# Design and Implementation of Speech Recognition Systems

*Fall 2014*
*Ming Li*

Class 12: Training with continuous speech
Nov 6th

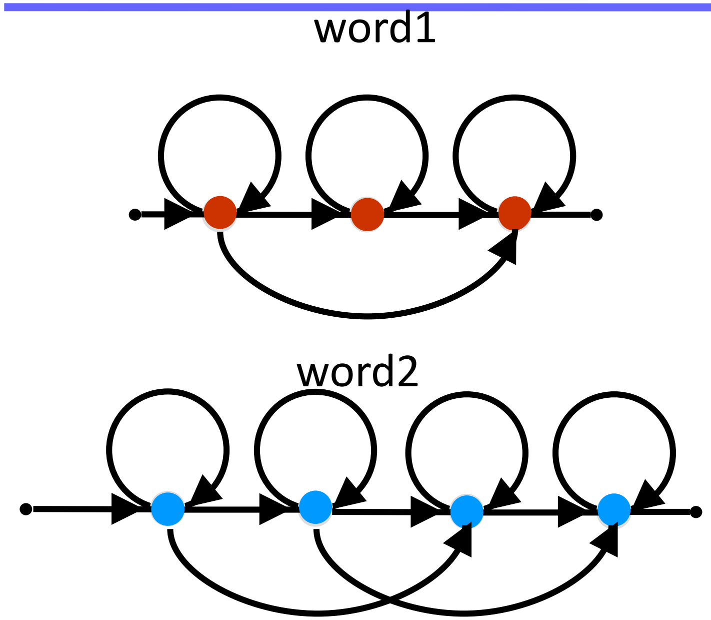
*Thanks to Professor Bhiksha Raj for the contribution of the slides*

# Training from Continuous Recordings

- Thus far we have considered training from isolated word recordings
  - Problems:  Very hard to collect isolated word recordings for all words in the vocabulary
  - Problem: Isolated word recordings have pauses in the beginning and end
    - The corresponding models too expect pauses around each word
    - People do not pause between words in continuous speech

- How to train from continuous recordings
  - I.e.,  how to train models for “0”, “1”, “2”, .. Etc. from recordings of the kind:  0123, 2310, 1121..

# Training HMMs for multiple words



Example: train HMMs for both words from recordings such as
“word1”
“word1”
..
“word2”
“word2”
..
Or from many recordings such as
“word1 word2 word2 word1 word1”

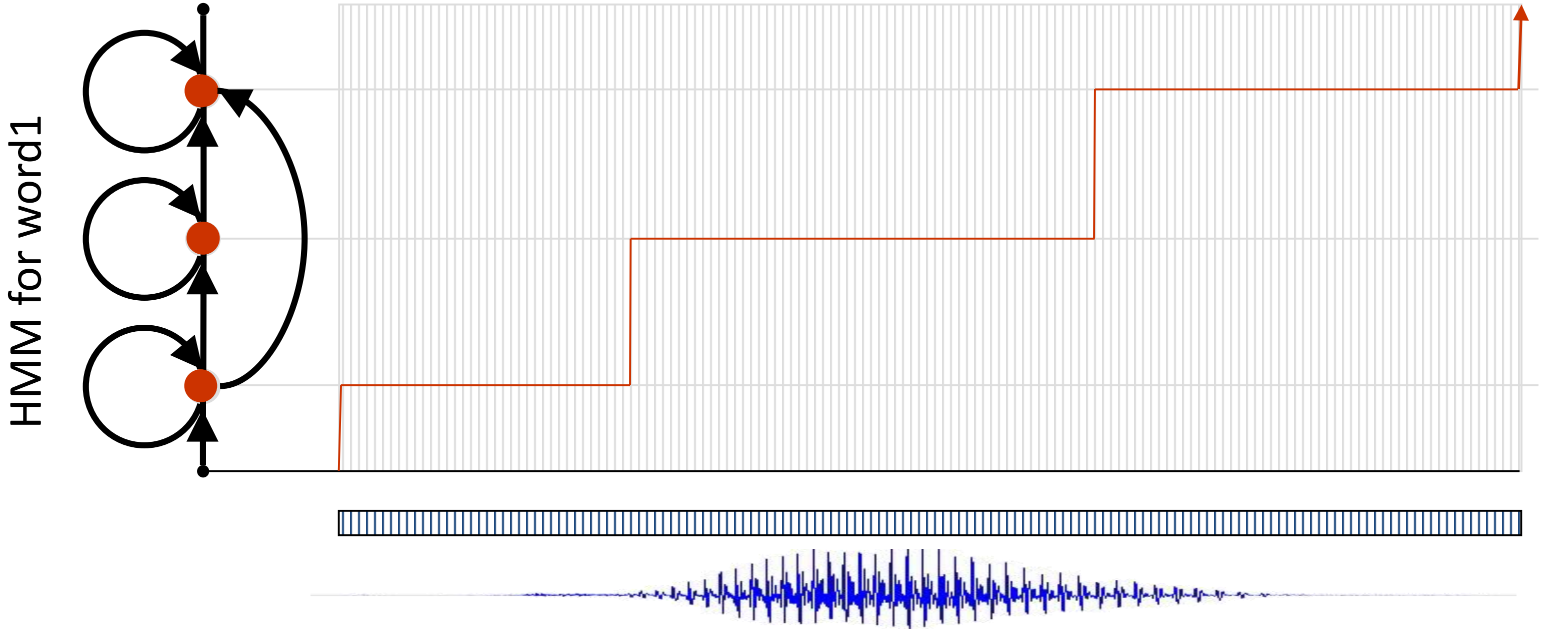
- Two ways of training HMMs for words in a vocabulary

- Isolated word recordings
  – Record many instances of each word (separately) to train the HMM for the word
  – HMMs for each word trained separately

Connected word recordings

– Record connected sequences of words
– HMMs for all words trained jointly from these “connected word” recordings

# Training word HMMs from isolated instances



- Assign a structure to the  HMM for word1
- Initialize HMM parameters
  - Initialize state output distributions as Gaussian

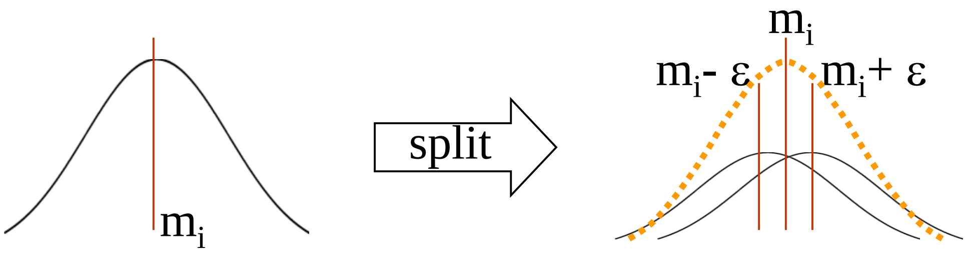
- Segmental K-means training : Iterate the following until convergence
  - Align the sequence of feature vectors derived from each recording of the word to the HMM (segment the recordings into states)
  - Reestimate HMM parameters from segmented training instances

# Segmental K-means

- All HMMs are initialized with Gaussian state output distributions
- The segmental K-means procedure results in HMMs with Gaussian state output distributions

- After the HMMs with Gaussian state output distributions have converged, *split* the Gaussians to generate Gaussian mixture state output distributions
  - Gaussian output distribution for a state i :: $P_i(x)$ = Gaussian(x, $m_i$, $C_i$)
  - New distribution
    $P_i(x)$ = 0.5Gaussian(x, $m_i$+e, $C_i$) + 0.5Gaussian(x, $m_i$-e, $C_i$)
  - e is a very small vector (typically 0.01 * $m_i$)

- Repeat Segmental K-means with updated models



- Repeat Segmental K-means procedure with modified HMMs until convergence

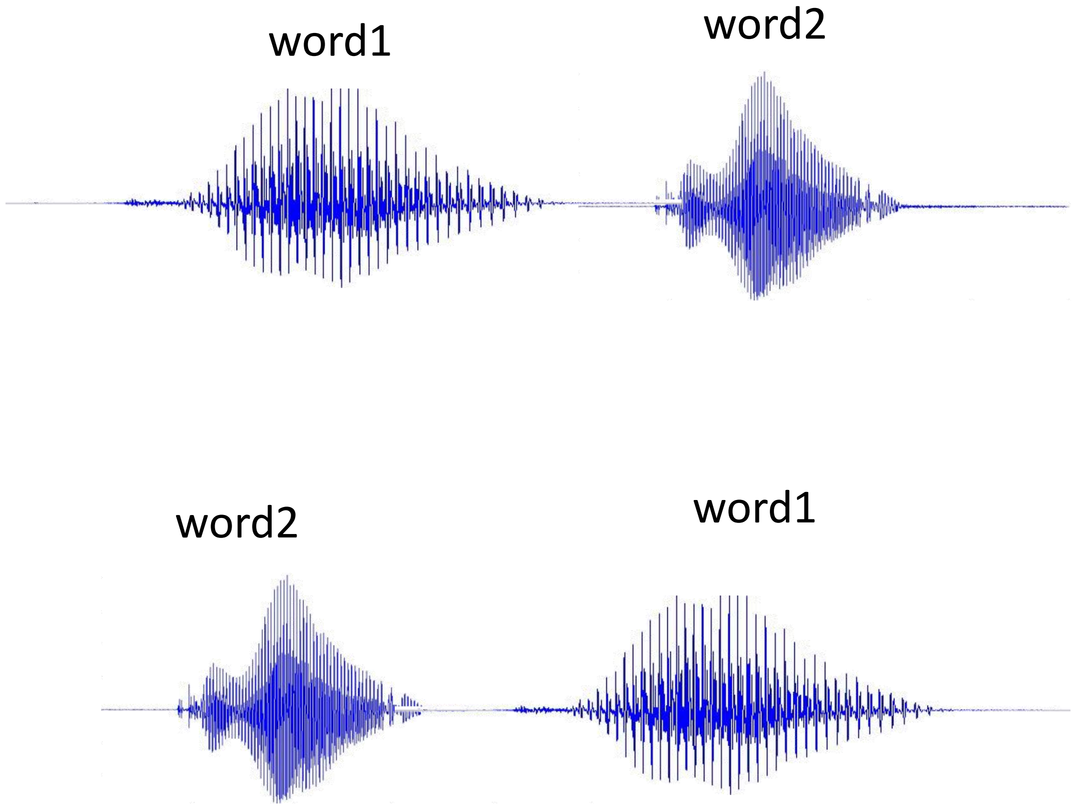
# Training HMM for words from isolated instances

- HMM training for all words in the vocabulary follows an identical procedure

- The training of any word does not affect the training of any other word
  - Although eventually we will be using all the word models together for recognition

- Once the HMMs for each of the words have been trained, we use them for recognition
  - Either recognition of isolated word recordings or recognition of connected words

# Training word models from connected words (continuous speech)

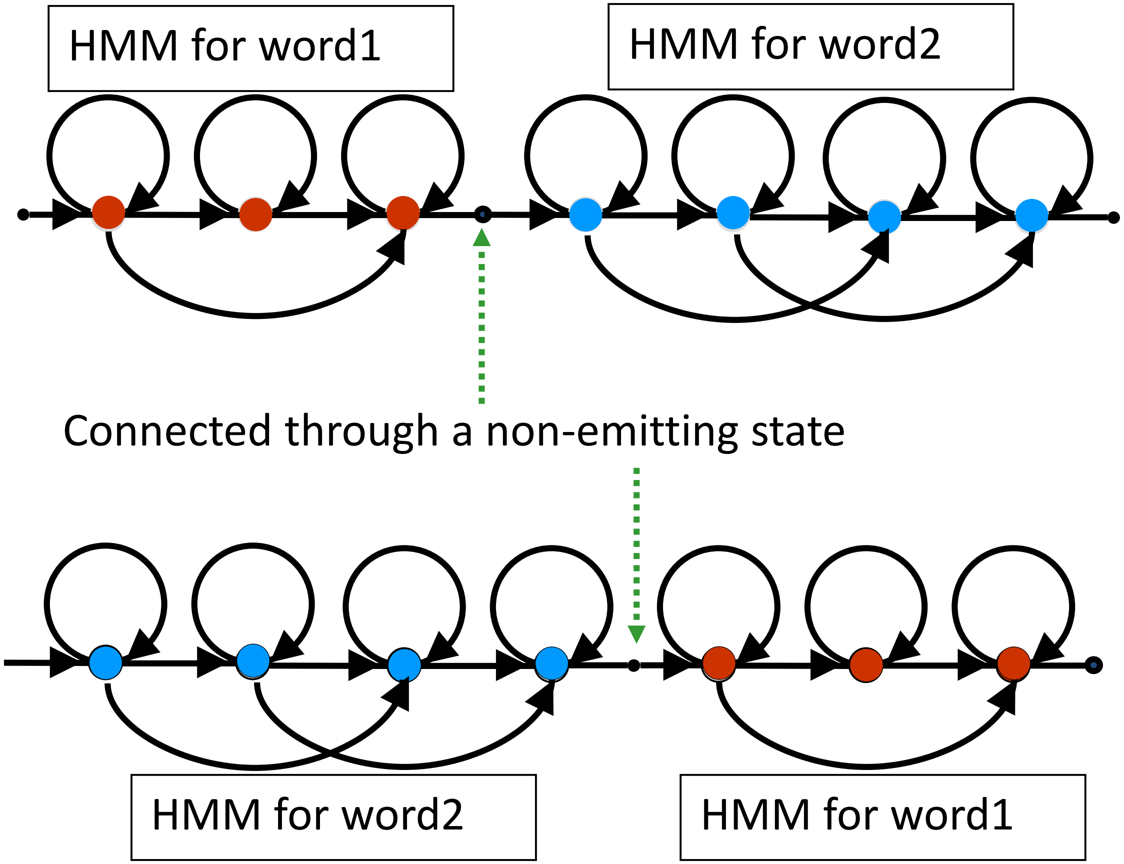


When training with connected word recordings, different training utterances may contain different sets of words, in any order

The goal is to train a separate HMM for each word from this collection of varied training utterances

# Training word models from connected words (continuous speech)

HMM for the word sequence “word1 word2”



HMM for word1

HMM for word2

Connected through a non-emitting state

HMM for word2

HMM for word1

HMM for the word sequence “word2 word1”

Each word sequence has a different HMM, constructed by connecting the HMMs for the appropriate words

# Training word models from connected words (continuous speech)

- Training from continuous recordings follows the same basic procedure as training from isolated recordings

