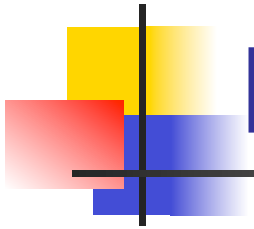
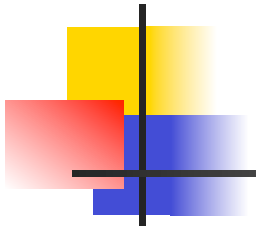


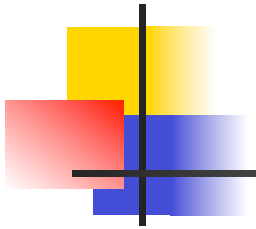
Reasoning with Uncertainty

Markov Models

System Models with Uncertainty in a Probabilistic Framework



- The situation of a system can be modeled using a probability distribution using multiple representations
  - Joint distributions
  - Bayesian Networks
- The behavior of the system over time can be modeled through conditional probabilities for transitions and observations and situation (or state) representations at different points in time



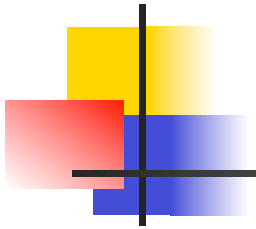
Markov Models

- Behavior in general systems can depend on an infinite history
  - Potentially infinite memory requirements to store system model
    - Infinitely many transition probabilities
    - Infinitely many situations have to be stored
- Many systems are Markovian and have independent, local observations
  - Markov assumption (1-st order Markov):

$$P(s_t \mid s_{t-1}, s_{t-1}, \ldots, s_1) = P(s_t \mid s_{t-1})$$

  - Observations only depend on state:

$$P(o_t \mid s_t, o_{t-1}, s_{t-1}, o_{t-2}, \ldots, s_1) = P(o_t \mid s_t)$$



Markov Models

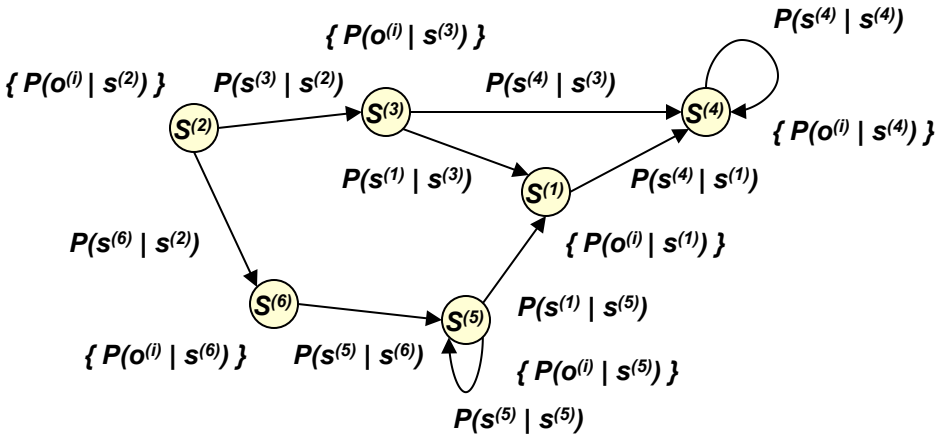
- A Markov model represents the behavior of a system (usually with a finite number of states) that has the Markov property
  - A Markov model for a stationary process contains:

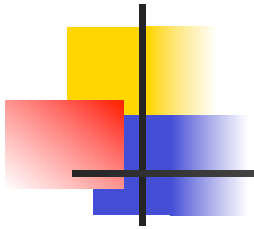
    $<S, O, T, B, \pi>$

    - $S=\{s^{(1)},\ldots,s^{(n)}\}$: State set
    - $O=\{o^{(1)},\ldots o^{(m)}\}$: Observation set
    - T: $P(s^{(i)} \mid s^{(j)})$: Transition probability distribution
    - B: $P(o^{(i)} \mid s^{(j)})$: Observation probability distribution
    - π: $P(s^{(i)})$: Prior state distribution

Hidden Markov Models (HMM)

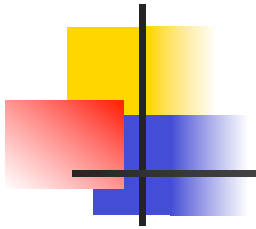
- The underlying system in a finite state HMM can be represented as a Discrete Markov Model with transition and observation probabilities represented by T and B, respectively.





