L24: Baum-Welch and Entropic Training

The Baum-Welch re-estimation procedure

Implementation issues

Continuous and semi-continuous HMMs

Types of HMM structure

Entropic training

This lecture is based on [Rabiner and Juang, 1993]

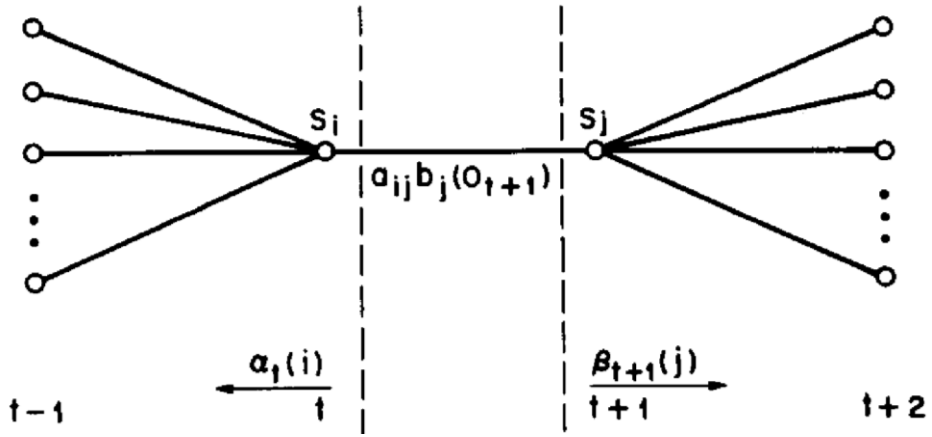
Baum-Welsh re-estimation

Problem 3: Parameter estimation

- The most important and difficult problem in HMMs is to estimate model parameters $\lambda = \{A, B, \pi\}$ from data
  - HMMs are trained with a Maximum Likelihood criterion: seek model parameters λ that best explain the observations, as measured by $P(O|\lambda)$
  - This problem is solved with an iterative procedure known as *Baum-Welch*, which is an implementation of the EM algorithm we discussed earlier
- As usual, we begin by defining a new variable, $\xi_t(i,j)$

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda)$$

  - which is the probability of being in S_i at time t, and S_j at time $t+1$



[Rabiner, 1989]

- From the definition of $\alpha_t(i)$, $\beta_t(i)$ and conditional probability:

$$\xi_t(i,j) = \frac{P(q_t = S_i, q_{t+1} = S_j, O|\lambda)}{P(O|\lambda)} = \frac{\alpha_t(i)\, a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j)}{P(O|\lambda)}$$
$$= \frac{\alpha_t(i)\, a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)\, a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j)}$$

- Intuitive interpretation of $\gamma_t(i)$ and $\xi_t(i,j)$
  - First note that, since $\gamma_t(i)$ is the probability of being in state S_i at time t given observation sequence O and model λ, $\xi_t(i,j)$ can be related to $\gamma_t(i)$ by

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

  - The sum of $\gamma_t(i)$ over time may be interpreted as the expected number of times that state S_i is visited or, excluding time $t = T$, the number of transitions from S_i

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{"expected number of transitions from state } S_i \text{ in O"}$$

  - Similarly, summation of $\xi_t(i,j)$ from $t = 1$ to $t = T - 1$ may be interpreted as the expected number of transitions from state S_i to state S_j

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{"expected number of transitions from state } S_i \text{ to state } S_j\text{"}$$

Re-estimation procedure

- Using this line of reasoning, we can produce a method to iteratively update the parameters of an HMM by simply "counting events"

$$\hat{\pi}_i = \text{"expected frequency (number of times) in state } S_i \text{ at time } (t=1)\text{"} = \gamma_1(i)$$

$$\hat{a}_{ij} = \frac{\text{"expected number of transitions from } S_i \text{ to } S_j\text{"}}{\text{"expected number of transitions from } S_i\text{"}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\hat{b}_j(k) = \frac{\text{"expected number of times in } S_j \text{ and obseving } v_k\text{"}}{\text{"expected number of times in } S_j\text{"}} = \frac{\sum_{\substack{t=1 \\ s.t. o_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

