) a_{mn} b_{mo_t} \beta_n(t+1)}\end{aligned}$$

Note that $\gamma_i(t) = \sum_{j=1}^{N} p_t(i,j)$.

## Expectations

Now, if we sum over the time index, this gives us expectations (counts):

$\sum_{t=1}^{T} \gamma_i(t)$ = expected number of transitions from state $i$ in $O$

$\sum_{t=1}^{T} p_t(i,j)$ = expected number of transitions from state $i$ to $j$ in $O$



## Parameter reestimation formulae

$$\begin{aligned}
\hat{\pi}_i &= \text{expected frequency in state } i \text{ at time } t = 1 \\
&= \gamma_i(1) \\
\hat{a}_{ij} &= \frac{\text{expected num. transitions from state } i \text{ to } j}{\text{expected num. transitions from state } i} \\
&= \frac{\sum_{t=1}^{T} p_t(i,j)}{\sum_{t=1}^{T} \gamma_i(t)} \\
\hat{b}_{ik} &= \frac{\text{expected num. times } k \text{ observed in state } i}{\text{expected num. transitions from } i} \\
&= \frac{\sum_{\{t:o_t=k,1\le t\le T\}} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}
\end{aligned}$$



## 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 respecting the stochastic constraints)
- Repeat, hoping to converge on optimal values for the model parameters $\mu$.



## We're guaranteed to get no worse

From $\mu = (A, B, \Pi)$, one iteration derives $\hat{\mu} = (\hat{A}, \hat{B}, \hat{\Pi})$.

Baum initially proved for HMMs, and the EM framework in general gives us that:

$$P(O|\hat{\mu}) \ge P(O|\mu)$$

But we may stay in a local optimum.