Hidden Markov Models (HMM)

- In a Hidden Markov Model (HMM) the underlying process (i.e. the state) is not observable
  - Observations are linked to the state only through the observation probabilities
  - The behavior of the process is assumed to be stationary
- HMMs are often used to construct models of a process from observations or to determine unobservable properties from the observations
  - Model: $\lambda=(T, B, \pi)$



HMM – Key Problems

- Three problems are of major interest in HMMs:
  - Evaluation Problem: How likely is a particular observation sequence?
  - Estimation Problem: What is the best state sequence to explain the observed data ?
  - Model Construction Problem: What is the best model, λ, to explain the observed data ?

The Evaluation Problem

- The main goal of the evaluation problem is to be able to determine the quality of a given model λ
  - The quality of the model can be measured in terms of its likelihood to explain (generate) the actually observed data.

$$P(o_1, o_2, \ldots, o_T \mid \lambda)$$

  - A better model should have a higher likelihood to generate the observed data, i.e. λ_1 is better that λ_2 if

$$P(o_1, o_2, \ldots, o_T \mid \lambda_1) > P(o_1, o_2, \ldots, o_T \mid \lambda_2)$$

The Evaluation Problem

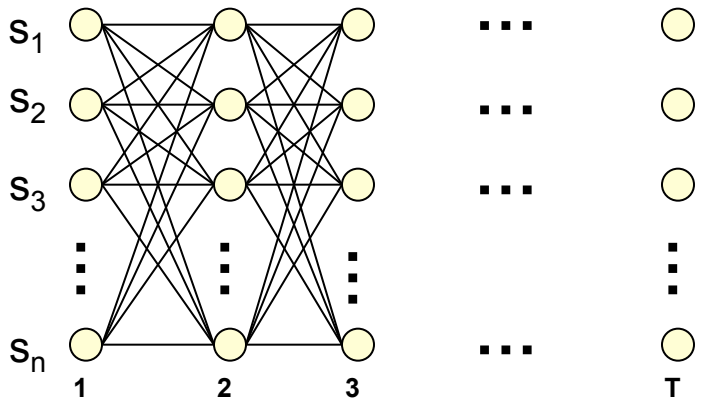
- An intuitive way to solve the evaluation problem is to condition the probability on the state sequence

$$P(o_1, o_2, \ldots, o_T \mid \lambda) = \sum_{\{s_1, s_2, \ldots, s_T\}} P(o_1, o_2, \ldots, o_T \mid s_1, s_2, \ldots, s_T, \lambda) P(s_1, s_2, \ldots, s_T \mid \lambda)$$
$$= \sum_{\{s_1, s_2, \ldots, s_T\}} \left(\prod_{t \in [1..T]} P(o_t \mid s_t, \lambda) \right) \left(\pi(s_1) \prod_{t \in [2..T]} P(s_t \mid s_{t-1}, \lambda) \right)$$

  - This method would have to compute the likelihood of the observation sequence for every possible state sequence and would therefore be $O(2Tn^T)$
  - This makes it intractable even for relatively concise models

The Evaluation Problem

- The main problem with the intuitive solution is that the different paths include a large number of repeated segments
  - Since the system is Markov, the individual transition and observation probabilities in a sequence no longer matter once the probability up to a certain point in the state sequence is computed



The Evaluation Problem – The Forward Algorithm

- It is more efficient to use dynamic programming to solve for the probability
  - $\alpha_i(t)$ is the probability of seeing the first t observations in the sequence and ending up in state $s^{(i)}$

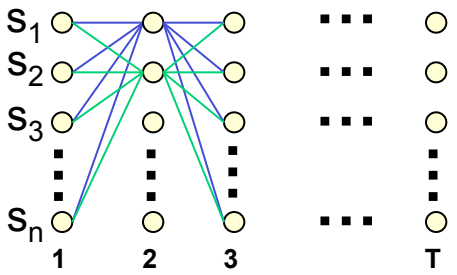
$$\alpha_i(t) = P(o_1, \ldots, o_t, s_t = s^{(i)} \mid \lambda)$$