  - where the rhs of the equations is computed from the "old" parameter values, and the lhs are the re-estimated "new" parameters

- It can be shown that each iteration of this procedure increases the likelihood of the data until a local minimum is found

$$P\left(O|\lambda^{(new)}\right) \geq P\left(O|\lambda^{(old)}\right)$$

  - This property is due to the fact that Baum-Welch is an implementation of the Expectation-Maximization algorithm

Baum-Welch is "simply" an implementation of the EM algorithm where

- The observation sequence $O = \{o_1, o_2, o_3 \ldots\}$ is the observed data
- The underlying state seq. $Q = \{q_1, q_2, q_3 \ldots\}$ is the missing or hidden data
- The incomplete-data likelihood is given by $P(O|\lambda)$
- The complete-data likelihood is $P(O, Q|\lambda)$

Therefore, the auxiliary Q function from EM becomes

$$Q\left(\theta|\theta^{(i-1)}\right) = E_Z\left[logp(X, Z|\theta)|X, \theta^{(i-1)}\right] \Rightarrow$$
$$Q\left(\lambda|\lambda^{(i-1)}\right) = E_Q\left[logp(O, Q|\lambda)|O, \lambda^{(i-1)}\right]$$

- from which the expected value $E_Q[\cdot]$ is computed by averaging over all state sequences

$$Q\left(\lambda|\lambda^{(i-1)}\right) = E_Q\left[logp(O, Q|\lambda)|O, \lambda^{(i-1)}\right] = \sum_{\forall q} logp(O, Q|\lambda)p(O, Q|\lambda)$$

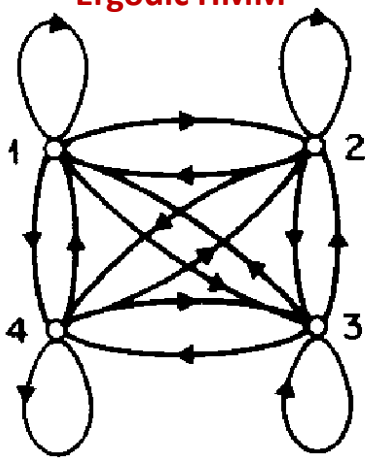
- The re-estimation formulas in the previous page can also be obtained from this auxiliary function
  - Details on this derivation can be found in [Rabiner and Juang, 1993; Bilmes, 1998]

Types of HMM structure

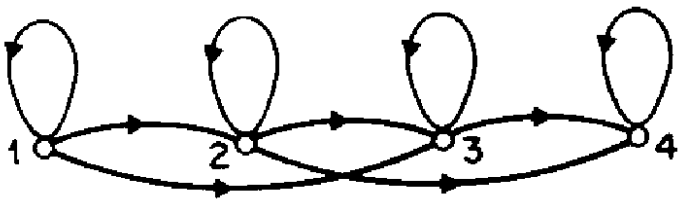
Ergodic vs. left-right HMMs

- An ergodic HMM is a fully connected model, where each state can be reached in one step from every other state
  - This is the most general type of HMM, and the one that has been implicitly assumed in the previous derivations
- A left-right or Bakis model is one where no transitions are allowed to states whose indices are lower than the current state: $a_{ij} = 0;\ \forall j < i$
  - Left-right models are best suited to model signals whose properties change over time, such as speech
  - When using left-right models, some additional constraints are used, such as preventing large transitions: $a_{ij} = 0;\ \forall j < i + \Delta$ ($\Delta = 3$ in the example below)