1. Assign HMM topology for the HMMs of all words

2. Initialize HMM parameters for all words
   - Initialize all state output distributions as Gaussian

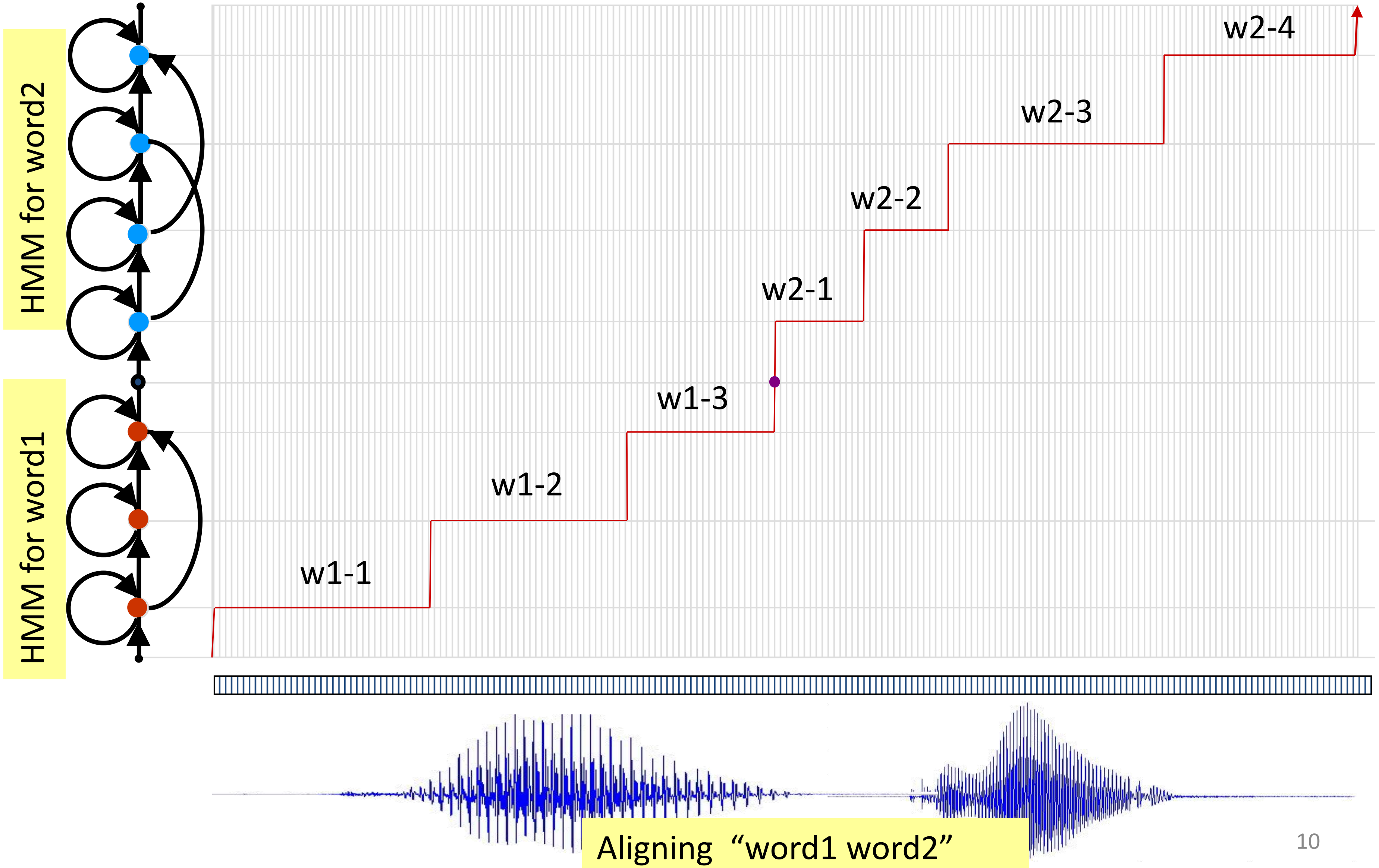
3. Segment all training utterances

4. Reestimate HMM parameters from segmentations

5. Iterate until convergence

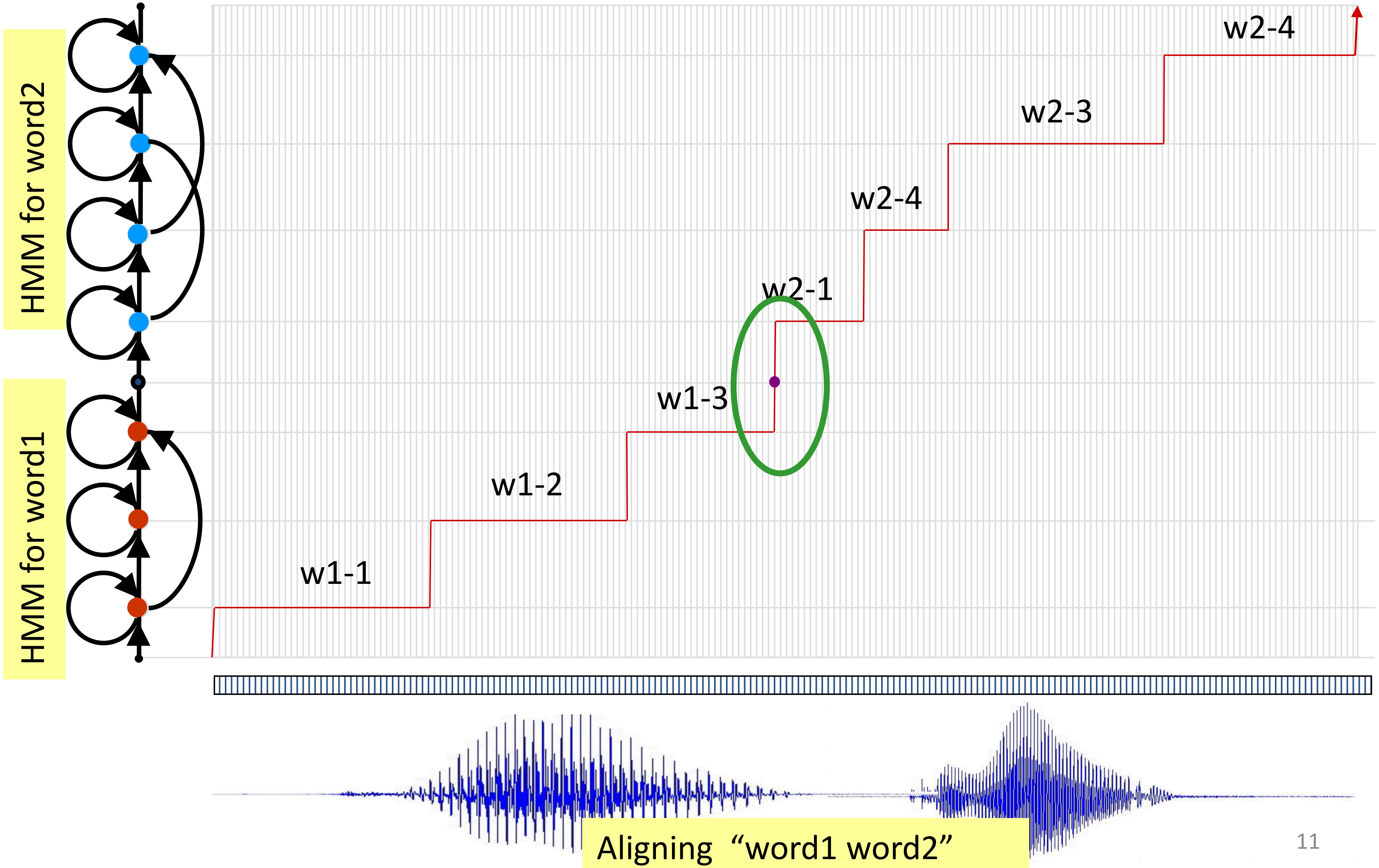
6. If necessary, increase the number of Gaussians in the state output distributions and return to step 3

# Segmenting connected recordings into states



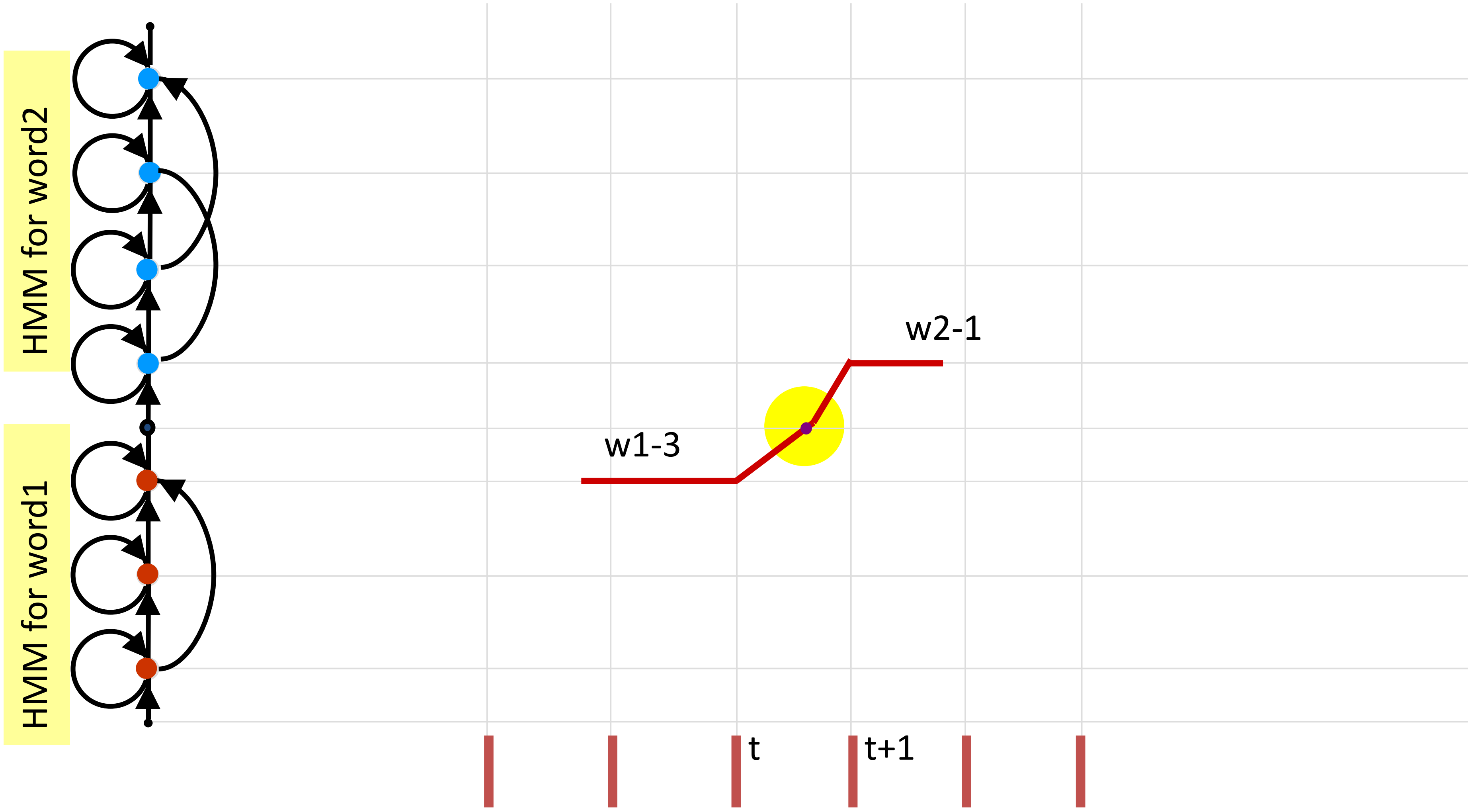
Aligning “word1 word2”

# Segmenting connected recordings into states



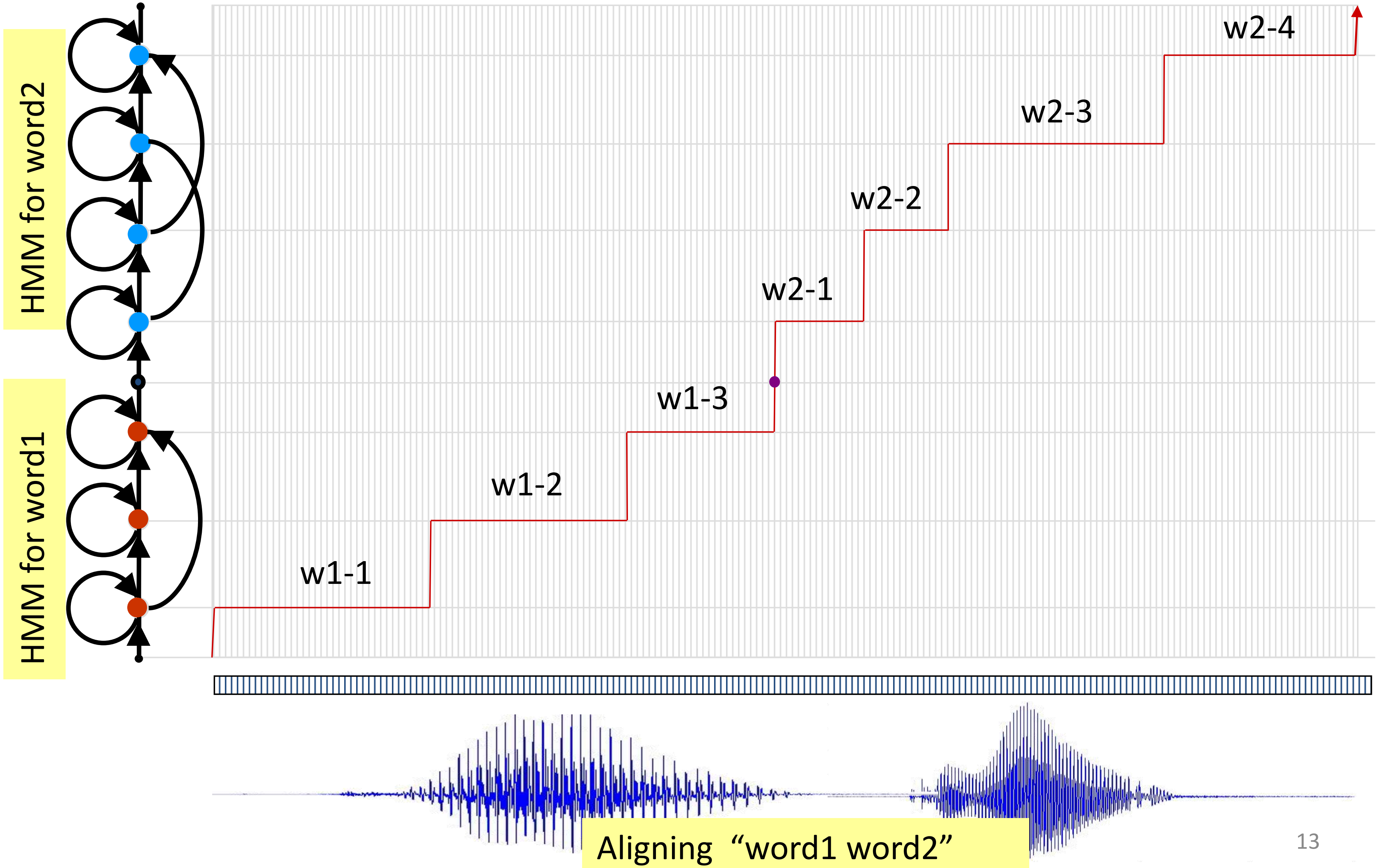
Aligning “word1 word2”