  - The idea is to solve for $\alpha_i(t)$ for increasing values of t (i.e. to solve for increasing lengths prefixes of the observed sequence and obtain the final result as:

$$P(o_1, o_2, \ldots, o_T \mid \lambda) = \sum_i \alpha_i(T)$$

The Evaluation Problem – The Forward Algorithm

- The Forward algorithm:
  - t=1: $\alpha_i(1) = \pi(s^{(i)})P(o_1 \mid s^{(i)}, \lambda)$
  - t+1:



$$\begin{aligned}
\alpha_i(t+1) &= P(o_1,\ldots,o_t,o_{t+1},s_{t+1}=s^{(i)} \mid \lambda) \\
&= \sum_j P(o_1,\ldots,o_t,o_{t+1},s_t=s^{(j)},s_{t+1}=s^{(i)} \mid \lambda) \\
&= \sum_j P(o_1,\ldots,o_t,s_t=s^{(j)} \mid \lambda)P(o_{t+1},s_{t+1}=s^{(i)} \mid o_1,\ldots,o_t,s_t=s^{(j)}\lambda) \\
&= \sum_j \alpha_j(t)P(o_{t+1},s_{t+1}=s^{(i)} \mid s_t=s^{(j)},\lambda) \\
&= \sum_j \alpha_j(t)P(s^{(i)} \mid s^{(j)},\lambda)P(o_{t+1} \mid s^{(i)},\lambda) = P(o_{t+1} \mid s^{(i)},\lambda)\sum_j \alpha_j(t)P(s^{(i)} \mid s^{(j)},\lambda)
\end{aligned}$$

  - Final: $P(o_1,o_2,\ldots,o_T \mid \lambda) = \sum_i \alpha_i(T)$
  - Complexity: $O(Tn^2)$

The Evaluation Problem – The Backward Algorithm

- The same result can be achieved by starting from the end of the observed sequence and incrementally working towards the entire sequence
  - $\beta_i(t)$ represents the probability that when starting from state $s^{(i)}$ at time t, the remainder of the observation sequence would be observed

$$\beta_i(t) = P(o_{t+1}, \ldots, o_T \mid s_t = s^{(i)}, \lambda)$$

  - Solution can be obtained as:

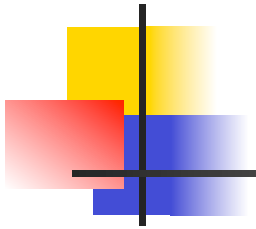
$$P(o_1, o_2, \ldots, o_T \mid \lambda) = \sum_i \pi(s^{(i)}) P(o_1 \mid s^{(i)}) \beta_i(1)$$

The Evaluation Problem – The Backward Algorithm

- The Backward algorithm:
  - t=T: $\beta_i(T) = 1.0$
  - t-1:

$$\begin{aligned}\beta_i(t-1) &= P(o_t, o_{t+1}, \ldots, o_T \mid s_{t-1} = s^{(i)}\lambda) \\ &= \sum_j P(o_t, o_{t+1}, \ldots, o_T, s_t = s^{(j)} \mid s_{t-1} = s^{(i)}, \lambda) \\ &= \sum_j P(o_t, s_t = s^{(j)} \mid s_{t-1} = s^{(i)}, \lambda) P(o_{t+1}, \ldots, o_T \mid o_t, s_t = s^{(j)}, \lambda) \\ &= \sum_j P(s^{(j)} \mid s^{(i)}, \lambda) P(o_t \mid s^{(j)}, \lambda) \beta_j(t)\end{aligned}$$

  - Final: $P(o_1, o_2, \ldots, o_T \mid \lambda) = \sum_i \pi(s^{(i)}) P(o_1 \mid s^{(i)}) \beta_i(1)$
  - Complexity: $O(Tn^2)$



The Prediction Problem

- The main goal of the prediction problem is to predict the actual behavior of the system (i.e. the underlying state sequence) for a given sequence of observations
  - The likelihood of a prediction can be determined in terms of the probability of particular states occurring at different times during the observation.
  - The best prediction is a prediction that maximizes the probability.

The Prediction Problem – First Interpretation

- Interpret the "best" prediction as the sequence of states, $\hat{s}_1, \ldots, \hat{s}_T$, where $\hat{s}_t$ is the most likely state at time t
  - The most likely state at time t:

$$\hat{s}_t = \arg\max_{s^{(j)}} P(s_t = s^{(j)} \mid o_1, \ldots, o_T, \lambda)$$

$$P(s_t = s^{(i)} \mid o_1, \ldots, o_T, \lambda) = \frac{P(o_1, \ldots, o_t, s_t = s^{(i)}, o_{t+1}, \ldots, o_T \mid \lambda)}{P(o_1, \ldots, o_T \mid \lambda)}$$

$$= \frac{P(o_1, \ldots, o_t, s_t = s^{(i)} \mid \lambda) P(o_{t+1}, \ldots, o_T \mid o_1, \ldots, o_t, s_t = s^{(i)}, \lambda)}{P(o_1, \ldots, o_T \mid \lambda)}$$

$$= \frac{P(o_1, \ldots, o_t, s_t = s^{(i)} \mid \lambda) P(o_{t+1}, \ldots, o_T \mid s_t = s^{(i)}, \lambda)}{P(o_1, \ldots, o_T \mid \lambda)} = \frac{\alpha_i(t)\beta_i(t)}{P(o_1, \ldots, o_T \mid \lambda)}$$