Ergodic HMM



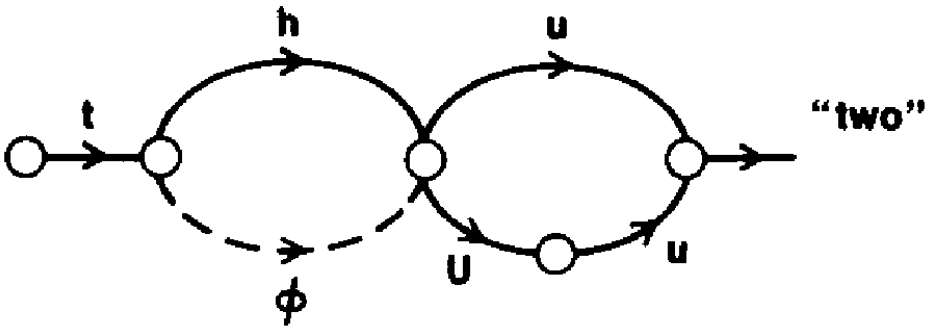
Left-right HMM



[Rabiner, 1989]

Null transitions

- In the HMM models presented in these lectures, observations are associated with the states. A number of people (IBM, CMU) have used HMM models where the observations are associated with the transitions between states
- In this type of models, it has been found useful to allow transitions that produce no observations. These are called *null* transitions
- In the example below, an HMM with null transition ϕ is used to model two different pronunciations for the English word "two"



[Rabiner, 1989]

Implementation issues for HMMs

Scaling

- Since $\alpha_t(i)$ involves the product of a large number of terms that are less than one, the machine precision is likely to be exceeded at some point in the computation
- To solve this problem, the $\alpha_t(i)$ are re-scaled periodically (e.g., every iteration t) to avoid underflow. A similar scaling is done to the $\beta_t(i)$ so that the scaling coefficients cancel out exactly

Multiple observation sequences

- The HMM derivation in these lectures is based on a single observation sequence. This becomes a problem in left-right models, since the transient nature of the states only allows a few observations to be used for each state
- For this reason, one has to use multiple observation sequences. Re-estimation formulas for multiple sequences can be found in [Rabiner and Juang, 1993]

Initial parameter estimates

- How are the initial HMM parameters chosen so that the local maximum to which Baum-Welch converges to is actually the global maximum?
- Random or uniform initial values for π and A have experimentally been found to work well in most cases
- Careful selection of initial values for B, however, has been found to be helpful in the discrete case and essential in the continuous case. These initial estimates may be found by segmenting the sequences with k-means clustering

The discussion thus far has focused on discrete HMMs

- Discrete HMMs assume that the observations are defined by a set of discrete symbols from a finite alphabet
- In most pattern recognition applications, however, observations are inherently multidimensional and having continuous features

There are two alternatives to handle continuous vectors with HMMs

- Convert the continuous multivariate observations into discrete univariate observations via a codebook (e.g., cluster the observations with k-means)
  - This approach, however, may lead to degraded performance as a result of the discretization of the continuous signals
- Employ HMM states that have continuous observation densities $b_j(\cdot)$
  - This is, in general, a much better alternative, which we explore next

Continuous HMMs model the observation probabilities with a continuous density function, as opposed to a multinomial

- To ensure that model parameters can be re-estimated in a consistent manner, some restrictions are applied to the observation pdf
- The most common form is the Gaussian mixture model of L14

$$b_j(o) = \sum_{k=1}^{M} c_{jk} N\left(o, \mu_{jk}, \Sigma_{jk}\right)$$

  - where o is the observation vector, and c_{jk}, μ_{jk} and Σ_{jk} are the mixture coefficient, mean and covariance for the k-th Gaussian component at state S_j, respectively

The re-estimation formulas for the continuous case generalize very gracefully from the discrete HMM