# Segmenting connected recordings into states



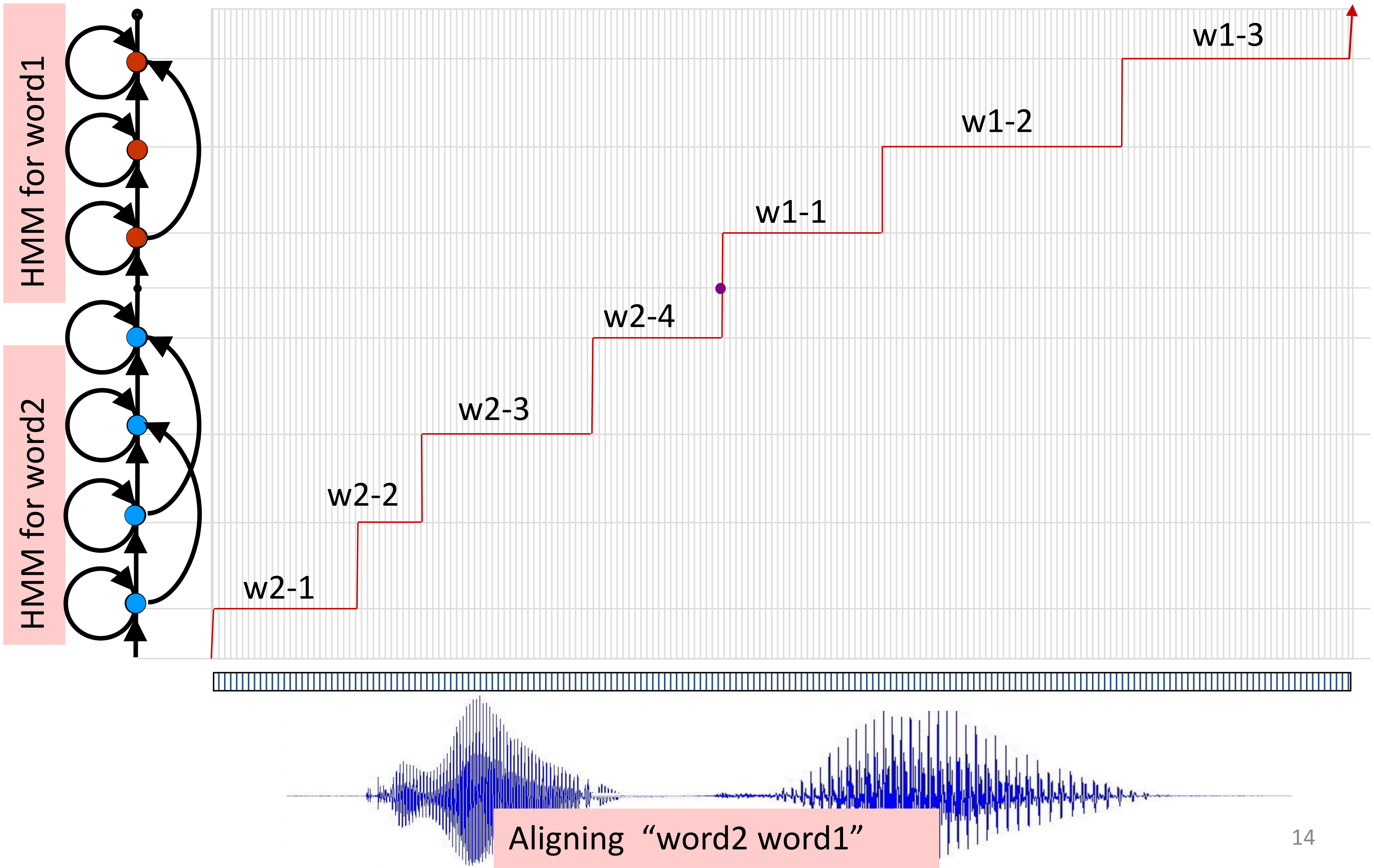
Aligning “word1 word2”

# Segmenting connected recordings into states



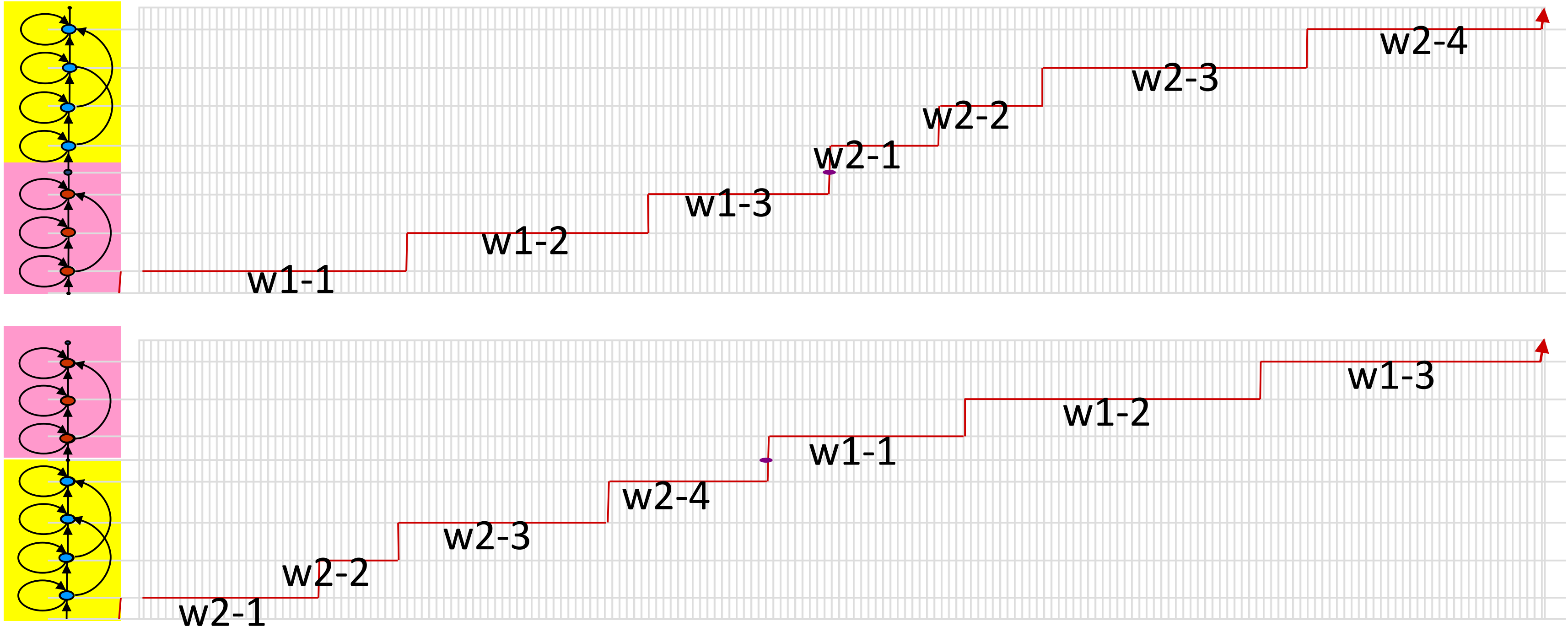
Aligning "word1 word2"

# Segmenting connected recordings into states

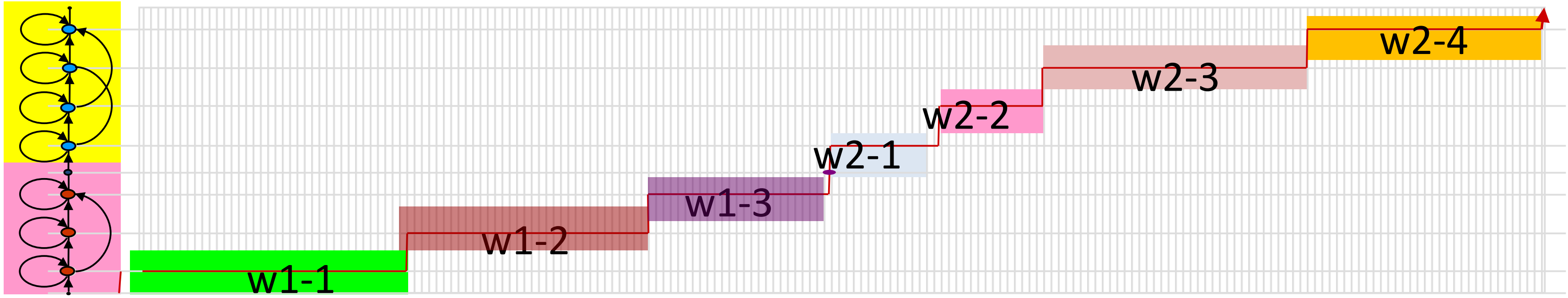


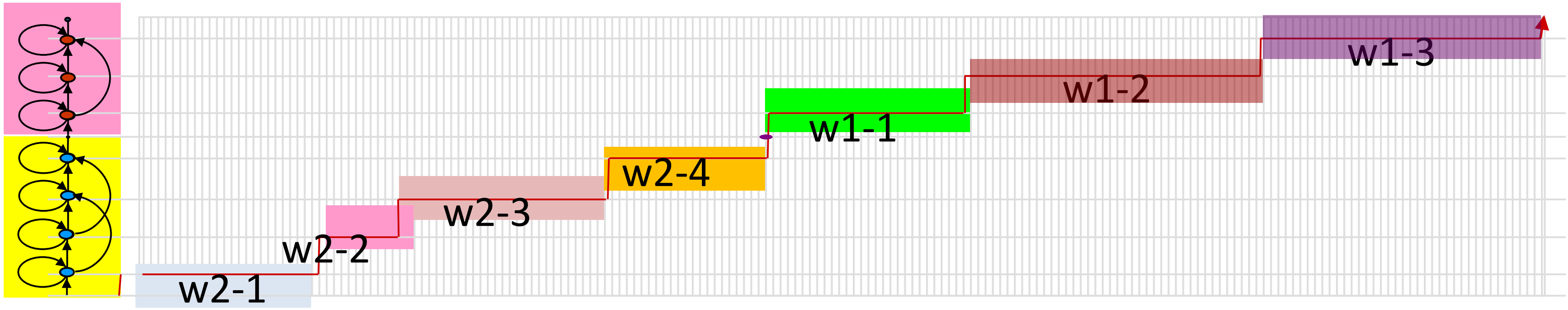
Aligning “word2 word1”

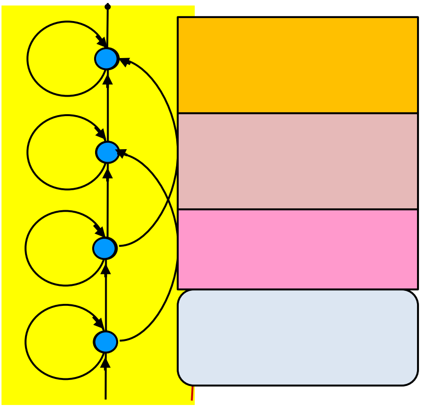
# Segmenting connected recordings into states

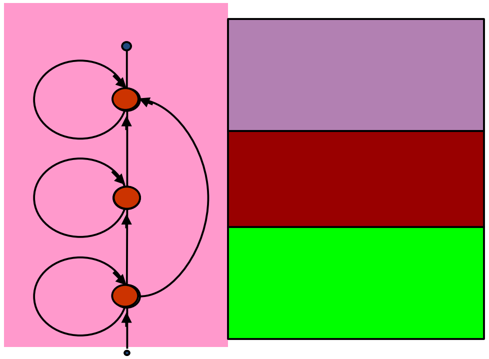


# Aggregating Vectors for Parameter Update





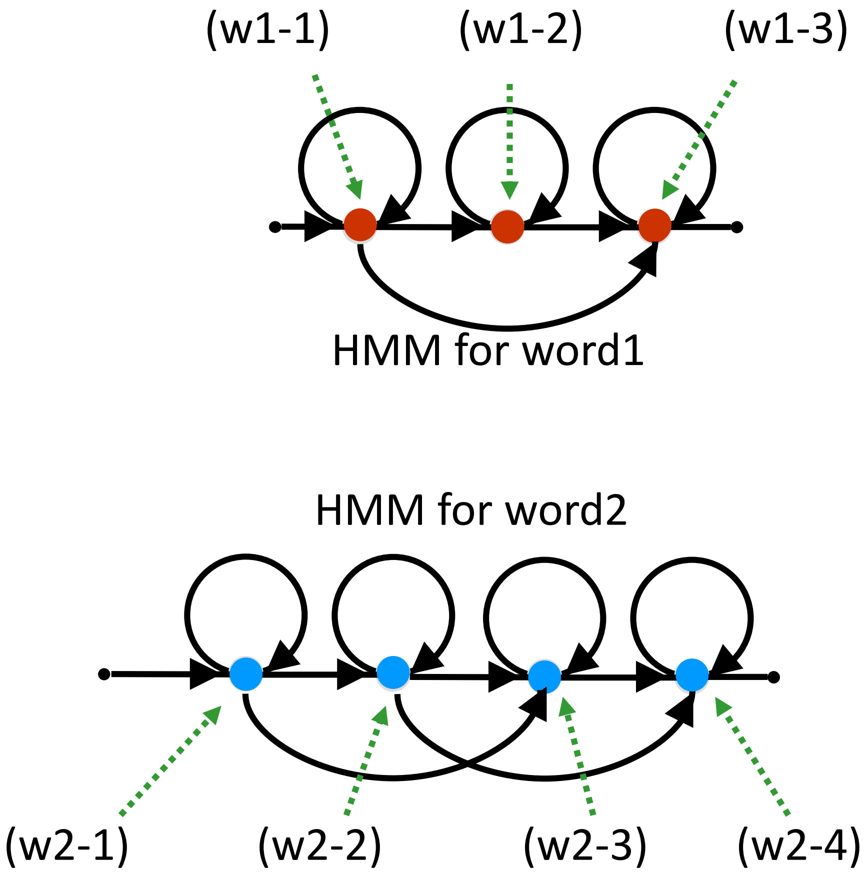




Each state has a "bin" of a particular color

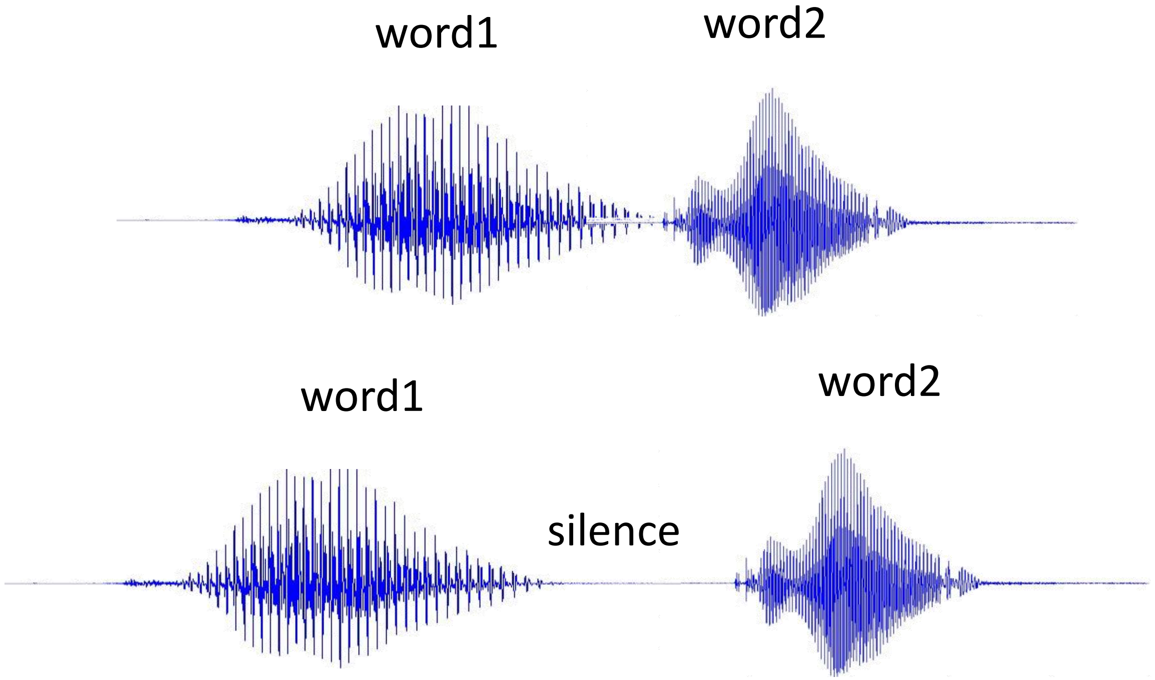
All data segments corresponding to the region of the best path marked by a color are aggregated into the "bin" of the same color

# Training HMMs for multiple words from segmentations of multiple utterances



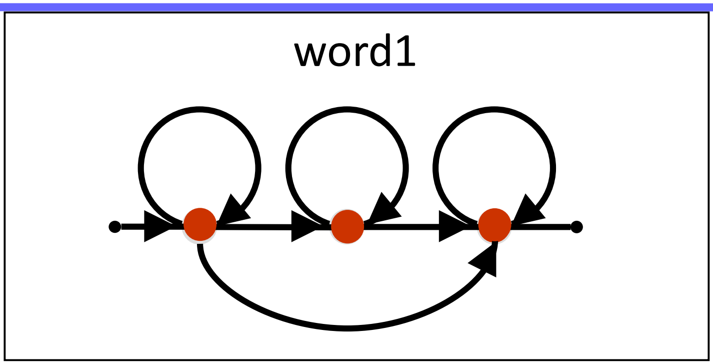
The HMM parameters (including the parameters of state output distributions and transition probabilities) for any state are learned from the collection of segments associated with that state