  - Problem: The sequence $\hat{s}_1, \ldots, \hat{s}_T$ might have very low probability (or be impossible) because $P(\hat{s}_{t+1} \mid \hat{s}_t, \lambda)$ could be very small (or 0) for some t

The Prediction Problem – Second Interpretation

- To address transition probabilities, the "best" prediction can be interpreted as the state sequence, $\hat{s}_1, \ldots, \hat{s}_T$, that has the highest likelihood to produce the observation sequence.

$$\hat{s}_1, \ldots, \hat{s}_T = \arg\max_{(s^{(j1)}, \ldots, s^{(jT)})} P(s_1 = s^{(j_1)}, \ldots, s_T = s^{(j_T)} \mid o_1, \ldots, o_T, \lambda)$$

  - The most direct way to compute this would be to compute the probability for each sequence and then pick the maximum but this would be prohibitively expensive.
    - Complexity: $O(n^T * Tn^2)$
  - Again, many transitions occur in a large number of sequences and should not be recomputed

The Prediction Problem - The Viterbi Algorithm

- The Markov property permits again to compute the solution iteratively.
  - $\delta_i(t)$ represents the probability of the most likely state sequence ending in state $s^{(i)}$ corresponding to the first t observations.

$$\delta_i(t) = \max_{(s_1,\ldots,s_{t-1})} P(s_1,\ldots,s_{t-1},s_t = s^{(i)},o_1,\ldots,o_t \mid \lambda)$$

  - To address the problem of the addition of a symbol changing the state sequence due to a low transition probability, $\Psi_i(t)$, represents the predecessor of the end state $s^{(i)}$ along the most likely state sequence of length t

$$\Psi_i(t) = \arg\max_j P(s_1,\ldots,s_{t-1} = s^{(j)},s_t = s^{(i)},o_1,\ldots,o_t \mid \lambda)$$

The Prediction Problem - The Viterbi Algorithm

- The Viterbi algorithm:
  - t=1: $\delta_i(1) = \pi(s^{(i)})P(o_1 \mid s^{(i)}, \lambda)$

    $\Psi_i(1) = 0$
  - t+1:

$$\delta_i(t+1) = \max_{(s_1,\ldots,s_t)} P(s_1,\ldots,s_t, s_{t+1} = s^{(i)}, o_1,\ldots,o_{t+1} \mid \lambda)$$

$$= \max_{(s_1,\ldots,s_t)} P(s_1,\ldots,s_t, o_1,\ldots,o_t \mid \lambda) P(s_{t+1} = s^{(i)}, o_{t+1} \mid s_1,\ldots,s_t, o_1,\ldots,o_t, \lambda)$$

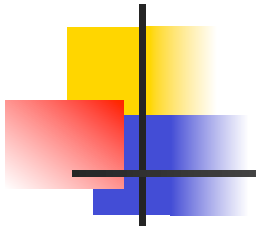
$$= \max_j \delta_j(t) P(s^{(i)} \mid s^{(j)}, \lambda) P(o_{t+1} \mid s^{(i)}, \lambda)$$

$$\Psi_i(t+1) = \arg\max_j \delta_j(t) P(s^{(i)} \mid s^{(j)}, \lambda) P(o_{t+1} \mid s^{(i)} \lambda)$$

  - Final: $\max_{(s_1,\ldots,s_T)} P(s_1,\ldots,s_T \mid o_1,\ldots,o_T, \lambda) = \dfrac{\max_i \delta_i(T)}{P(o_1,\ldots,o_T \mid \lambda)}$

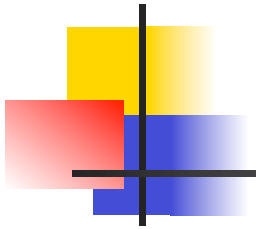
$$\hat{s}_t = s^{(i_t)} : \; i_T = \arg\max_i \delta_i(T) \;, \; i_{t-1} = \Psi_{i_t}(t)$$