- The term $\gamma_t(j)$ generalizes to $\gamma_t(j,k)$, which is the probability of being in state S_j at time t with the k-th mixture component accounting for observation o_t

$$\gamma_t(j,k) = \underbrace{\left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)}\right]}_{\substack{same\ as\ in \\ discrete\ HMMs}} \underbrace{\left[\frac{c_{jk}N\left(o_t, \mu_{jk}, \Sigma_{jk}\right)}{\sum_{m=1}^{M} c_{jm}N\left(o_t, \mu_{jm}, \Sigma_{jm}\right)}\right]}_{\substack{term\ due\ to \\ k^{th}\ Gaussian}}$$

The re-estimation formulas for the continuous HMM become

$$\bar{c}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_t(j,k)}; \qquad \bar{\mu}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)\, o_t}{\sum_{t=1}^{T} \gamma_t(j,k)};$$

$$\bar{\Sigma}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)\left(o_t - \mu_{jk}\right)\left(o_t - \mu_{jk}\right)^t}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

- The re-estimation formula for c_{jk} is the ratio between the expected number of times the system is in state S_j using the k-th mixture component, and the expected number of times the system is in state S_j
- The re-estimation formula for the mean vector μ_{jk} weights the numerator in the equation for c_{jk} by the observation, to produce the portion of the observation that can be accounted by that mixture component
  - The re-estimation formula for the covariance term can be interpreted similarly
- The re-estimation formula for the transition probabilities aij is the same as in the discrete HMM

Semi-continuous HMMs

Continuous HMMs avoid the distortions introduced by a discrete codebook, but this comes at a price

- A large number of mixtures are generally required to improve the recognition accuracy as compared to D-HMMs [Huang, 1992]
- As a result, the computational complexity of C-HMMs increases considerably with respect to D-HMMs
- In addition, the number of free parameters increases significantly, which means that a larger amount of training data is required to properly train the model

Semi-continuous HMMs (SC-HMMs) represent a compromise between discrete and continuous HMMs

- In SC-HMMs, the observation space is modeled with a Gaussian mixture whose components (μ, Σ) are shared by all HMM states
- Each state in the HMM, though, is allowed to have a different mixing coefficient c_{jk} for each of the k Gaussian components in the "common" mixture

Entropic training

Selecting the HMM model structure

- Given that the process being modeled by an HMM is hidden, how can an appropriate model structure be selected?
  - In most cases, this is achieved by training several models with different structures and selecting the best one through cross-validation
- Nonetheless, even after an appropriate model is selected, conventional training (Baum-Welch) leads to HMMs that are too ambiguous, too difficult to interpret
  - In an HMM it is not rare to find many slightly different state sequences that are virtually equally likely. The Viterbi sequence, for instance, may represent only a small fraction of the total probability mass

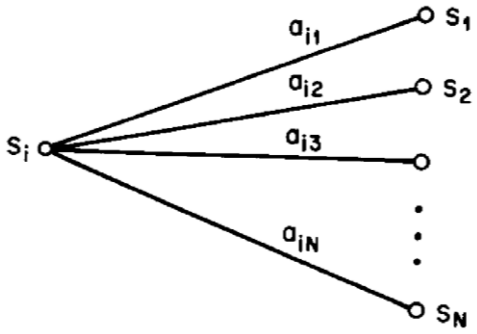
An alternative procedure, known as entropic training, can be used to learn sparse HMM models

- Conventional HMM training (Baum-Welch) is based on a Maximum Likelihood criterion: find model parameters $\lambda = \{A, B, \pi\}$ that maximize the likelihood of the observation sequence $P(O|\lambda)$
- Entropic training is based on a MAP criterion $\lambda = argmax\ P(\lambda|O)$ with a prior term $P(\lambda)$ that favors low-entropy multinomials

[Brand, 1998]