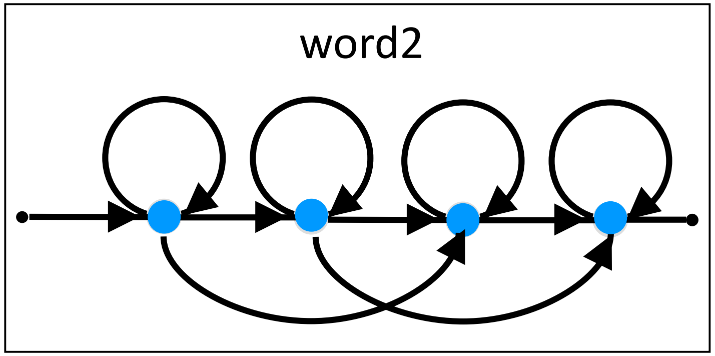
# Training word models from connected words (continuous speech)

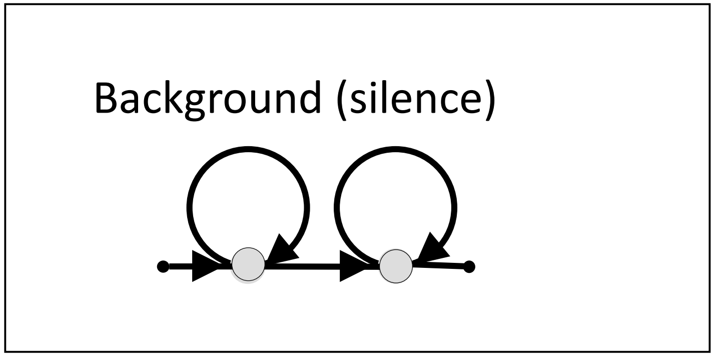


- Words in a continuous recording of connected words may be spoken with or without intervening pauses
- The two types of recordings are different
- A HMM created by simply concatenating the HMMs for two words would be inappropriate if there is a pause between the two words
    - No states in the HMM for either word would represent the pause adequately

# Training HMMs for multiple words

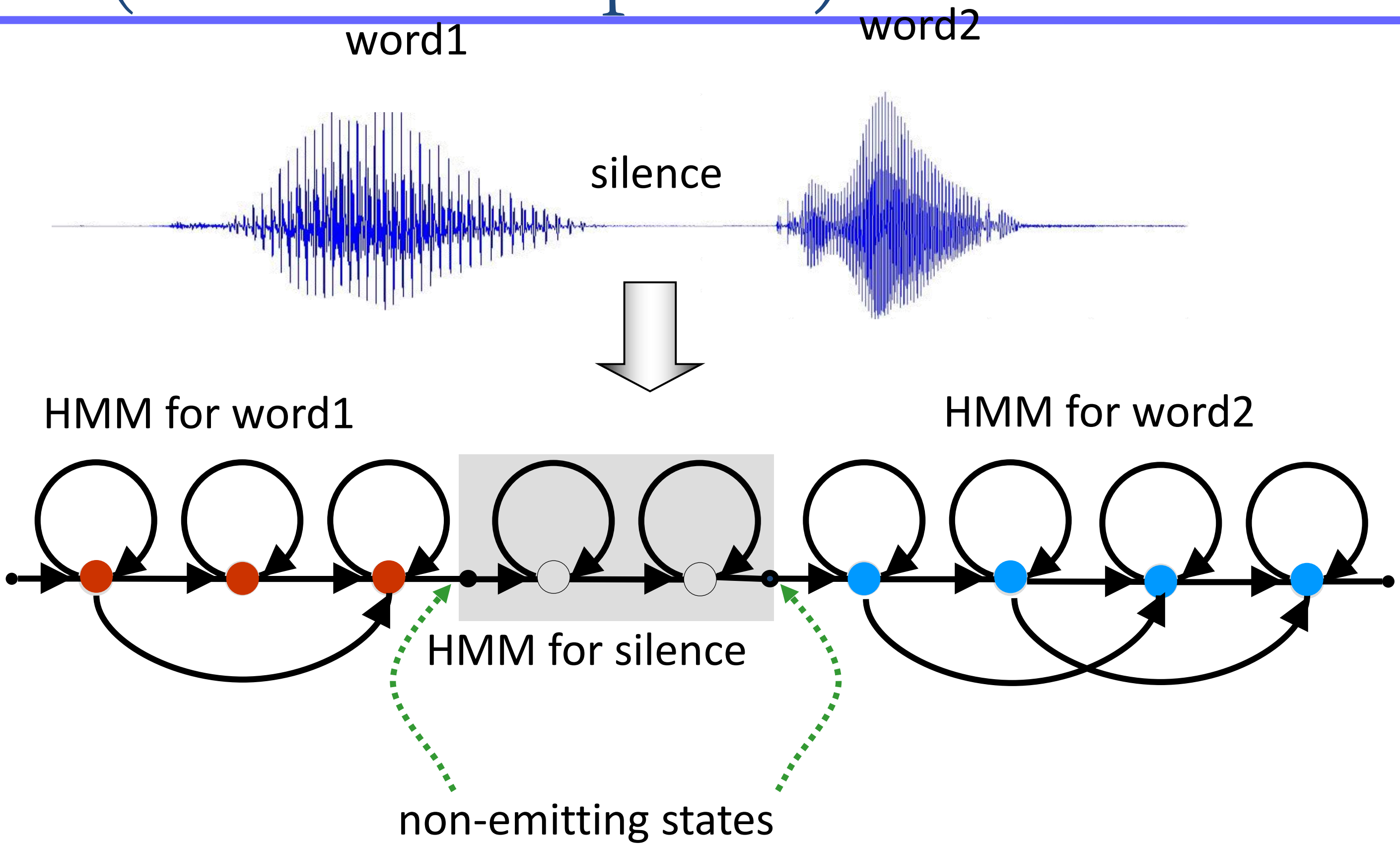






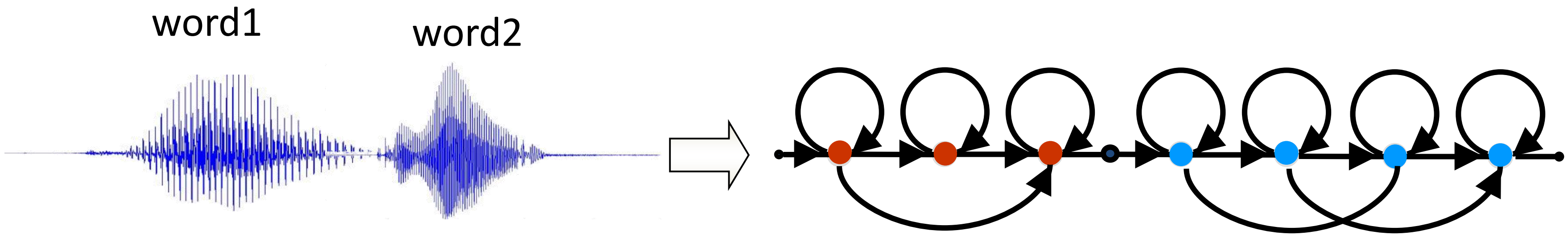
- To account for pauses between words, we must  model pauses
- The signal recorded in pauses is mainly silence
  - In reality it is the background noise for the recording environment
- These pauses are modeled by an HMM of their own
  - Usually called the *silence* HMM, although the term *background sound* HMM may be more appropriate
- The parameters of this HMM are learned along with other HMMs

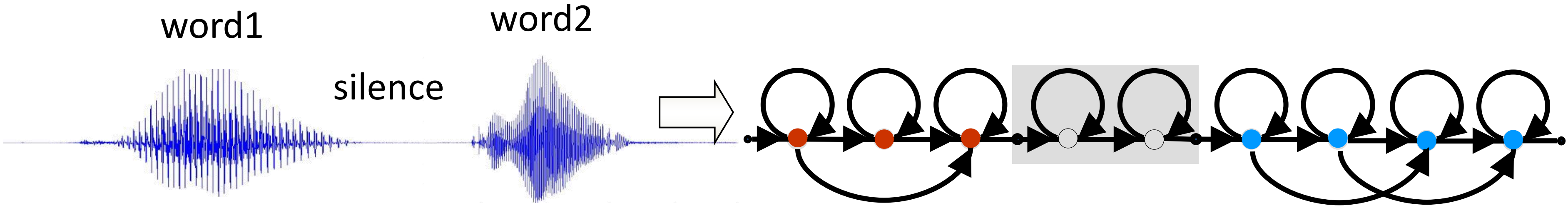
# Training word models from connected words (continuous speech)



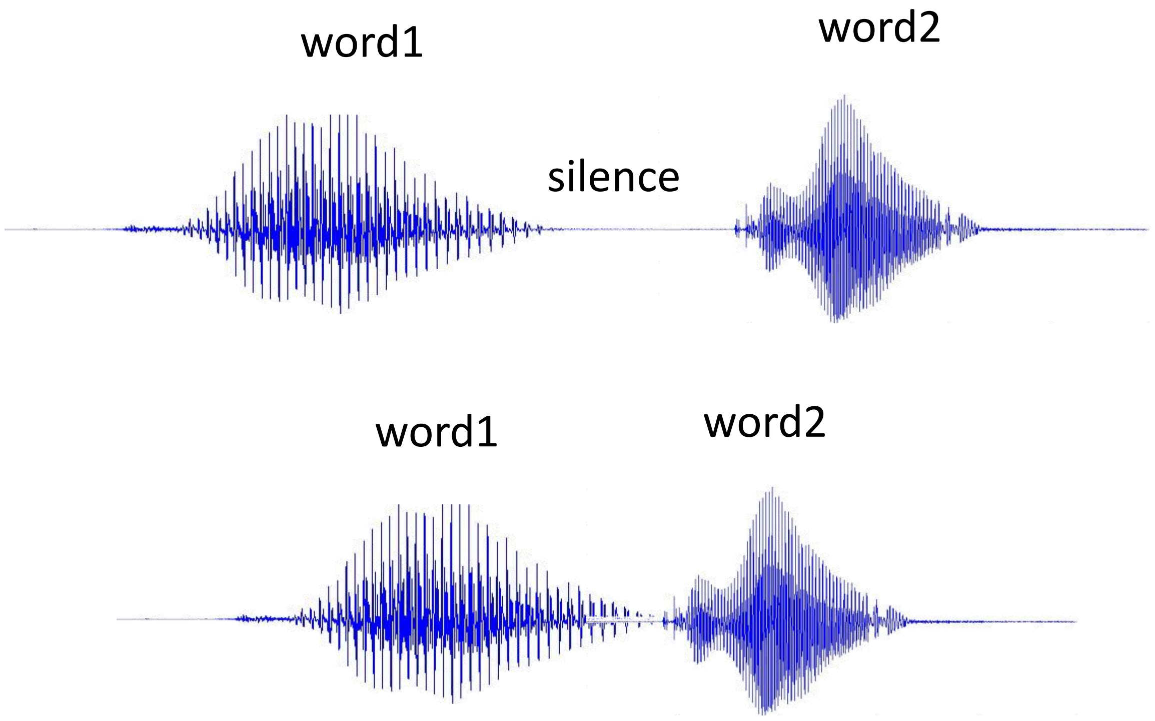
If it is known that there is a pause between two words, the HMM for the utterance must include the silence HMM between the HMMs for the two words

# Training word models from connected words (continuous speech)



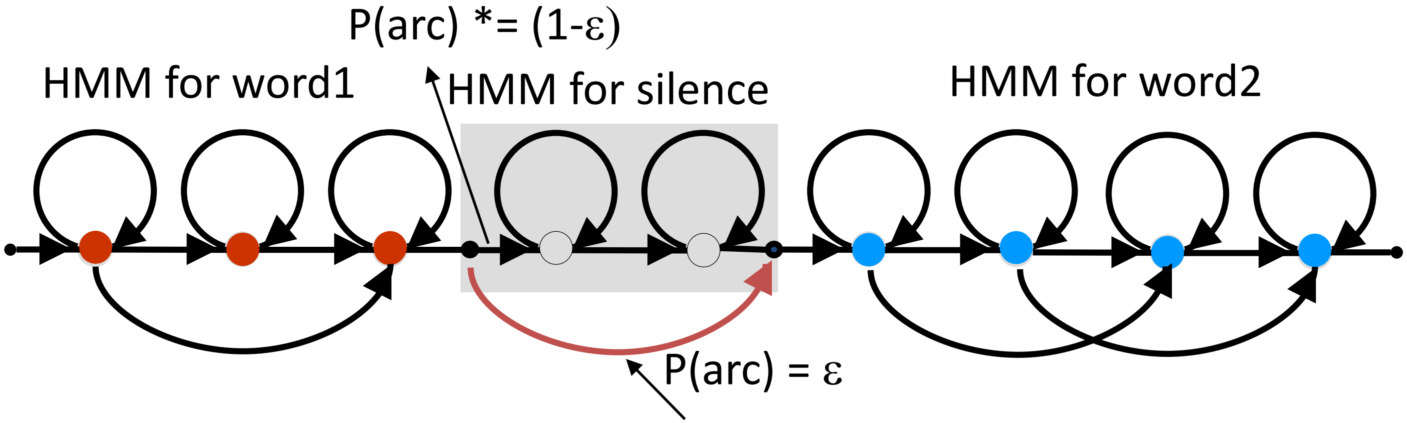


# Training word models from connected words (continuous speech)



- It is often not known a priori if there is a significant pause between words
- Accurate determination will require manually listening to and tagging or transcribing the recordings
  - Highly labor intensive
  - Infeasible if the amount of training data is large

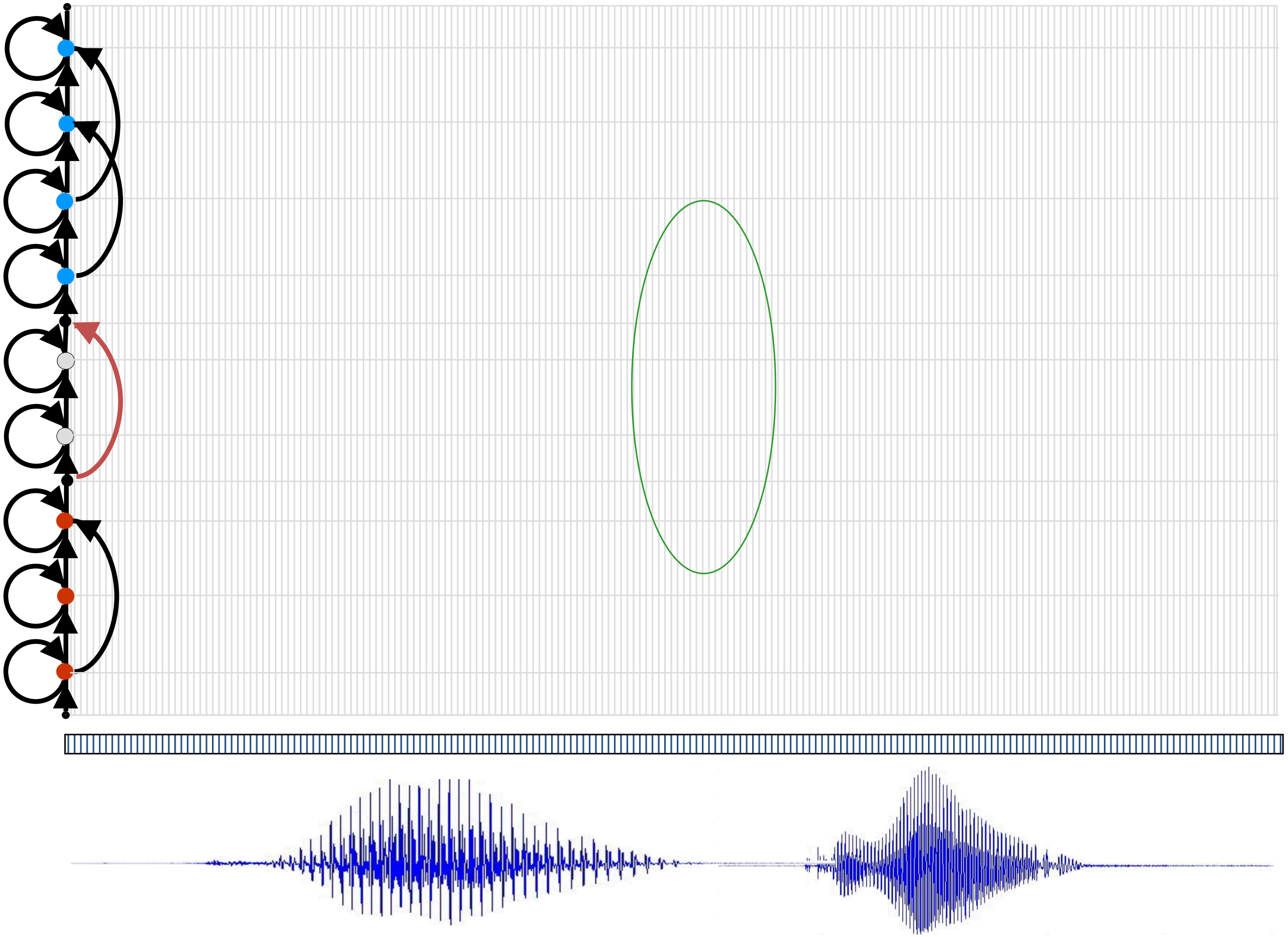
# Training word models from connected words (continuous speech)



