Training Discrete Observation HMMs

Training refers to the problem of finding $\{\pi(1), \boldsymbol{A}, \boldsymbol{B}\}$ such that the model, M, after an iteration of training, better represents the training data than the previous model. The number of states is usually not varied or reestimated, other than via the modification of the model inventory. The apriori probabilities of the likelihood of a model, $\pi(1)$, are normally not reestimated as well, since these typically come from the language model.

The first algorithm we will discuss is one based on the **Forward-Backward** algorithm (Baum-Welch Reestimation):

$$u_{j|i} \equiv \text{label for a transition from state } i \text{ to state } j$$

$$u_{\bullet|i} \equiv \text{set of transitions exiting state } i$$

$$u_{j|\bullet} \equiv \text{set of transitions entering } j$$

Also, $\underline{u}(t)$ denotes a random variable that models the transitions at time t and $\underline{y}_j(t)$ a random variable that models the observation being emitted at state j at time t. The symbol "•" is used to denote an arbitrary event.

Next, we need to define some intermediate quantities related to particular events at a given state at a given time:

$$\begin{aligned}\zeta(i, j; t) &\equiv P(\underline{u}(t) = u_{j|i} \big| y, M) \\ &= P(\underline{u}(t) = u_{j|i}, y \big| M) / P(y|M) \\ &= \left\{\begin{array}{ll} \dfrac{\alpha(y_1^t, i) a(j|i) b(y(t+1)|j) \beta(y_{t+2}^T \big| j)}{P(y|M)}, & t = 1, 2, \ldots, T \\ 0\,, & \text{other } t \end{array}\right\}\end{aligned}$$

where the sequences α, β, a, and b were defined previously (last lecture). Intuitively, we can think of this as the probability of observing a transition from state i to state j at time t for a particular observation sequence, y, (the utterance in progress), and model M.

We can also make the following definition:

$$\gamma(i;t) \equiv P(\underline{u}(t) \in u_{\bullet|i} \big| y, M) = \sum_{j=1}^{S} \zeta(i, j;t)$$

$$= \left\{ \begin{array}{ll} \dfrac{\alpha(y_1^t, i)\beta(y_{t+1}^T \big| i)}{P(y|M)}, & t = 1, 2, \ldots, T \\ 0, & \text{other } t \end{array} \right\}$$

This is the probability of exiting state i. Also,

$$\nu(j;t) \equiv P(\underline{x}(t) = j | y, M)$$

$$= \left\{ \begin{array}{ll} \gamma(j;t), & t = 1, 2, \ldots, T \\ \alpha(y_1^T, j), & t = T \\ 0, & \text{other } t \end{array} \right\}$$

$$= \left\{ \begin{array}{ll} \dfrac{\alpha(y_1^t, j)\beta(y_{t+1}^T \big| j)}{P(y|M)}, & t = 1, 2, \ldots, T \\ 0, & \text{other } t \end{array} \right\}$$

which is the probability of being in state j at time t. Finally,

$$\delta(j,k;t) \equiv P(\underline{y}_j(t) = k \big| y, M)$$

$$= \left\{ \begin{array}{ll} \nu(j;t), & \text{if } y(t) = k \text{ and } 1 \le t \le T \\ 0, & \text{otherwise} \end{array} \right\}$$

$$= \left\{ \begin{array}{ll} \dfrac{\alpha(y_1^t, j)\beta(y_{t+1}^T \big| j)}{P(y|M)}, & \text{if } y(t) = k \text{ and } 1 \le t \le T \\ 0, & \text{otherwise} \end{array} \right\}$$

which is the probability of observing symbol k at state j at time t.

Note that we make extensive use of the forward and backward probabilities in these computations. This will be key to reducing the complexity of the computations by allowing an interactive computation.

From these four quantities, we can define four more intermediate quantities:

$$\zeta(i, j;\bullet) = P(\underline{u}(\bullet) \in u_{j|i} \big| y, M) = \sum_{t=1}^{T} \zeta(i, j;t)$$

$$\gamma(i;\bullet) = P(\underline{u}(\bullet) \in u_{\bullet|i} \big| y, M) = \sum_{t=1}^{T} \gamma(i;t)$$

$$\nu(j;\bullet) = P(\underline{u}(\bullet) \in u_{j|\bullet} \big| y, M) = \sum_{t=1}^{T} \nu(j;t)$$

$$\delta(j, k;\bullet) = P(\underline{y}_j(\bullet) = k \big| y, M) = \sum_{t=1}^{T} \delta(j, k;t) = \sum_{\substack{t = 1 \\ y(t) = k}}^{T} \nu(j;t)$$

Finally, we can begin relating these quantities to the problem of reestimating the model parameters. Let us define four more random variables:

$$\underline{n}(u_{j|i}) \equiv \text{number of transitions of the type } u_{j|i}$$

$$\underline{n}(u_{\bullet|i}) \equiv \text{number of transitions of the type } u_{\bullet|i}$$

$$\underline{n}(u_{j|\bullet}) \equiv \text{number of transitions of the type } u_{j|\bullet}$$

$$\underline{n}(\underline{y}_j(\bullet) = k) \equiv \text{number of times the observation } k \text{ and state } j \text{ jointly occur}$$

We can see that:

$$\zeta(i, j;\bullet) = E\{\underline{n}(u_{j|i}) \big| y, M\}$$

$$\gamma(i;\bullet) = E\{\underline{n}(u_{\bullet|i}) \big| y, M\}$$

$$\nu(j;\bullet) = E\{\underline{n}(u_{j|\bullet}) \big| y, M\}$$

$$\delta(j, k;\bullet) = E\{\underline{n}(\underline{y}_j(\bullet) = k) \big| y, M\}$$

What we have done up to this point is to develop expressions for the estimates of the underlying components of the model parameters in terms of the state sequences that occur during training.

