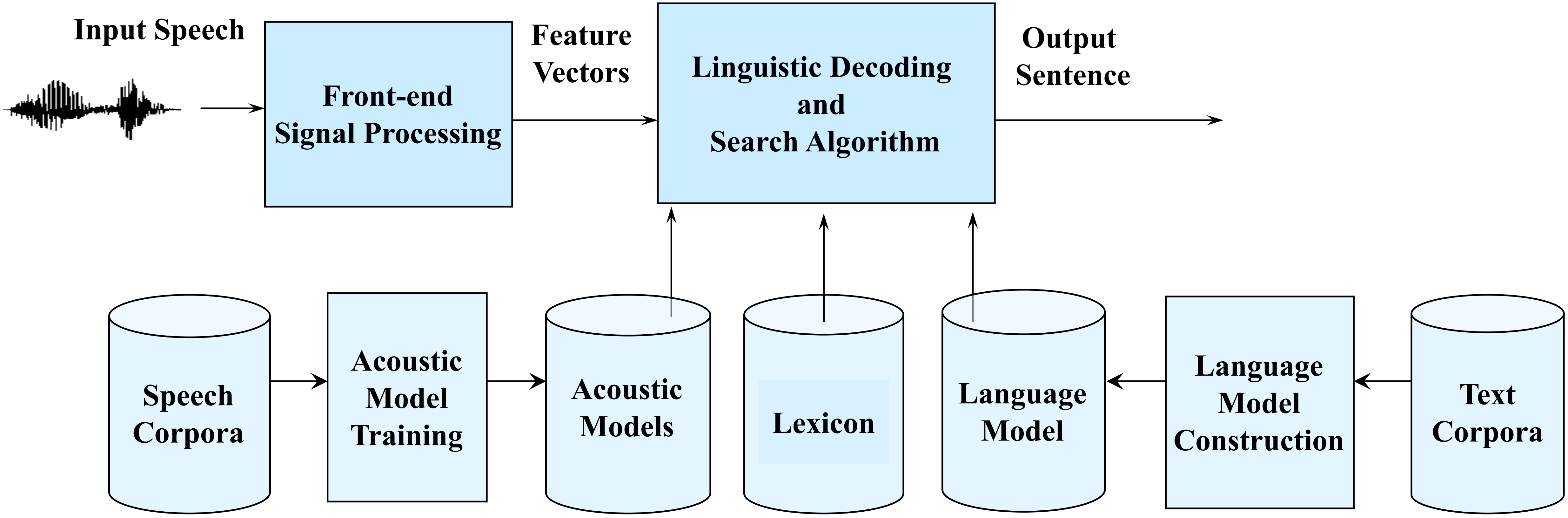
Automatic Speech Recognition

Presented by Yan Li

Outline

- Overview of ASR system
- Hidden Markov Models (HMM)
- HMM + Artificial Neural Network
- Connectionist temporal classification

Overview of ASR system



The function of each component in ASR

→ This is speech

- Acoustic Models: phoneme (音素) recognition

  (th-ih-s-ih-z-s-p-ih-ch)

- Lexicon: a sequence of phonemes to possible words

  (th-ih-s) → this

  (ih-z) → is

  (s-p-iy-ch) → speech

- Language Model: the probability of words given text corpora

  P(this) P(is | this) P(speech | this is)

  $P(w_i|w_{i-1})$ bi-gram language model

  $P(w_i|w_{i-1},w_{i-2})$ tri-gram language model, etc

Overview of ASR system

- W = (w_1,w_2,...,w_U) a word sequence
- O = (o_1,o_2,...,o_T), feature vectors from a speech utterance
- Aim: find best word sequence based on maximum posterior probability