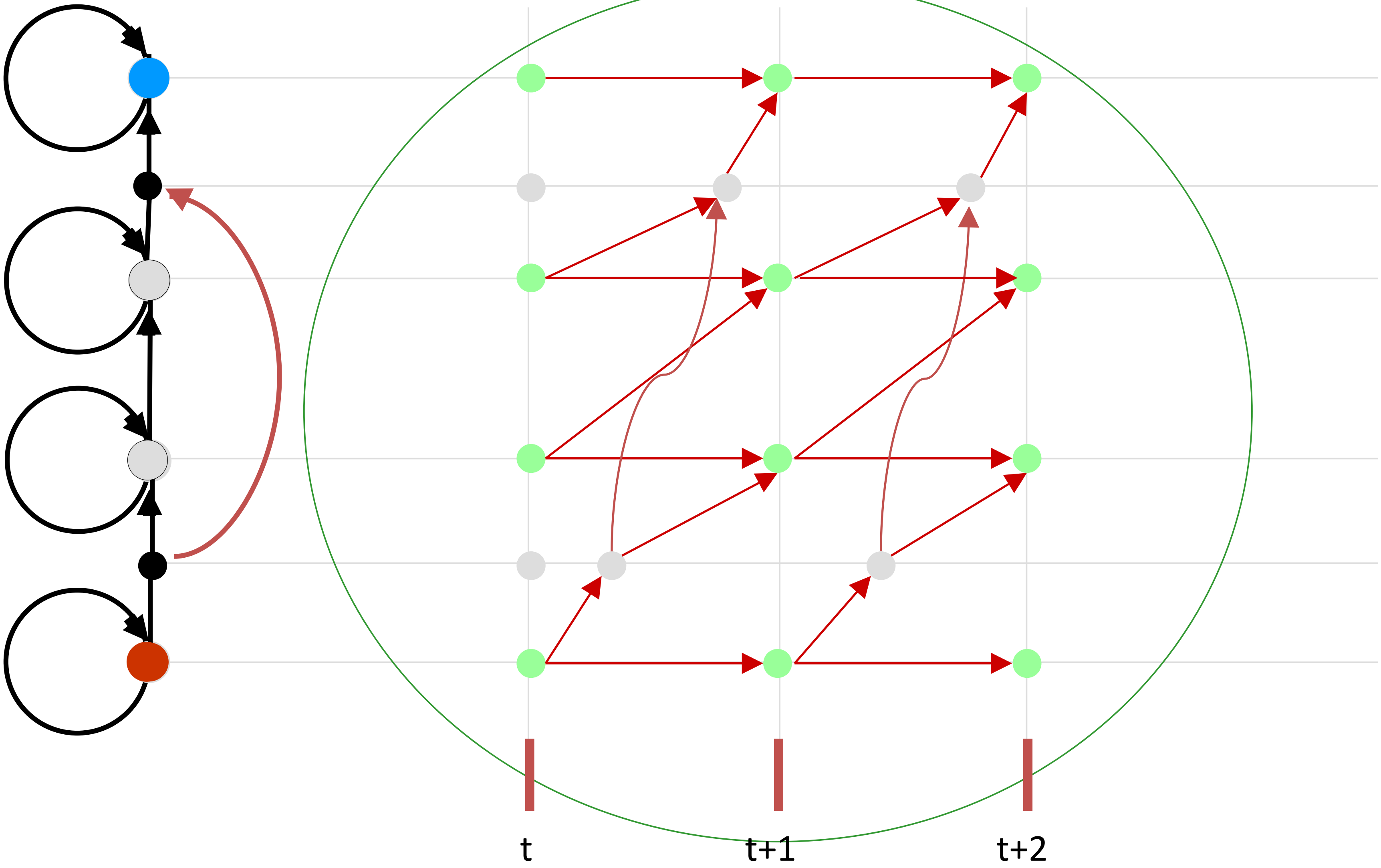
This arc permits direct entry into word2 from word1, without an intermediate pause.

- Fortunately, it is not necessary to explicity tag pauses
- The HMM topology can be modified to accommodate the uncertainty about the presence of a pause
  - The modified topology actually represents a combination of an HMM that does not include the pause, and an HMM that does
  - The probability ε given to the new arc reflects our confidence in the absence of a pause. The probability of other arcs from the non-emitting node must be scaled by 1- ε to compensate

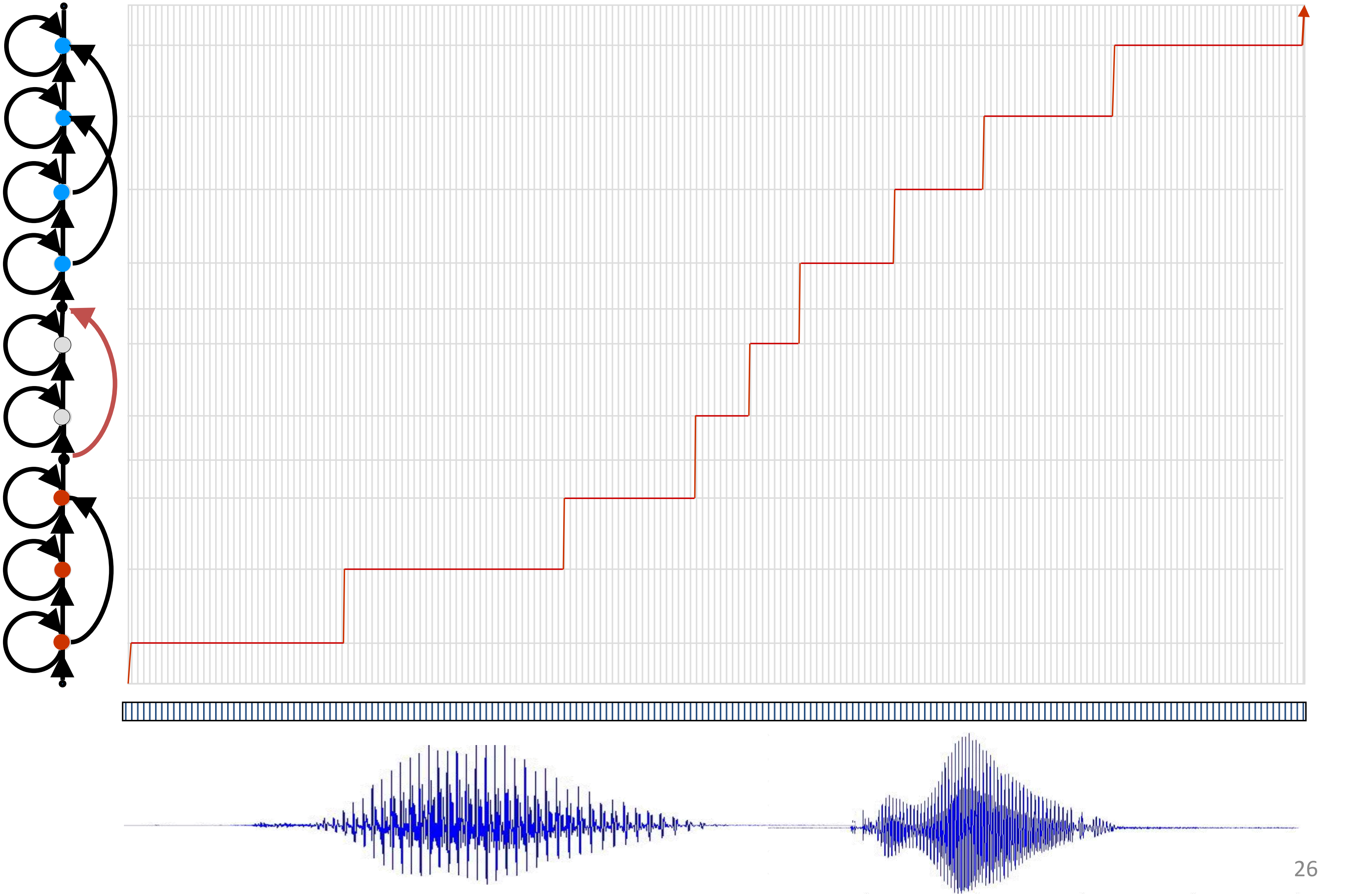
# Training word models from connected words (continuous speech)



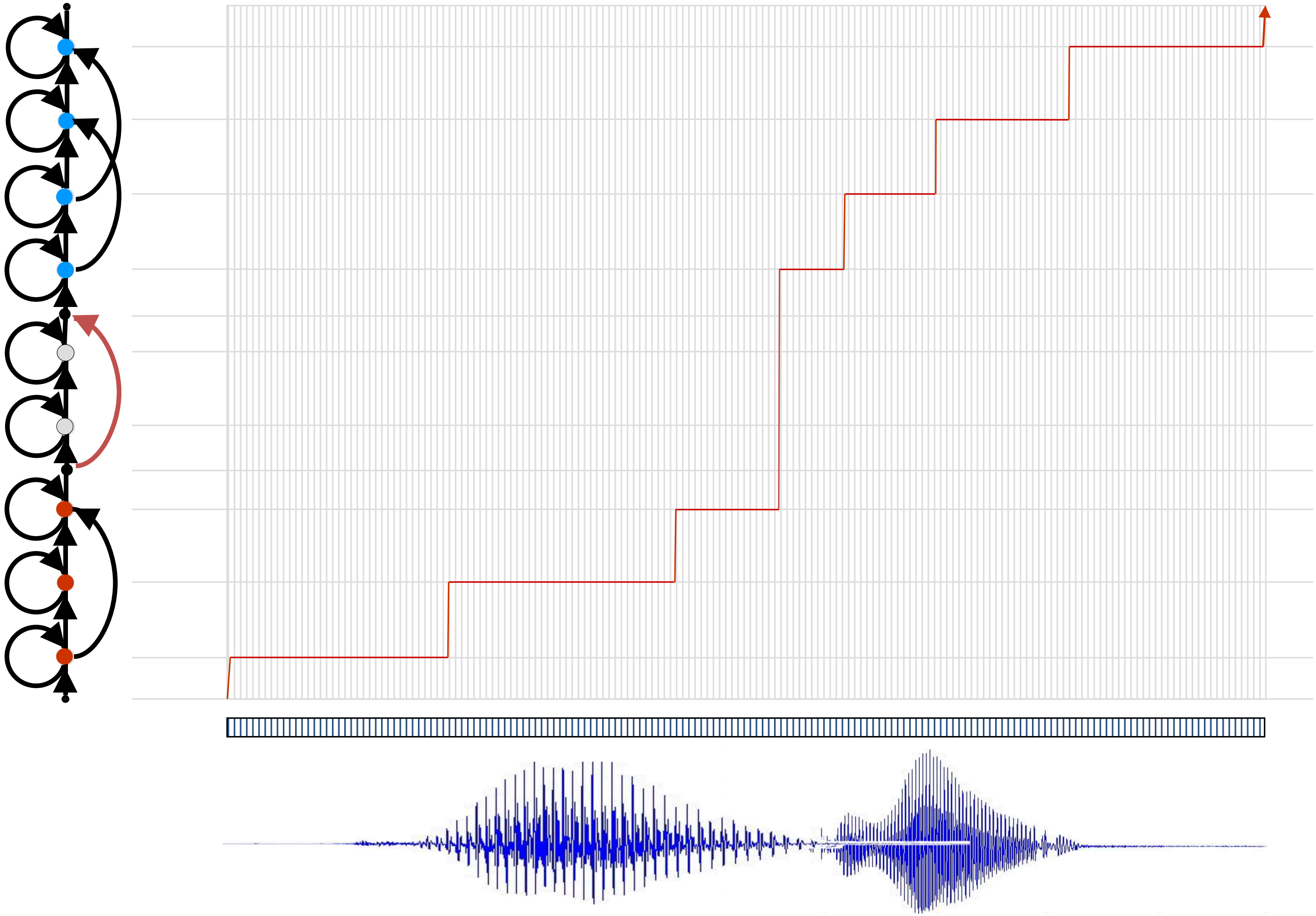
# Training word models from connected words (continuous speech)