But how can this be when the internal structure of the model is **hidden**?

Following this line of reasoning, an estimate of the transition probability is:

$$\bar{a}(j|i) = \frac{E\{\underline{n}(u_{j|i})\big|y,M\}}{E\{\underline{n}(u_{\bullet|i})\big|y,M\}} = \frac{\zeta(i,\, j;\bullet)}{\gamma(i;\bullet)}$$

$$= \frac{\sum_{t=1}^{T-1} \alpha(y_1^t, i)a(j|i)b(y(t+1)|j)\beta(y_{t+2}^T\big|j)}{\sum_{t=1}^{T-1} \alpha(y_1^t, i)\beta(y_{t+1}^T\big|i)}$$

Similarly,

$$\bar{b}(k|j) = \frac{E\left\{\underline{n}(\underline{n}(\underline{y}_j(\bullet) = k)\big|y,M)\big|y,M\right\}}{E\{\underline{n}(u_{j|\bullet})\big|y,M\}} = \frac{\zeta(i,\, j;\bullet)}{\gamma(i;\bullet)}$$

$$= \frac{\sum_{\substack{t=1 \\ y(t)=k}}^{T} \alpha(y_1^t, j)\beta(y_t^T\big|j)}{\sum_{t=1}^{T} \alpha(y_1^t, j)\beta(y_{t+1}^T\big|j)}$$

Finally,

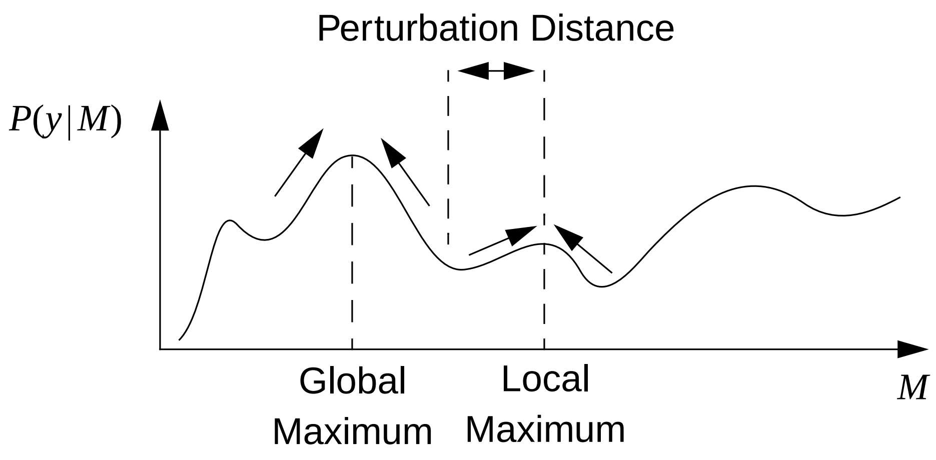
$$P(\underline{x}(1) = i) = \frac{\alpha(y_1^1, i)\beta(y_2^T\big|i)}{P(y|M)}$$

This process is often called reestimation by recognition, because we need to recognize the input with the previous models in order that we can compute the new model parameters from the set of state sequences used to recognize the data (hence, the need to iterate).

But will it converge? Baum and his colleagues showed that the new model guarantees that:

$$P(y|\overline{M}) \geq P(y|M)$$

Since this is a highly nonlinear optimization, it can get stuck in local minima:



We can overcome this by starting training from a different initial point, or “bootstrapping” models from previous models.

Analogous procedures exist for the **Viterbi algorithm**, though they are much simpler and more intuitive (and more DP-like):

$$\bar{a}(j|i) = \frac{E\{\underline{n}(u_{j|i})|y,M\}}{E\{\underline{n}(u_{\bullet|i})|y,M\}}$$

and,

$$\bar{b}(k|j) = \frac{E\{\underline{n}(u_{j|i})|y,M\}}{E\{\underline{n}(u_{\bullet|i})|y,M\}}$$

These have been shown to give comparable performance to the forward-backward algorithm at significantly reduced computation. It also is generalizable to alternate formulations of the topology of the acoustic model (or language model) drawn from formal language theory. (In fact, we can even eliminate the first-order Markovian assumption.)

Further, the above algorithms are easily applied to many problems associated with language modeling: estimating transition probabilities and word probabilities, efficient parsing, and learning hidden structure.

But what if a transition is never observed in the training database?