The prior term employed by entropic training is expressed by

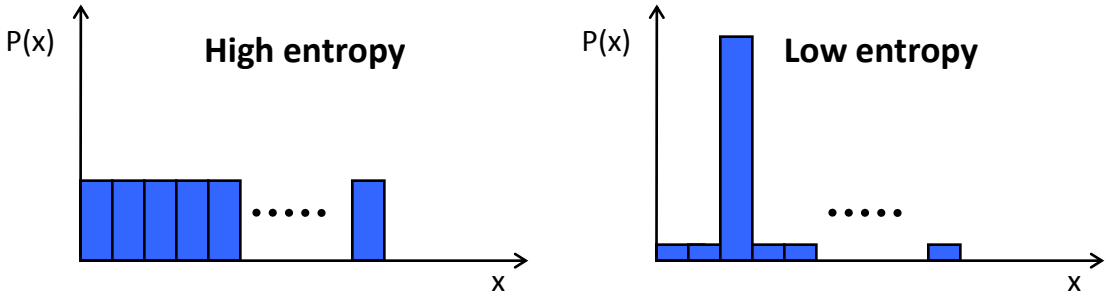
$$P_e(\theta) = e^{-H(\theta)} = \exp\left[\sum_{i=1}^{N} \theta_i \log \theta_i\right] = \prod_{i=1}^{N} \theta_i^{\theta_i}$$



- where θ_i are multimodal parameters, such as the set of transition probabilities a_{ij} from a state, or the mixture coefficient in a GMM

This prior favors multinomials that have low entropy $H(\theta)$

- The highest entropy multinomial is a uniform histogram
  - This is called a "non informative" prior because it does not tell us anything about the parameter value
- The lowest entropy corresponds to a histogram where all but one bin are zero
  - This histogram has no uncertainty: only one parameter value is possible



Assume that you are given a collection of events $\{\omega_i\}$, where ω_i is the # occurrences of the i-th event in the multinomial

- The likelihood of the collection of events ω_i given multinomial θ_i is

$$P(\omega|\theta) = \prod_{i=1}^{N} \theta_i^{\omega_i}$$

Merging the entropic prior with the posterior leads to the following MAP objective function

$$\underbrace{P(\theta|\omega)}_{posterior} \propto \underbrace{P_e(\theta)}_{prior} \underbrace{P(\omega|\theta)}_{likelihood} \propto \left[\prod_{i=1}^{N} \theta_i^{\theta_i}\right]\left[\prod_{i=1}^{N} \theta_i^{\omega_i}\right] = \prod_{i=1}^{N} \theta_i^{\theta_i+\omega_i}$$

- The MAP solution represents a compromise between the prior and the likelihood
  - If there is sufficient training data, the term $\theta_i + \omega_i$ is dominated by ω_i (note that ω_i represents an event "count", whereas θ_i is a probability,) and the MAP solution converges to the Maximum Likelihood solution
  - If the training data is scarce, the term $\theta_i + \omega_i$ will be dominated by θ_i, and the MAP solution will converge to the Minimum Entropy solution

– To find the optimal model parameters θ_i, we set the derivative of the log-posterior to zero using a Lagrange multiplier ρ to ensure $\sum_i \theta_i = 1$

$$\frac{\partial log P(\theta|\omega)}{\partial \theta_i} = \frac{\partial}{\partial \theta_i}\left[log \prod_{i=1}^{N} \theta_i^{\theta_i+\omega_i} + \rho\left(\sum_{i=1}^{N} \theta_i - 1\right)\right] = 0$$

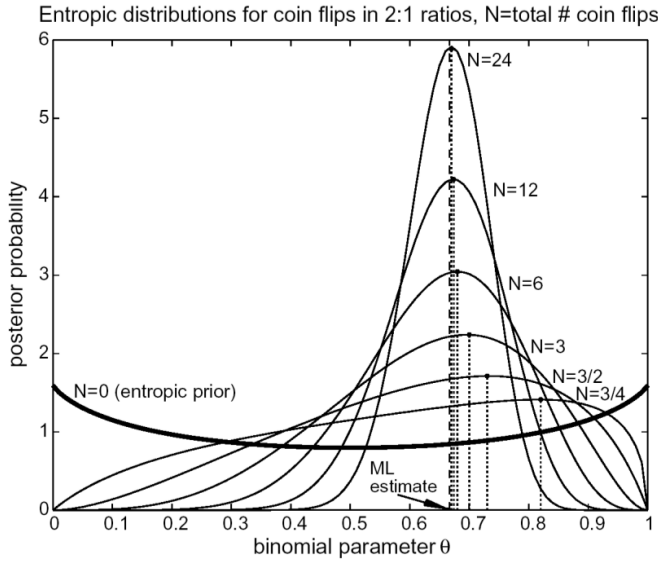
$$\sum_{i=1}^{N} \frac{\partial}{\partial \theta_i}\left[(\theta_i + \omega_i)\log \theta_i\right] + \rho \sum_{i=1}^{N} \frac{\partial}{\partial \theta_i}\theta_i = 0$$