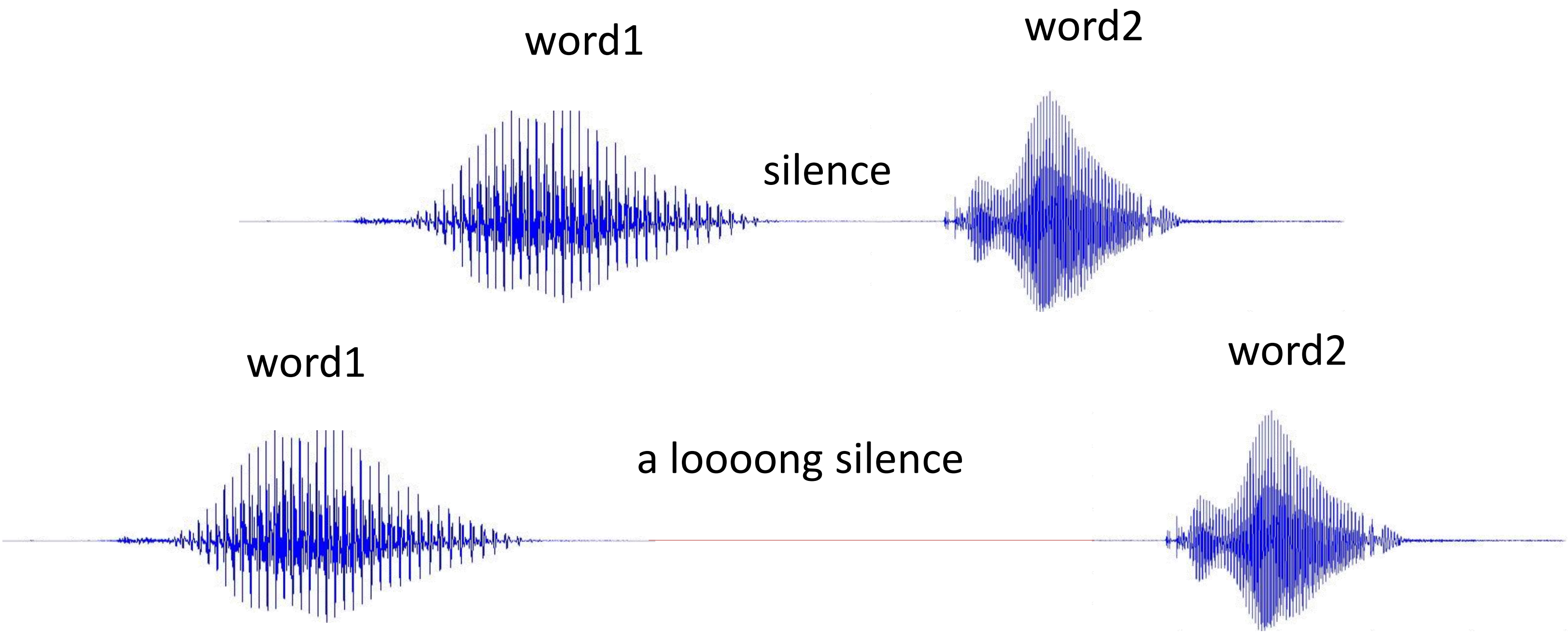
# Training data has a pause



# Training data has no pause

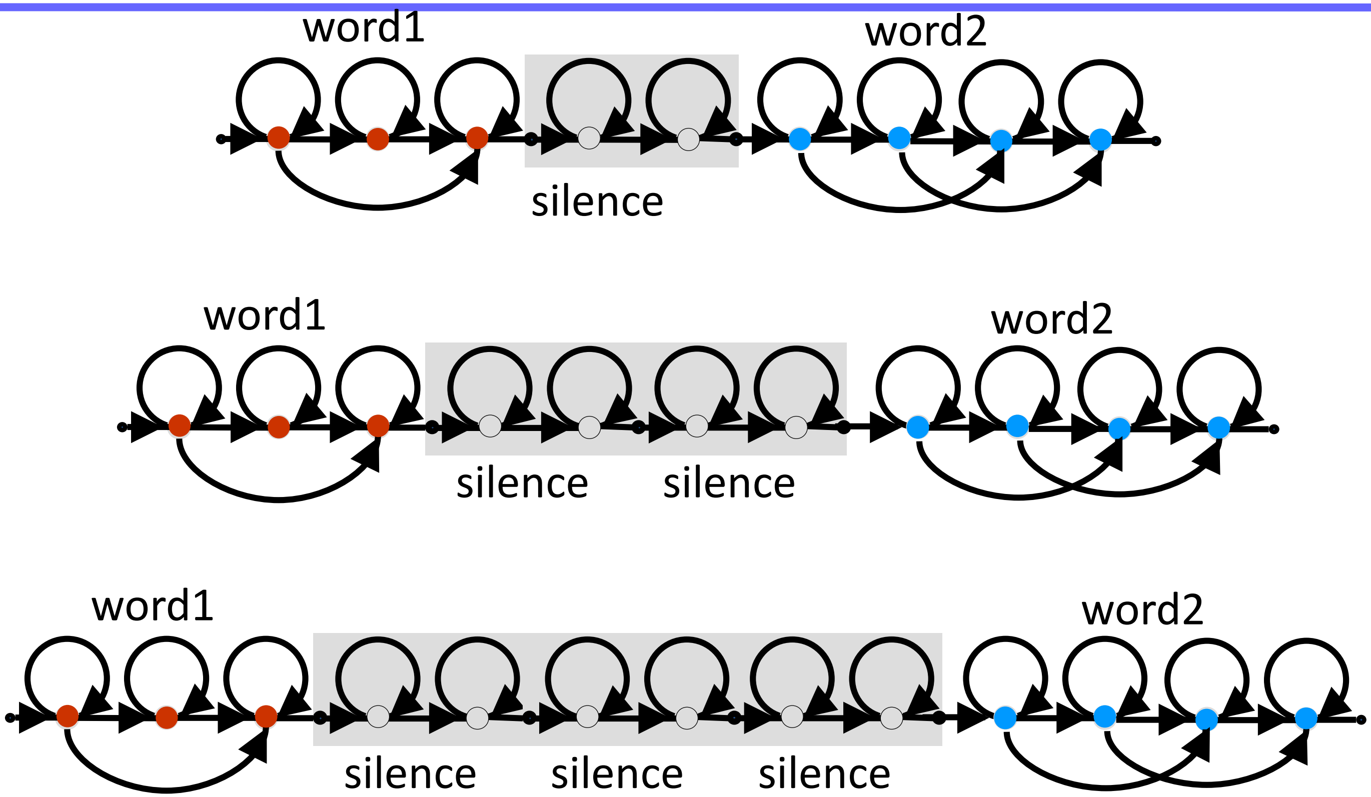


# Training word models from connected words (continuous speech)



- Are long pauses equivalent to short pauses?
- How long are the pauses
  - These questions must be addressed even if the data are hand tagged
- Can the same HMM model represent both long and short pauses?
  - The transition structure was meant to capture sound with specific duration patterns
  - It is not very effective at capturing arbitrary duration patterns

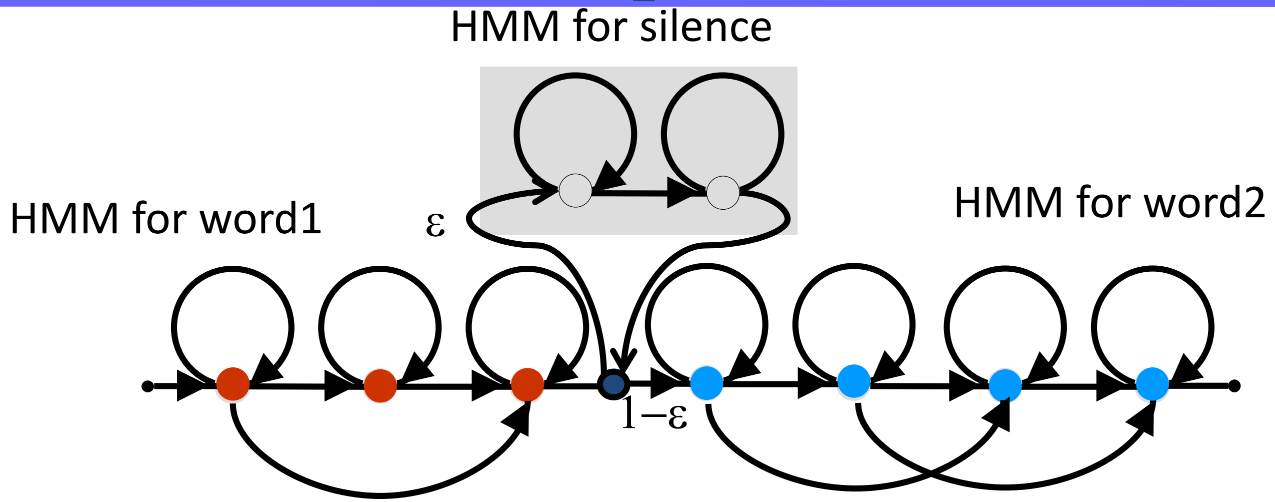
# Accounting for the length of a pause



One solution is to incorporate the silence model more than once. A longer pause will require more repetitions of the silence model

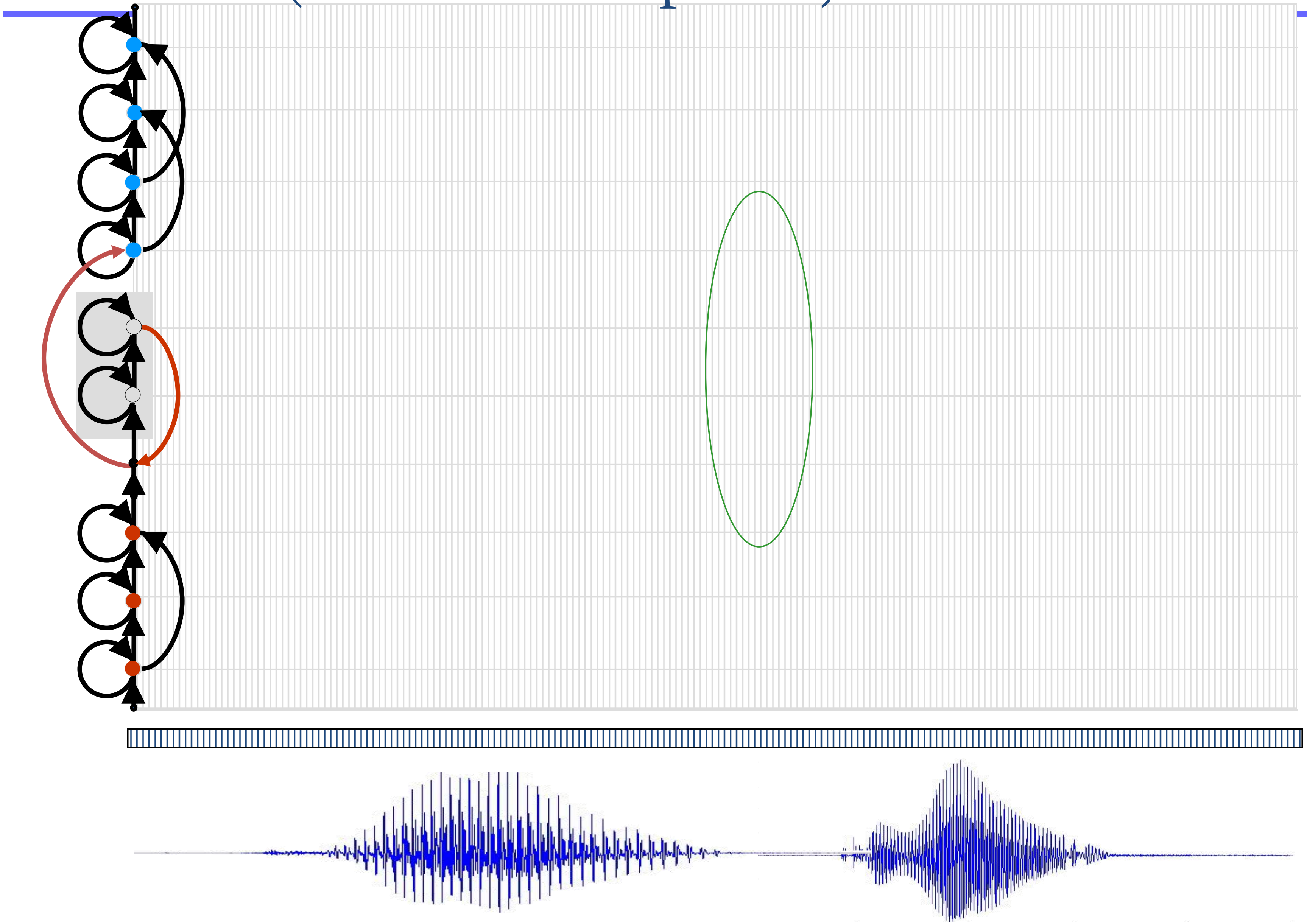
The length of the pause (and the appropriate number of repetitions of the silence model, cannot however be determined beforehand

# Training word models from connected words (continuous speech)

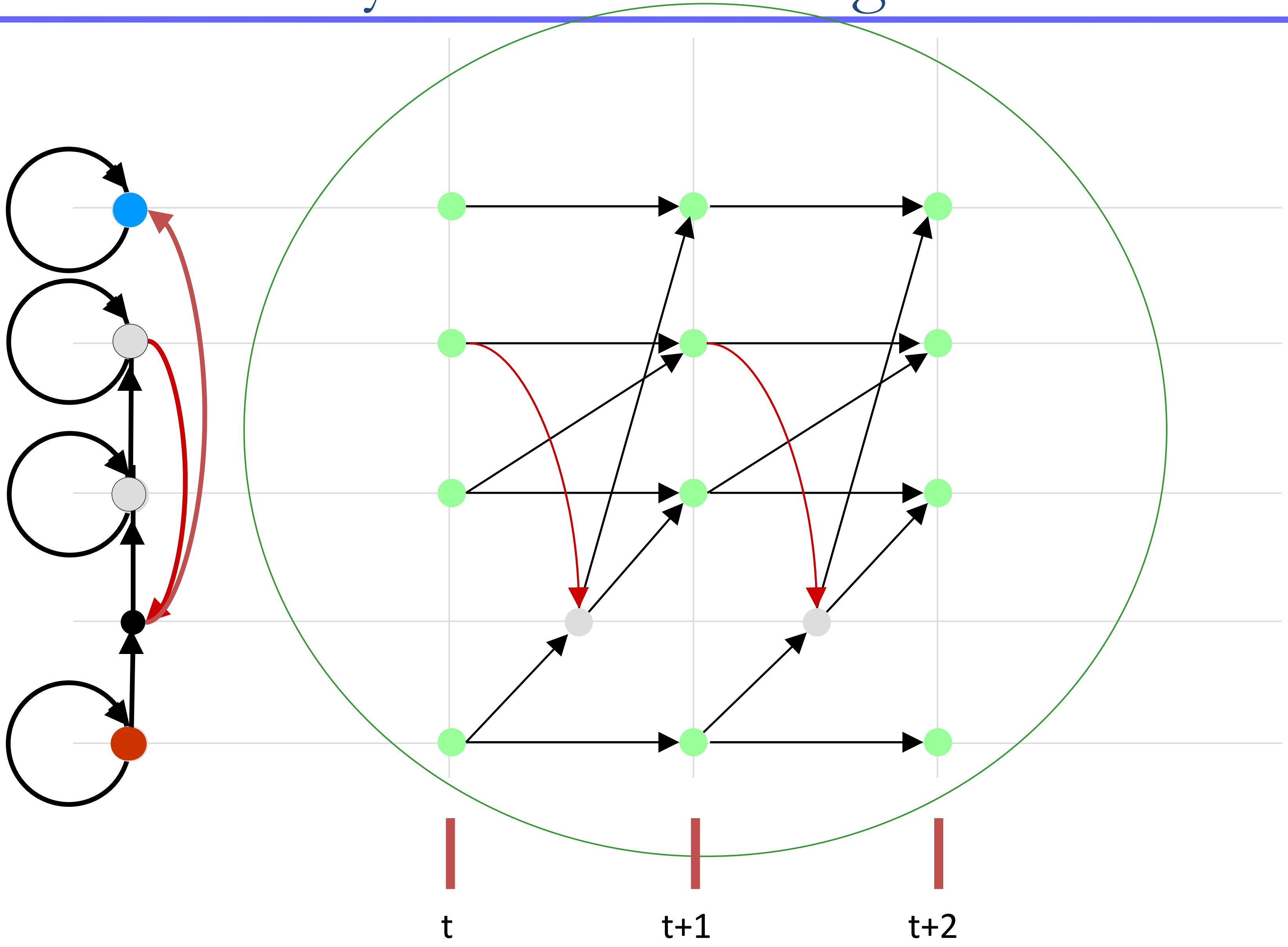


- An arbitrary number of repetitions of the silence model can be permitted with a single silence model by looping the silence
  - A probability ε must be associated with entering the silence model
  - 1- ε must be applied to the forward arc to the next word

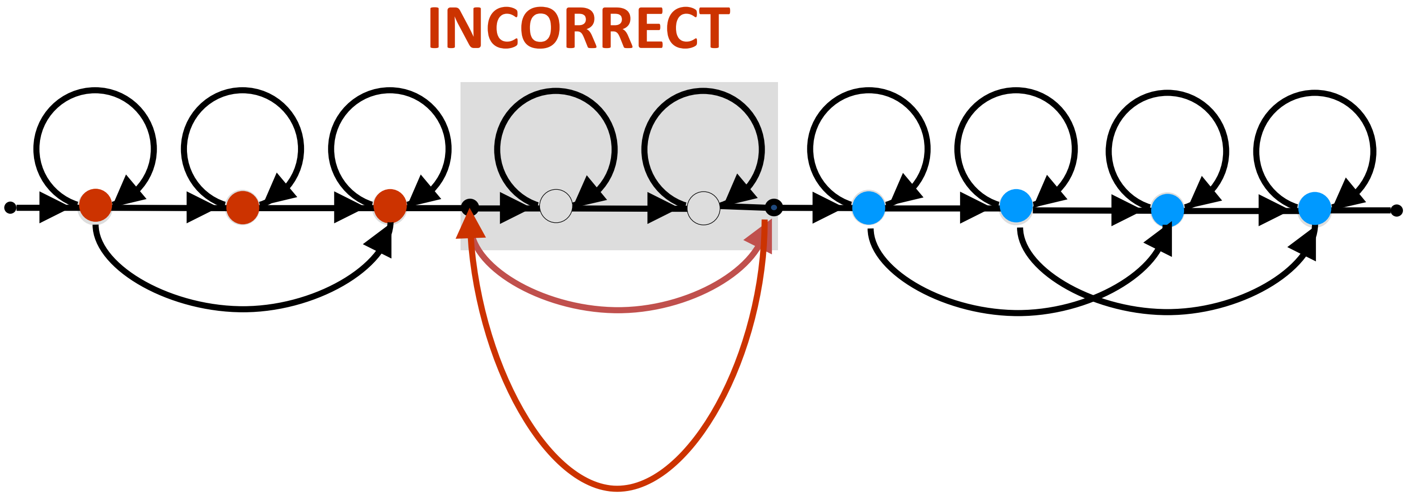
# Training word models from connected words (continuous speech)