  - Complexity: $O(Tn^2)$



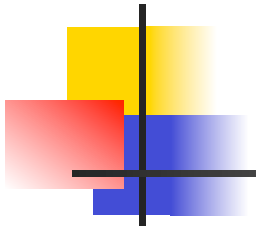
The Model Construction Problem

- The main goal of the model construction problem is to find the model that is most likely to have generated the observed data.
  - Determine T, B, and π that maximize the evaluation probability for a model with n states.
  - Note: Increasing the number of states will generally increase the likelihood but makes the model prone to overfitting (i.e. the model will not only try to explain the system but also the noise)
    - There is always a model with T states (one state per observation) that perfectly explains the data but is generally not a good description of the actual system.



The Model Construction Problem

- There is no analytic solution for the model construction problem but multiple iterative approaches exist.
  - Maximum Likelihood approaches
    - Baum-Welch algorithm
    - Gradient-based algorithms
  - Maximum Mutual Information approaches
    - Gradient-based algorithms



Maximum Likelihood Approaches

- Maximum likelihood approaches try to compute the model that has the highest probability to generate the data

$$\hat{\lambda} = \arg\max_{\lambda} P(o_1,\ldots,o_T \mid \lambda)$$

  - Search over the model parameters is intractable
    - There is an infinite number of n-state models since the model parameters are continuous
  - Analytic solution of this optimization problem is not possible

Maximum Likelihood Approaches

- Through marginalization of the maximum likelihood formulation we get

$$\hat{\lambda} = \text{argmax}_\lambda P(o_1,\ldots,o_T \mid \lambda) = \text{argmax}_\lambda \sum_{s_1,\ldots,s_T} P(o_1,\ldots,o_T,s_1,\ldots,s_T \mid \lambda)$$

$$= \text{argmax}_\lambda \sum_{s_1,\ldots,s_T} P(o_1,\ldots,o_T \mid s_1,\ldots,s_T,\lambda) P(s_1,\ldots,s_T \mid \lambda)$$

$$= \text{argmax}_\lambda \sum_{s_1,\ldots,s_T} \pi(s_1) P(o_1 \mid s_1) \prod_{t=2}^{T} P(s_t \mid s_{t-1}) P(o_t \mid s_t)$$

  - Gradient methods can be used to solve the problem by computing gradient using the Markov property to avoid the exponential sum
    - Same principle as for Forward, Backward and Viterbi
    - But: Gradient is highly complex

Expectation Maximization

- Expectation maximization gives a different way to solve marginal likelihood optimization
  - Convert the marginal likelihood to log likelihood

$$\hat{\lambda} = \text{argmax}_{\lambda} \sum_{s_1,\ldots,s_T} P(o_1,\ldots,o_T,s_1,\ldots,s_T \mid \lambda)$$

$$= \text{argmax}_{\lambda} \log \sum_{s_1,\ldots,s_T} P(o_1,\ldots,o_T,s_1,\ldots,s_T \mid \lambda)$$

    - Observed variables: sequence of o_t
    - Observed variables: sequence of s_t
    - Parameters: λ

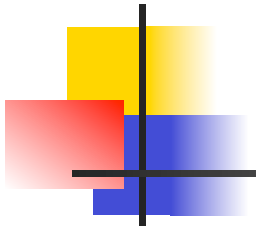
Expectation Maximization

- Expectation maximization solves this problem by alternating between two steps
  - Expectation step: determine expected values for hidden variables using the current parameters
    - Due to Markov property this does not require estimating all possible sequences but only

      $P(s_t)$, $P(s^{(i)} \mid s^{(j)})$
  - Maximization step: Find best parameters assuming expectations for the hidden variables

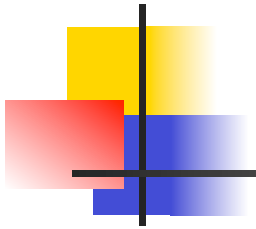
$$\tilde{\lambda}_{t+1} = \operatorname{argmax}_{\lambda} E\left[\log P(o_1,\ldots,o_T,s_1,\ldots,s_T \mid \lambda)\right]$$

$$= \arg\max_{\lambda} \sum_{s_1,\ldots,s_T} P(s_1,\ldots,s_T \mid o_1,\ldots,o_T,\lambda_t) \log P(o_1,\ldots,o_T,s_1,\ldots,s_T \mid \lambda)$$



Expectation Maximization

- Expectation maximization is a general algorithm for marginal likelihood maximization
  - The Expectation and the Maximization steps are solves this problem by alternating between two steps are generally much easier to solve than the gradient ascent problem for marginal likelihood
- The application of EM to the model learning problem in HMMs is the Baum-Welch algorithm.