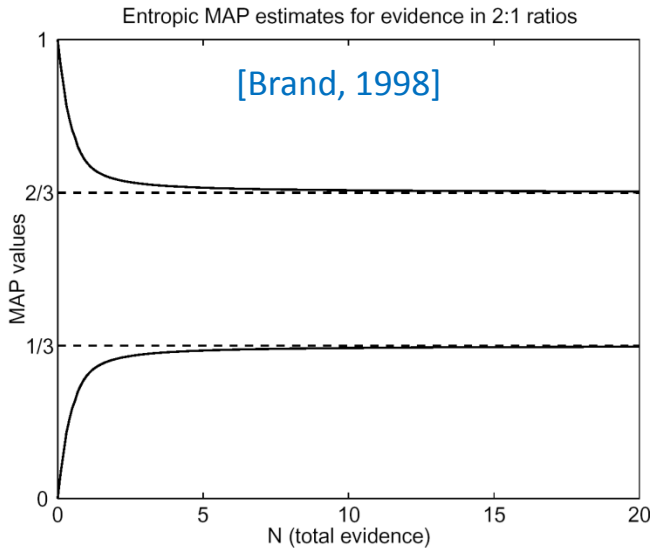
$$\frac{\omega_i}{\theta_i} + \log \theta_i + 1 + \rho = 0$$

– This last expression defines a system of non-linear equations, whose solution can be found in [Brand, 1998]

Examples

- The right viewgraph shows the posterior of a binomial (e.g., a coin toss experiment) where heads occur twice as often as tails, and $\theta = P(H)$
- In the absence of data, the posterior favors minimum entropy: either $\theta = 0$ or $\theta = 1$
- As the number of coin-tosses increases, the maximum of the posterior becomes closer to the ML solution $\theta = 2/3$
- The left viewgraph shows the asymptotic evolution of the MAP parameter estimates as the number of examples increases to $N \to \infty$