# Arranging multiple non emitting states: Note strictly forward-moving transitions

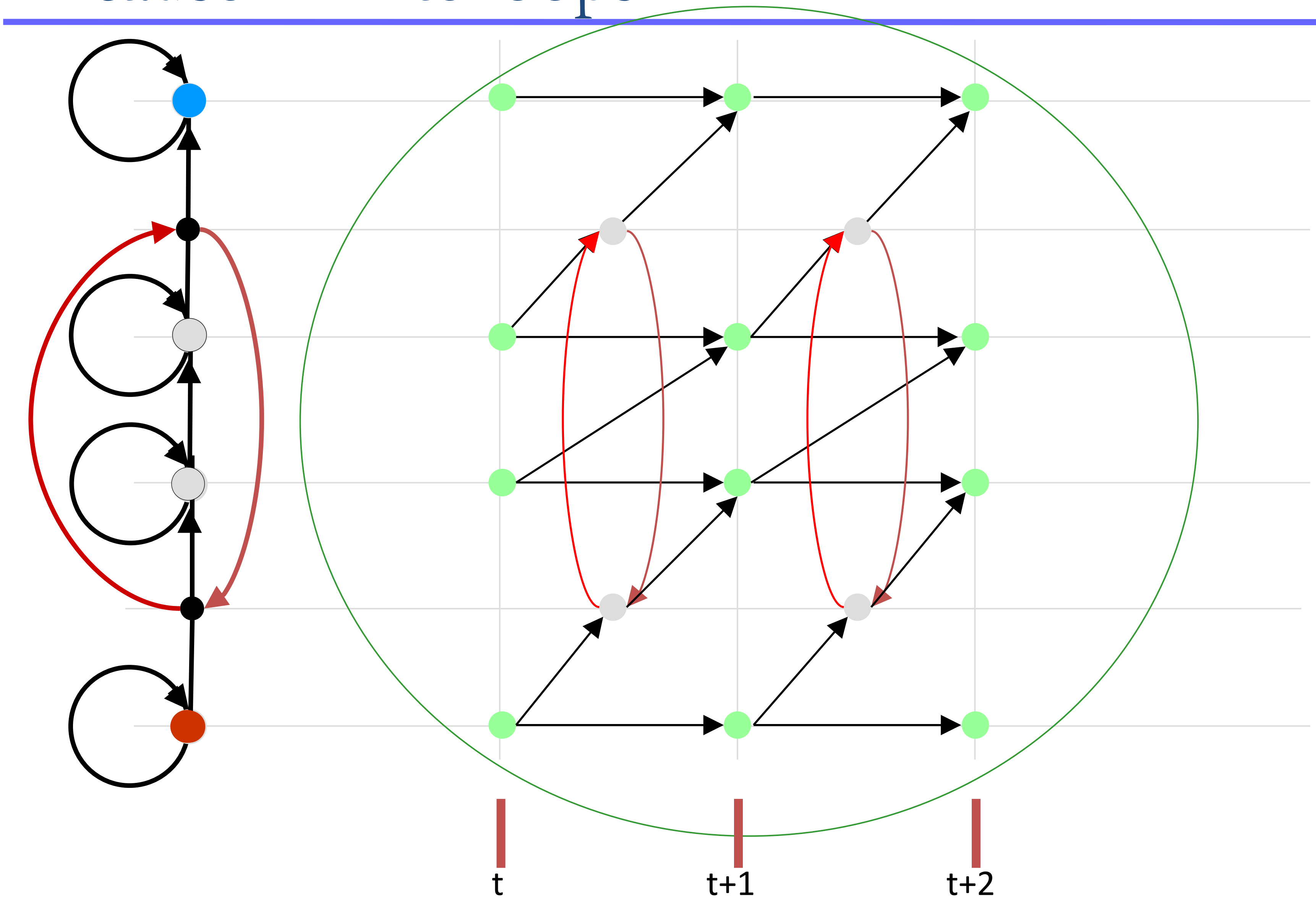


# Training word models from connected words (continuous speech)

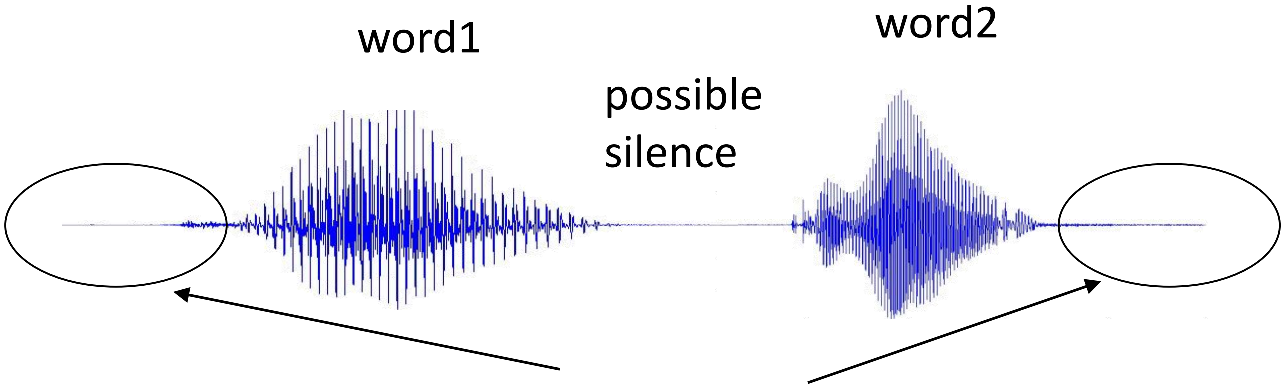


- Loops in the HMM topology that do not pass through an emitting state can result in infinite loops in the trellis
- HMM topologies that permit such loops must be avoided

# Badly arranged non-emitting states can cause infinite loops



# Training word models from connected words (continuous speech)



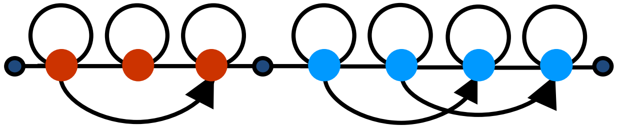
These may be silence regions too (depending on how the recording has been edited)

# Including HMMs for silence

- Must silence models always be included between words (and at the beginnings and ends of utterances)?

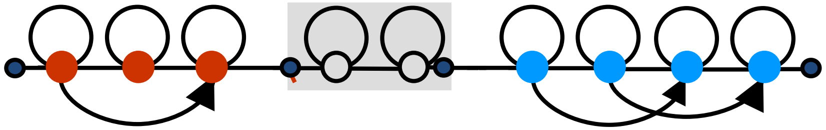
- No, if we are certain that there is no pause

HMM for word1 HMM for word2



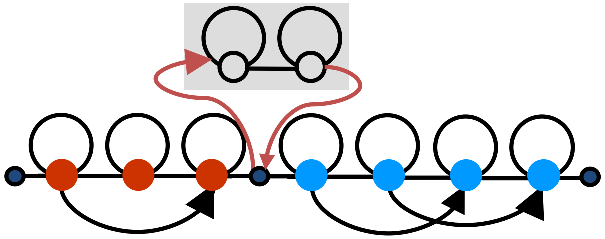
- Definitely (without permitting the skip around the silence) if we are certain that there is a pause

Definite silence



- Include with skip permitted only if we cannot be sure about the presence of a pause

Uncertain silence



- Using multiple silence models is better when we have some idea of length of pause (avoid loopback)

# Initializing silence HMMs

- Initializing silence models
  - While silence models can be initialized like all other models, this is suboptimal
    - Can result in very poor models
  - Silences are important anchors that locate the uncertain boundaries between words in continuous speech
  - Good silence models are critical to locating words correctly in the continuous recordings
  - It is advantageous to pre-train initial silence models from regions of silence before training the rest of the word models

- Identifying silence regions
  - Pre-identifying silences and pauses can be very useful
  - This may be done either manually or using automated methods
  - A common technique is "forced alignment"
    - Alignment of training data to the text using previously trained models, in order to locate silences.

# Segmental K-means Initialization (critical)

- Initialize all word models
  - Assign an HMM topology to the word HMMs
  - Initially assign a Gaussian distribution to the states
  - Initialize HMM parameters for words
    - Segmental K-means training from a small number of isolated word instances
    - Uniform initialization: all Gaussians are initialized with the global mean and variance of the entire training data. Transition probabilities are uniformly initialized.

# Segmental K-means with multiple training utterances

1. For each training utterance construct its own specific utterance HMM, composed from the HMMs for silence and the words in the utterance
   - Words will occur in different positions in different utterances, and may not occur at all in some utterances

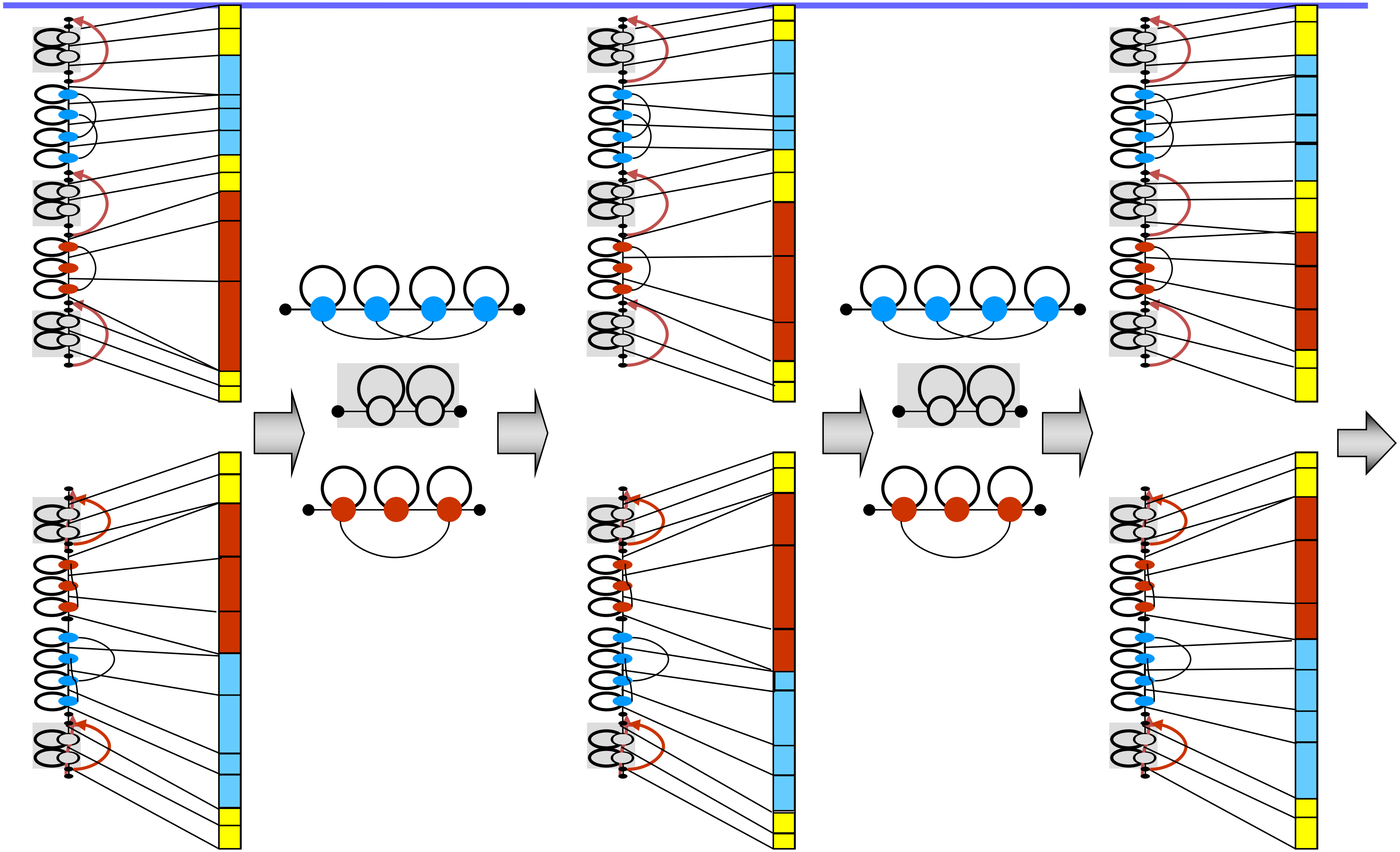
2. Segment each utterance into states using its own utterance HMM

3. Update HMM parameters for every word from all segments, in all utterances, that are associated with that word.

4. If not converged return to 2

5. If the desired number of Gaussians have not been obtained in the state output distributions, split Gaussians in the state output distributions of the HMMs for all words and return to 2

# Segmental K-means with two training utterances



# Practical issue: Trellis size and pruning

- The number of words in an HMM is roughly proportional to the length of the utterance
- The number of states in the utterance HMM is also approximately proportional to the length of the utterance
- $N^2T$ operations are required to find the best alignment of an utterance
  - N is the number of states in the utterance HMM
  - T is the number of feature vectors in the utterance
  - The computation required to find the best path is roughly proportional to the cube of  utterance length
- The size of the trellis is NT
  - This is roughly proportional to the square of the utterance length

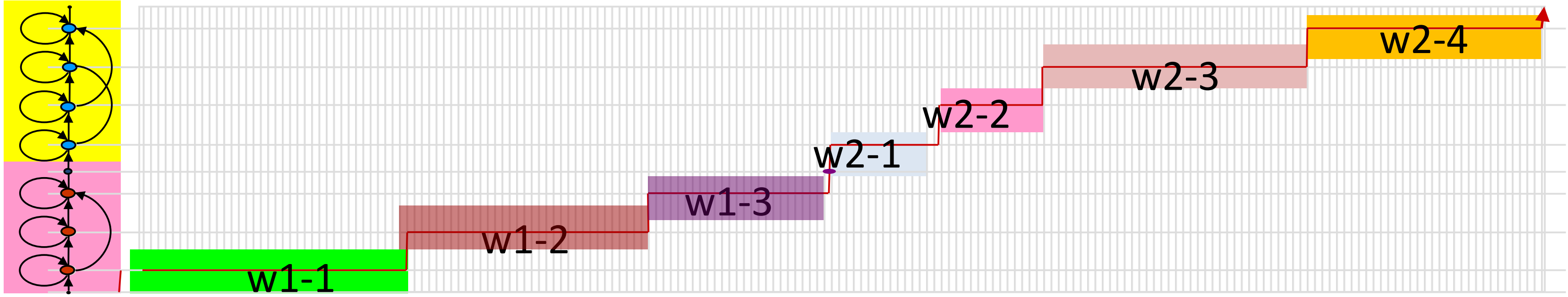
# Baum Welch Training

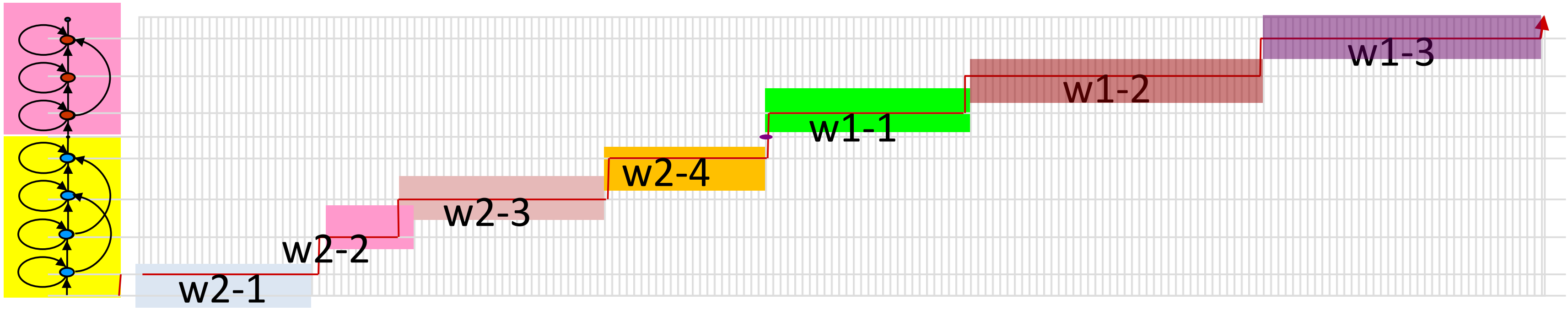
- Isolated word → continuous speech training modifications similar to segmental K-means