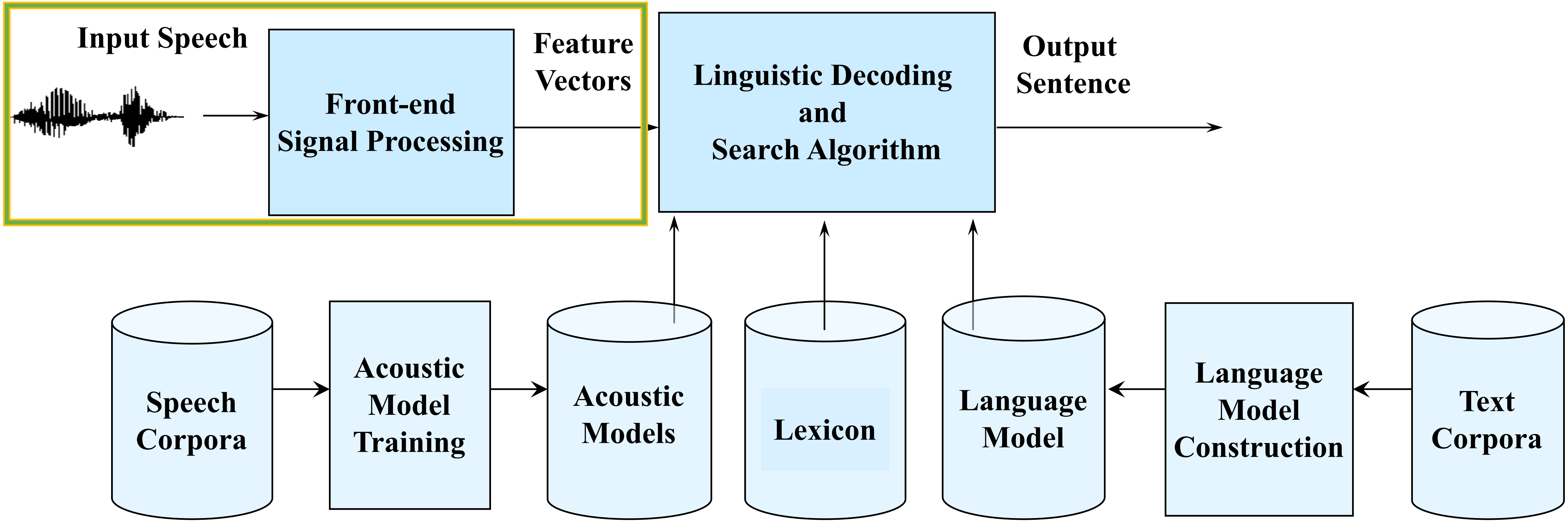
$$W^* = \operatorname*{Arg\,max}_{W} P(W|O)$$