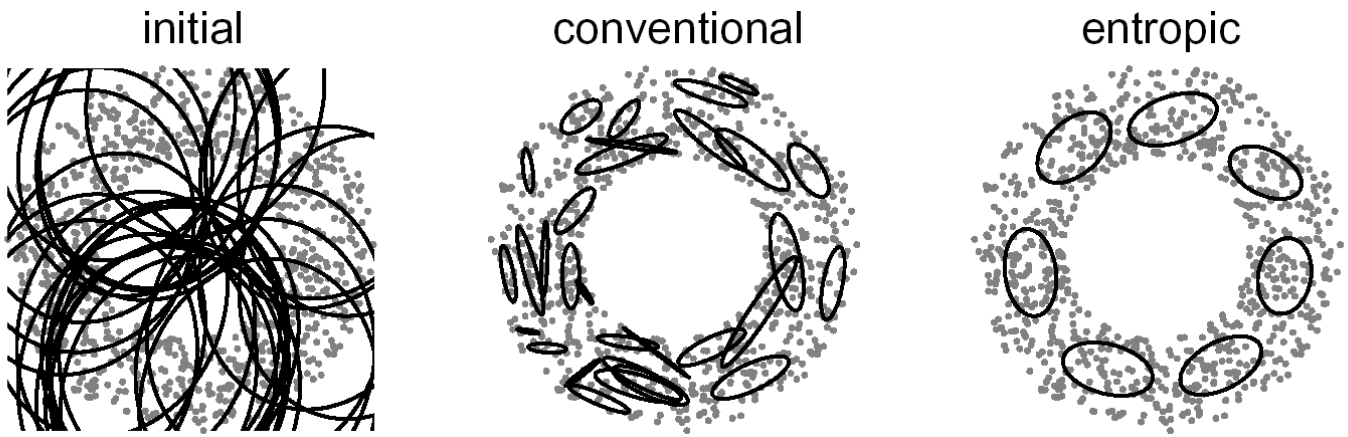


How is entropic training used in practice?

- In the context of mixture modeling, the parameters θ_i are the mixing coefficients of the different Gaussian components, and the "evidence" is the probability of each Gaussian component given the data

$$\omega_i = \sum_{n=1}^{N_{EX}} p\left(c_i | x^{(n)}\right)$$

- The figures below illustrate the results on the classical annulus problem for conventional (EM) training and entropic training. The latter leads to a more concise Gaussian Mixture Model

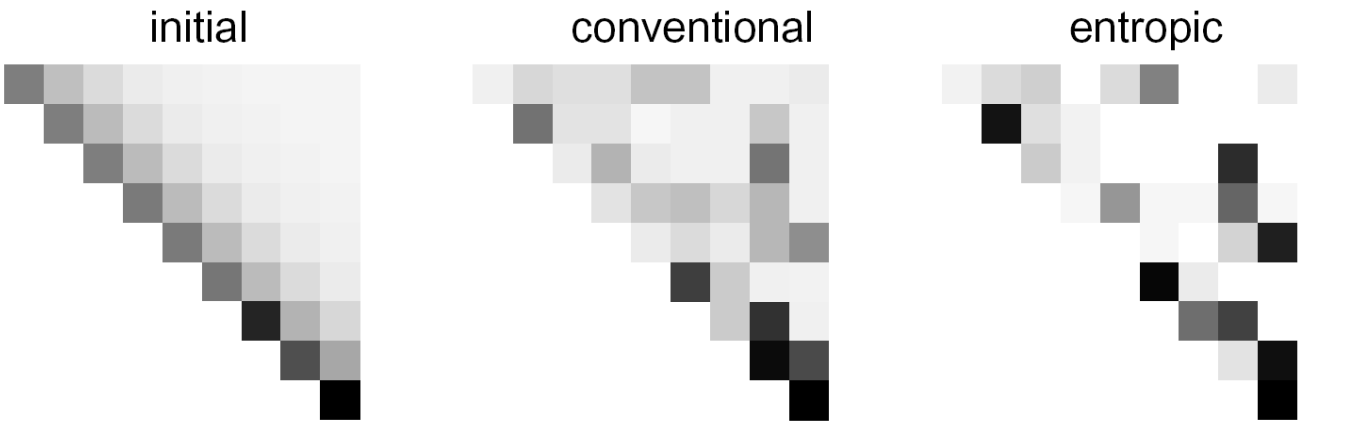


[Brand, 1998]

- In HMM training, each state has a set of parameters θ that represent the transitions probabilities from that state, and the "evidence" is the expected number of state transitions as measured by the E-step in Baum-Welch

$$\omega_i = \sum_{t=1}^{T-1} \xi_t(i,j) = \sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

  - Thus, entropic training treats HMMs as a collection of multinomials, one per state
  - The figures below illustrate the transition matrix for a left-right HMM trained with Baum-Welch and entropic training. The latter leads to a sparse matrix
- In either situation, convergence is accelerated by "trimming" parameters that fall below a threshold (see [Brand, 1998] for details)
  - An added advantage of entropic training is that you can start with a very large HMM (or GMM) and let the algorithm trim the model down to a smaller one



[Brand, 1998]