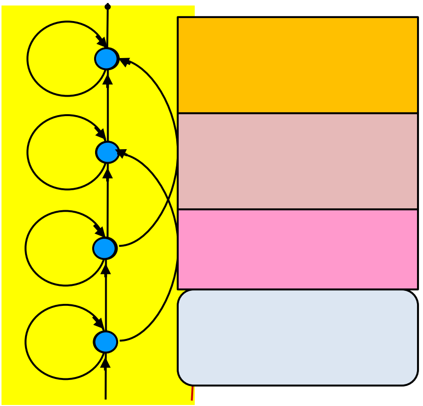
- Compose HMM for sentences
- Perform forward-backward computation on the trellis for the entire sentence HMM

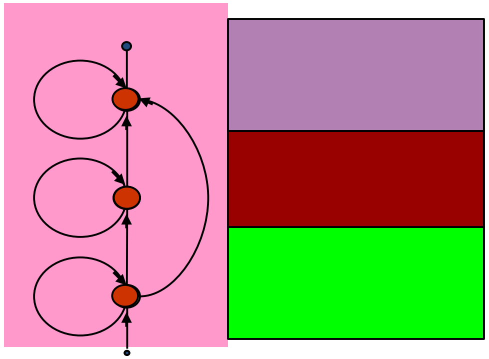
- All sentence HMM states corresponding to a particular state of a particular word contribute to the parameter updates of that state

# Segmental K-Means Training





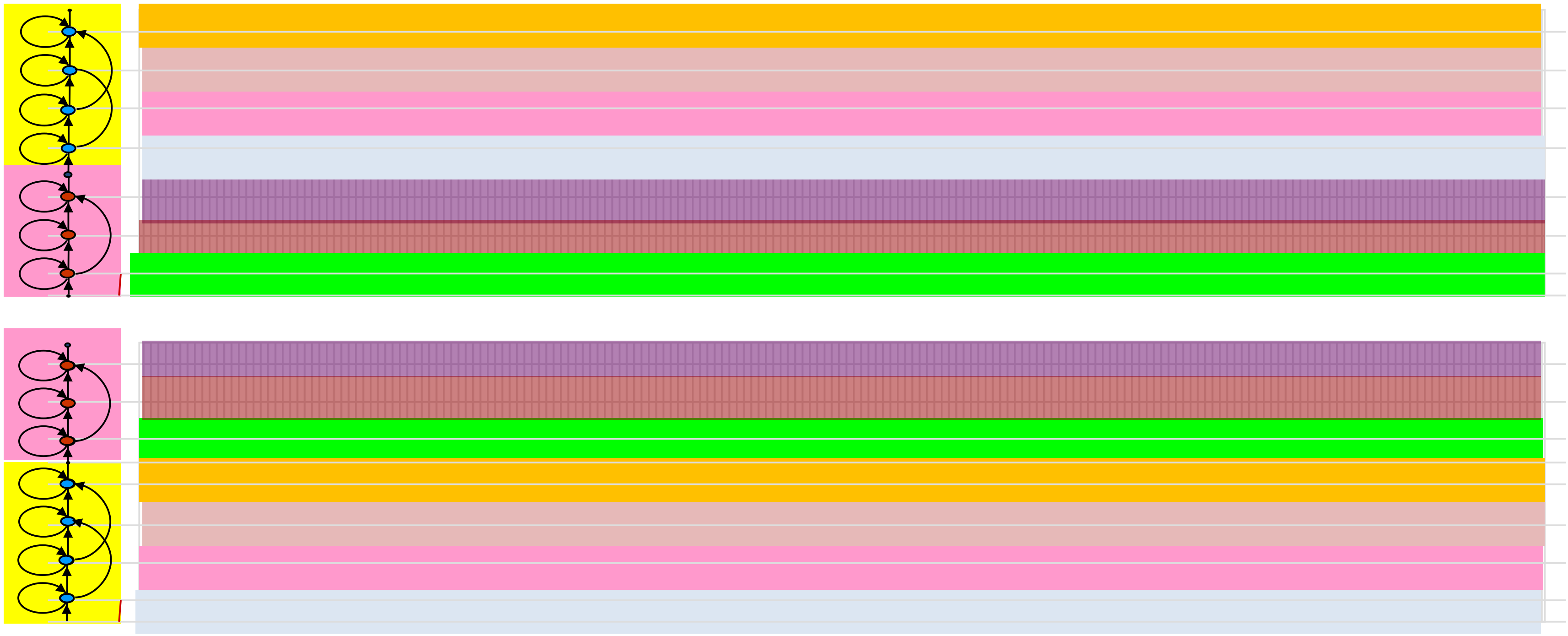


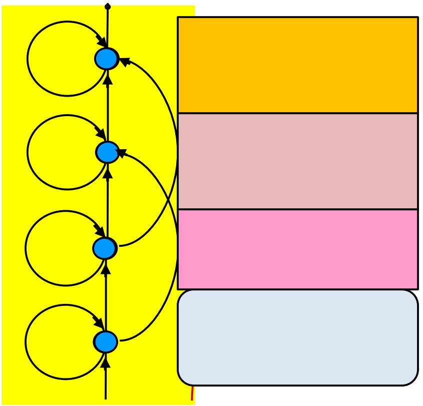


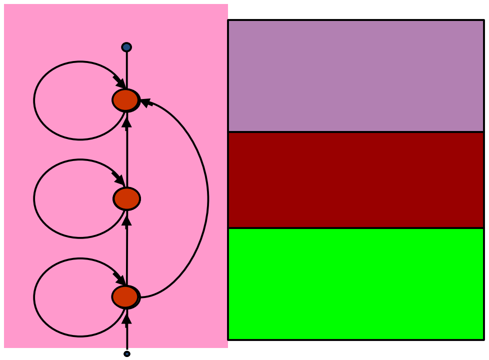
Each state has a "bin" of a particular color

All data segments corresponding to the region of the best path marked by a color are aggregated into the "bin" of the same color

# Baum-Welch Training



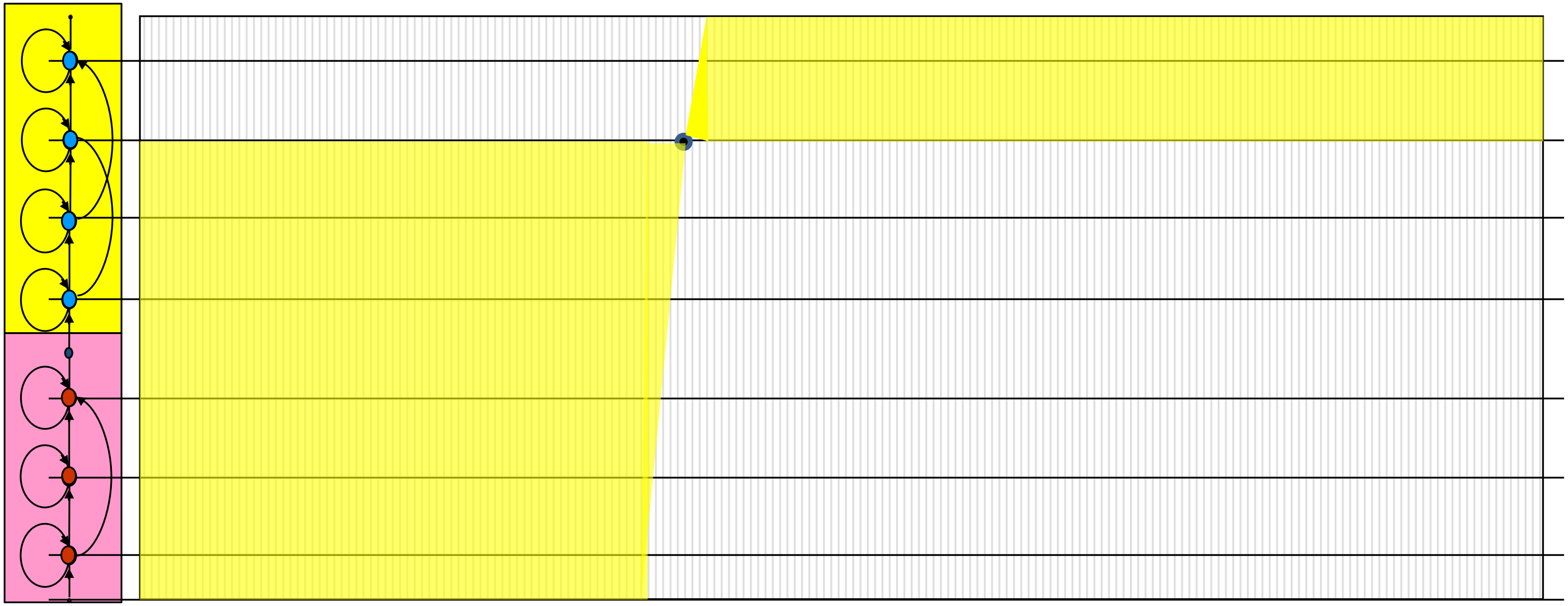




Each state has a "bin" of a particular color

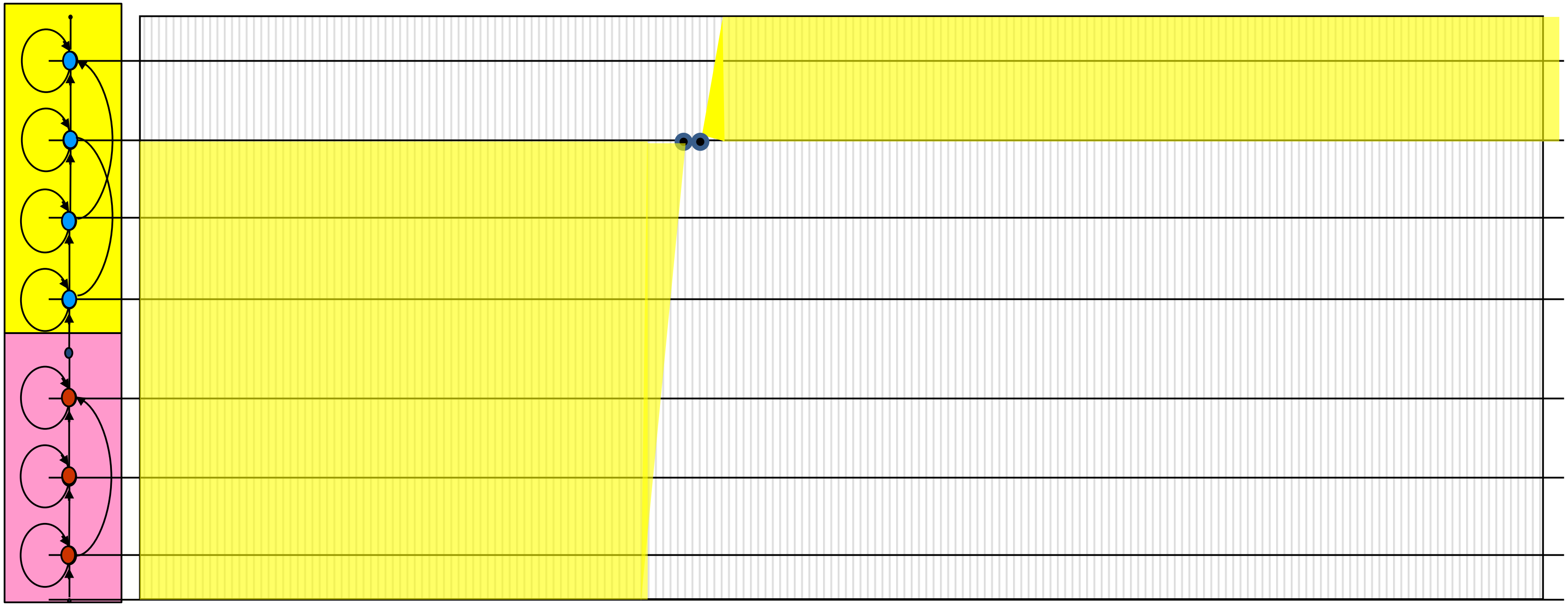
All trellis segments corresponding to the region marked by a color contribute to the corresponding state parameters

# Baum Welch Updates



- $\gamma_w(u,s,t)$ is the a posteriori probability of state $s$ at time $t$, computed for word $w$ in utterance $u$
  - Computed using the forward-backward algorithm
  - If $w$ occurs multiple times in the sentence, we will get *multiple* gamma terms at each instant, one for each instance of $w$

# Baum Welch Updates



- $\gamma_w(u,s,t,s',t+1)$ is the a posteriori probability of state $s$ at time $t$, and $s'$ at $t+1$ computed for word $w$ in utterance $u$
  - Computed using the forward-backward algorithm
  - If $w$ occurs multiple times in the sentence, we will get *multiple* gamma terms at each instant, one for each instance of $w$

# Updating HMM Parameters

$$\mu_{w,s} = \frac{\sum_u \sum_t \gamma_w(u,s,t) x_{u,t}}{\sum_u \sum_t \gamma_w(u,s,t)}$$

$$P(s'|s) = \frac{\sum_u \sum_t \gamma_w(u,s,t,s',t+1)}{\sum_u \sum_t \gamma_w(u,s,t)}$$

$$C_{w,s} = \frac{\sum_u \sum_t \gamma_w(u,s,t)\left(x_{u,t} - \mu_{w,s}\right)\left(x_{u,t} - \mu_{w,s}\right)^T}{\sum_u \sum_t \gamma_w(u,s,t)}$$

- Note: Every observation contributes to the update of parameter values of every Gaussian of every state of every word in the sentence for that recording

- For a single Gaussian per state

# Updating HMM Parameters

$$\mu_{w,s,k} = \frac{\sum_u \sum_t \gamma_w(u,s,k,t) x_{u,t}}{\sum_u \sum_t \gamma_w(u,s,k,t)}$$

$$P(s'|s) = \frac{\sum_u \sum_t \gamma_w(u,s,t,s',t+1)}{\sum_u \sum_t \gamma_w(u,s,t)}$$

$$C_{w,s,k} = \frac{\sum_u \sum_t \gamma_w(u,s,k,t)\left(x_{u,t} - \mu_{w,s}\right)\left(x_{u,t} - \mu_{w,s}\right)^T}{\sum_u \sum_t \gamma_w(u,s,k,t)}$$

- Note: Every observation contributes to the update of parameter values of every Gaussian of every state of every word in the sentence for that recording

- For a Gaussian mixture per state

# BW training with continuous speech recordings

- For each word in the vocabulary
  1. Decide the topology of all word HMMs
  2. Initialize HMM parameters
  3. For each utterance in the training corpus
     - Construct utterance HMM (include silence HMMs if needed)
     - Do a forward pass through the trellis to compute alphas at all nodes
     - Do a backward pass through the trellis to compute betas and gammas at all nodes
     - Accumulate sufficient statistics (numerators and denominators of re-estimation equations) for each state of each word
  4. Update all HMM parameters
  5. If not converged return to 3
- The training is performed jointly for all words

# Some important types of training

- Maximum Likelihood Training
  - Baum-Welch
  - Viterbi
- Maximum A Posteriori (MAP) training
- Discriminative training
- On-line training

- Speaker adaptive training
  - Maximum Likelihood Linear Regression