From Acoustic Models

Where $$P(W|X) = \frac{P(O|W)P(W)}{P(O)}$$

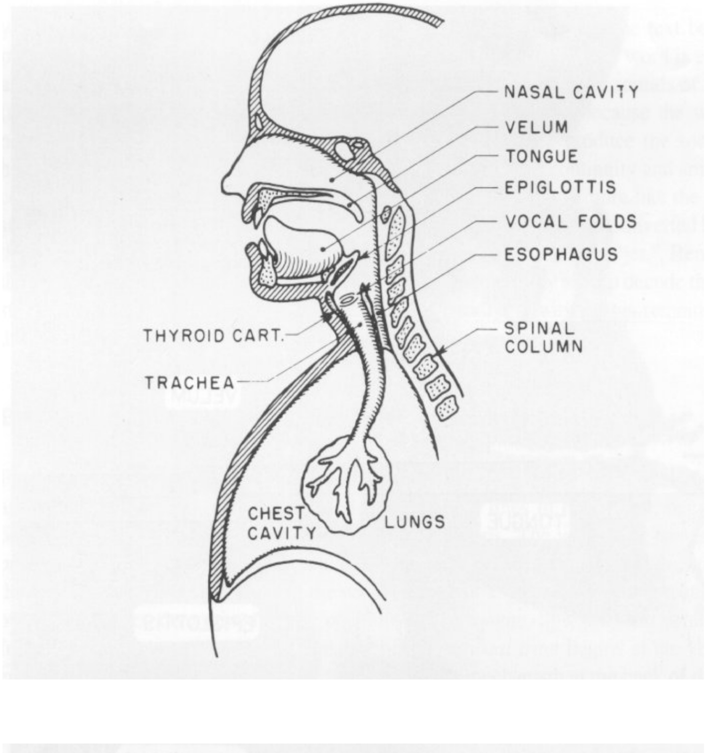
From Language Models

Feature extraction from speech

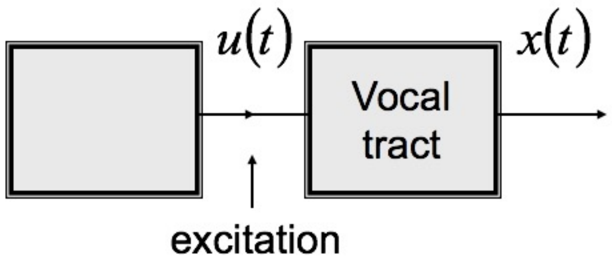


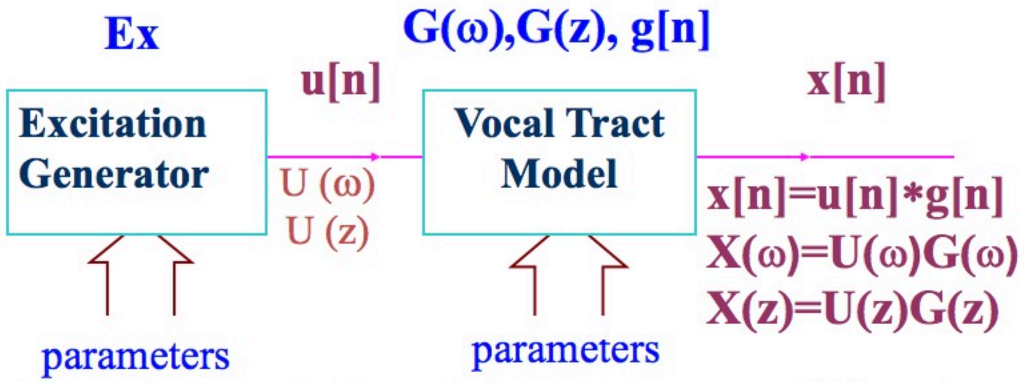
Speech Production and Source Model