The Baum-Welch Algorithm

- The Baum-Welch algorithm is an instance of an Expectation Maximization (EM) algorithm to determine the maximum likelihood model
  - Starting from an initial model $\lambda_0=(T_0, B_0, \pi_0)$ and an observation sequence, the algorithm iterates the following:
    - Expectation step: calculate the forward and backward probabilities for the current model and the actual observation sequence.
    - Maximization step: use the forward and backward probabilities from the expectation step and the actual observation sequence to determine the optimal estimates for the model parameters.
  - The model derived in the maximization step is always at least as good as the previous model since it is based more closely on the actually observed data.

The Model Construction Problem The Baum-Welch Algorithm

- Expectation Step:
  - Calculate the probabilities that, given the observation sequence, the system was in state i at time t, $\gamma_i(t)$, and the probability that it performed a transition from state i to state j at time t, $\xi_{i,j}(t)$.

$$\gamma_i(i) = P(s_t = s^{(i)} \mid o_1, \ldots, o_T, \lambda) = \frac{\alpha_i(t)\beta_i(t)}{P(o_1, \ldots, o_T \mid \lambda)}$$

$$\xi_{i,j}(t) = P(s_t = s^{(i)}, s_{t+1} = s^{(j)} \mid o_1, \ldots, o_T, \lambda)$$

$$= \frac{\alpha_i(t)P(s_{t+1} = s^{(j)} \mid s_t = s^{(i)}, \lambda)P(o_{t+1} \mid s_{t+1} = s^{(j)}, \lambda)\beta_j(t+1)}{P(o_1, \ldots, o_T \mid \lambda)}$$

$$= \frac{\alpha_i(t)P(s^{(j)} \mid s^{(i)}, \lambda)P(o_{t+1} \mid s^{(j)}, \lambda)\beta_j(t+1)}{P(o_1, \ldots, o_T \mid \lambda)}$$

The Model Construction Problem The Baum-Welch Algorithm

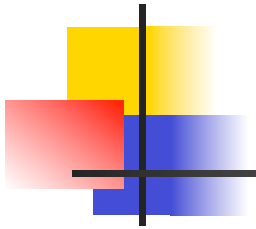
- Maximization Step:
  - Assuming that the probabilities of states and transitions in the expectation step are accurate, calculate the optimal values for the model parameters.

$$T : P(s^{(i)} \mid s^{(j)}) = \frac{\#\,transitions\ from\ s^{(j)}\ to\ s^{(i)}}{\#\,transitions\ out\ of\ s^{(j)}} = \frac{\sum_{t=1}^{T-1} \xi_{j,i}(t)}{\sum_{t=1}^{T-1} \gamma_j(t)}$$

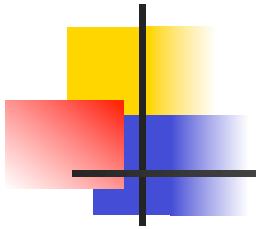
$$B : P(o^{(i)} \mid s^{(j)}) = \frac{\#\,in\ state\ s^{(j)}\ when\ o^{(i)}\ was\ observed}{\#\,in\ state\ s^{(j)}} = \frac{\sum_{t=1, o_t = o^{(i)}}^{T} \gamma_j(t)}{\sum_{t=1}^{T} \gamma_j(t)}$$

$$\pi : \pi(s^{(i)}) = \gamma_i(1)$$

The Model Construction Problem The Baum-Welch Algorithm



- The Baum-Welch algorithm iterates until the model no longer changes (or improvement drops below a specific threshold)
  - Algorithm is guaranteed to converge to a local optimum (no guarantee for global optimum)
    - Initial model selection is important for:
      - Convergence speed
      - Quality of final model found (by determining which local optimum will be found)
- To learn values for prior $\pi(s)$, observations have to contain multiple sequences



Markov Models

- Markov models represent a powerful mechanism to model stochastic processes that have finite state and observation sets.
- Hidden Markov Models (HMM) provide techniques to automatically
  - Evaluate a model
  - Predict the underlying state sequence from observations
  - Learn a model from observed data
- Markov models are more difficult to use if the state space is not finite.