- Human vocal mechanism

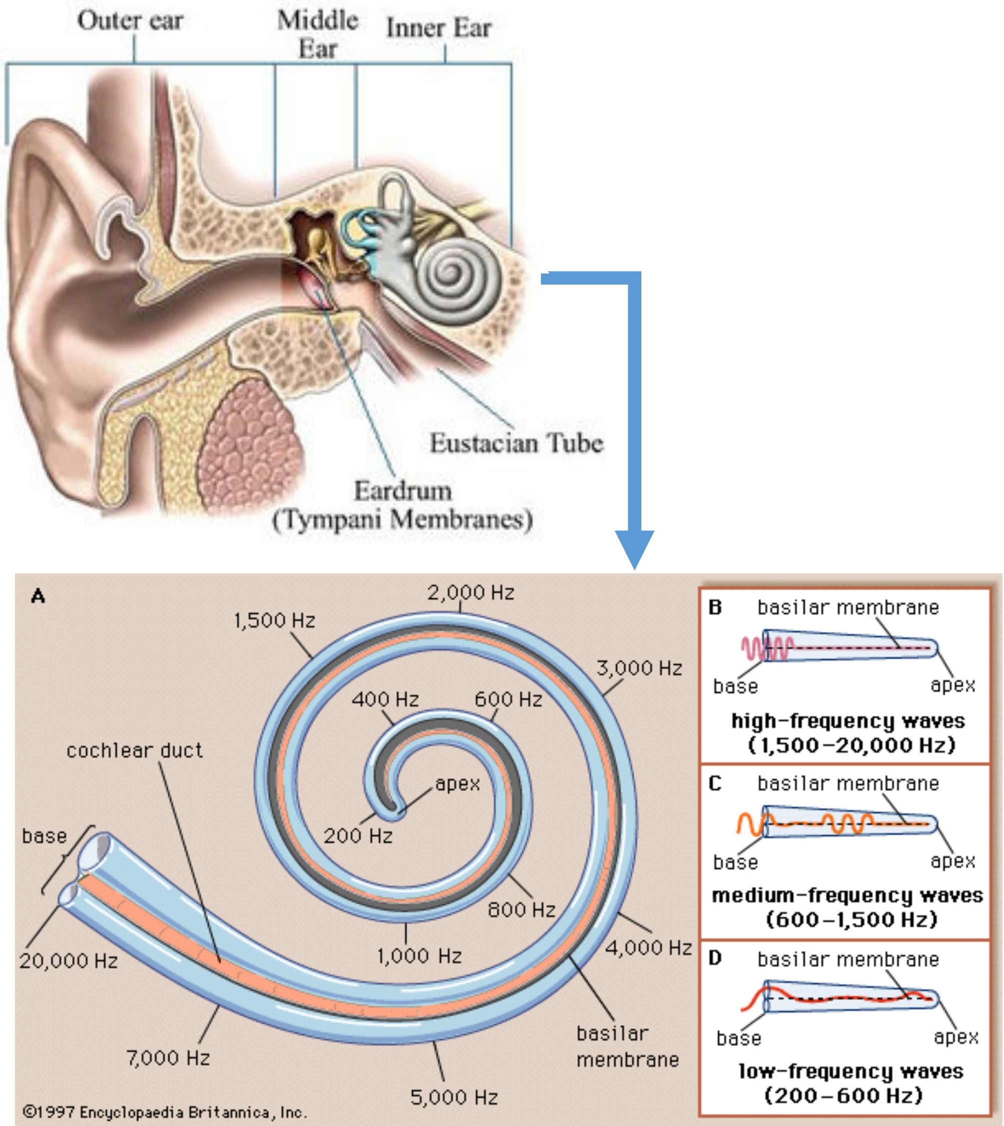


- Speech Source Model

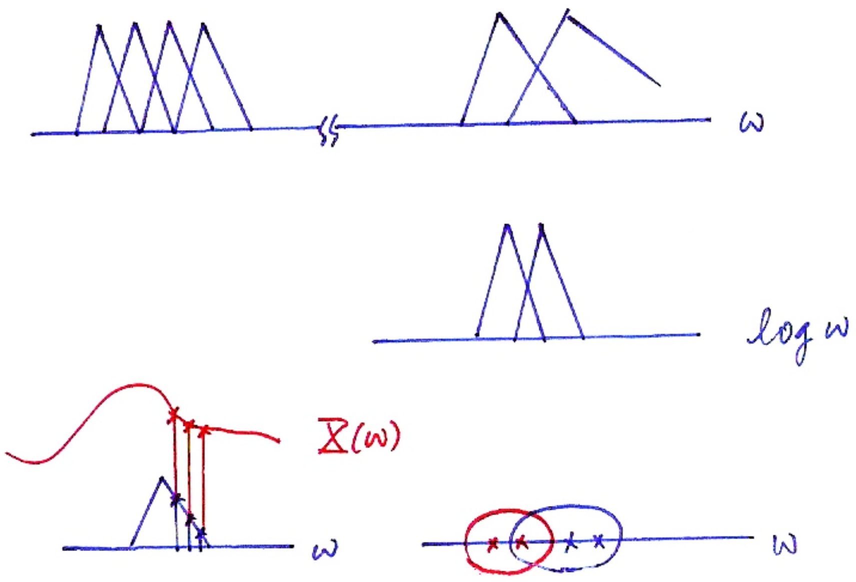




Peripheral Processing for Human Perception

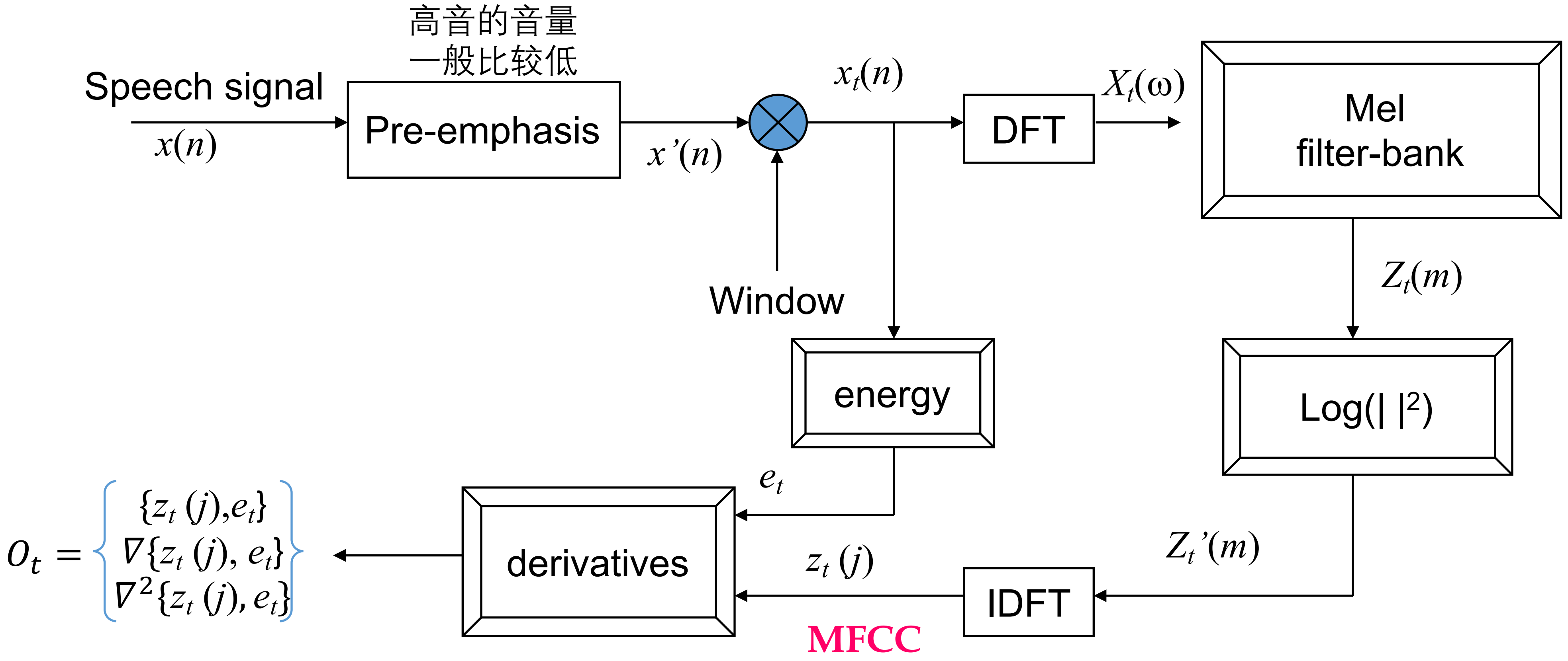


Mel-scale Filter Bank simulates human ear perception



- This frequency band is referred to as the critical band.
- These critical bands somehow overlap with each other.
- The critical bands are roughly distributed linearly in the logarithm frequency scale (including the center frequencies and the bandwidths), specially at higher frequencies.

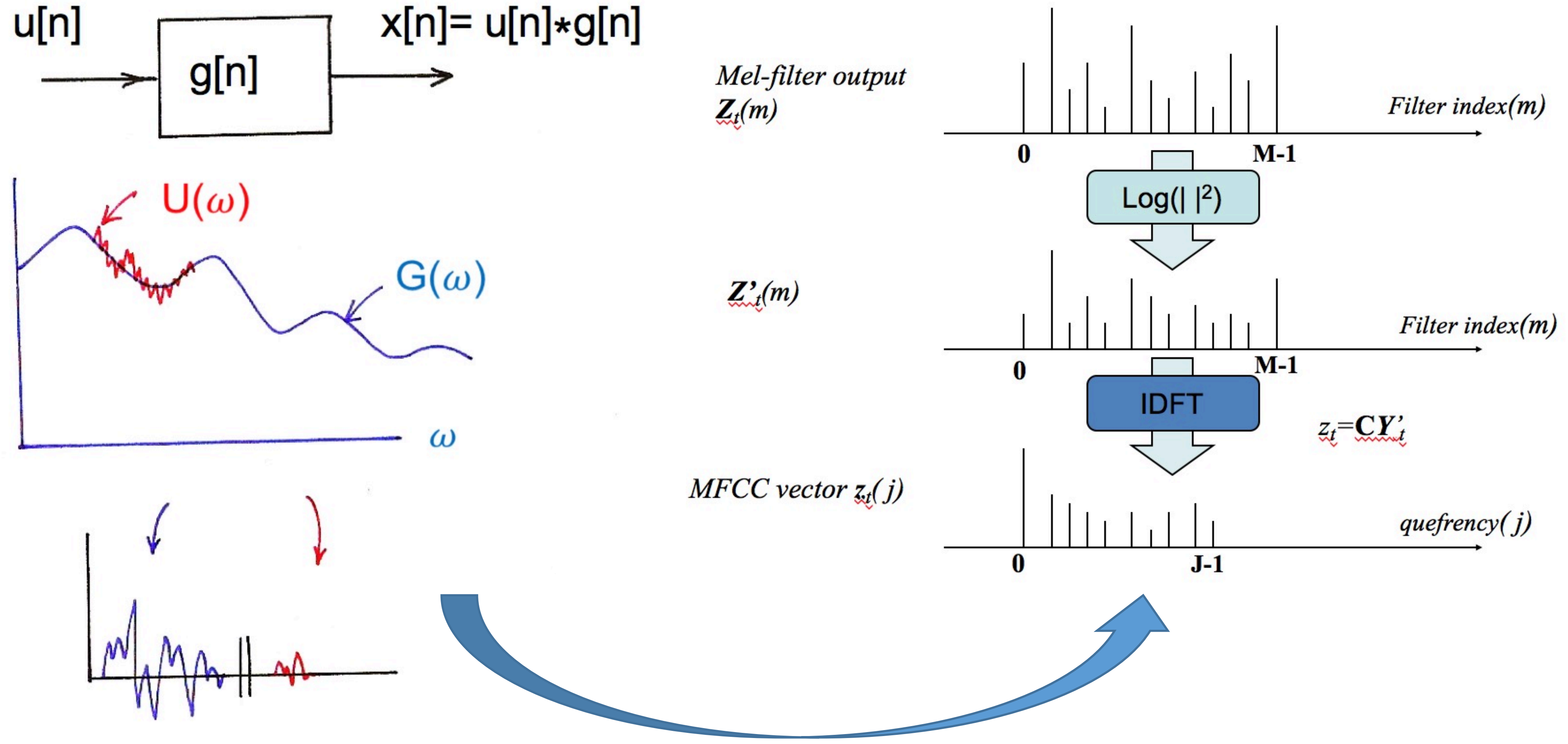
The overview of MFCC generation



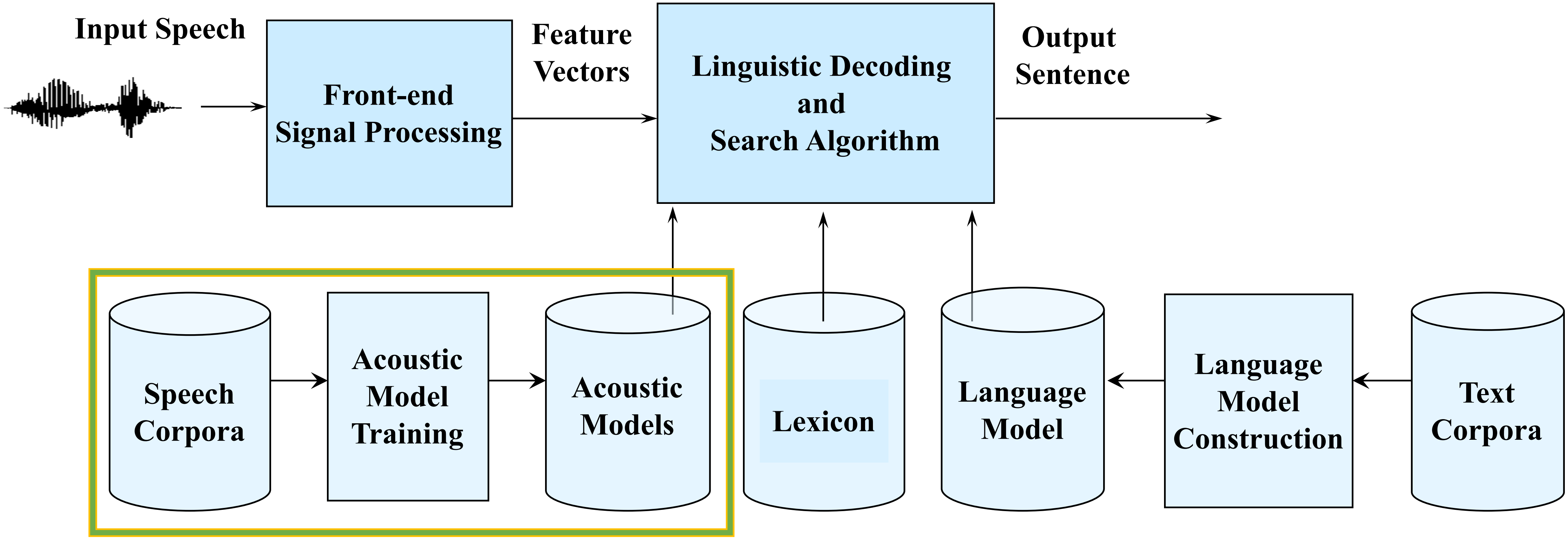
$x(n)=u(n)*g(n)$ ➔ $X(\omega)=U(\omega)G(\omega)$

➔ $|X(\omega)|=|U(\omega)||G(\omega)|$ ➔ $\log|X(\omega)|=\log|U(\omega)|+\log|G(\omega)|$

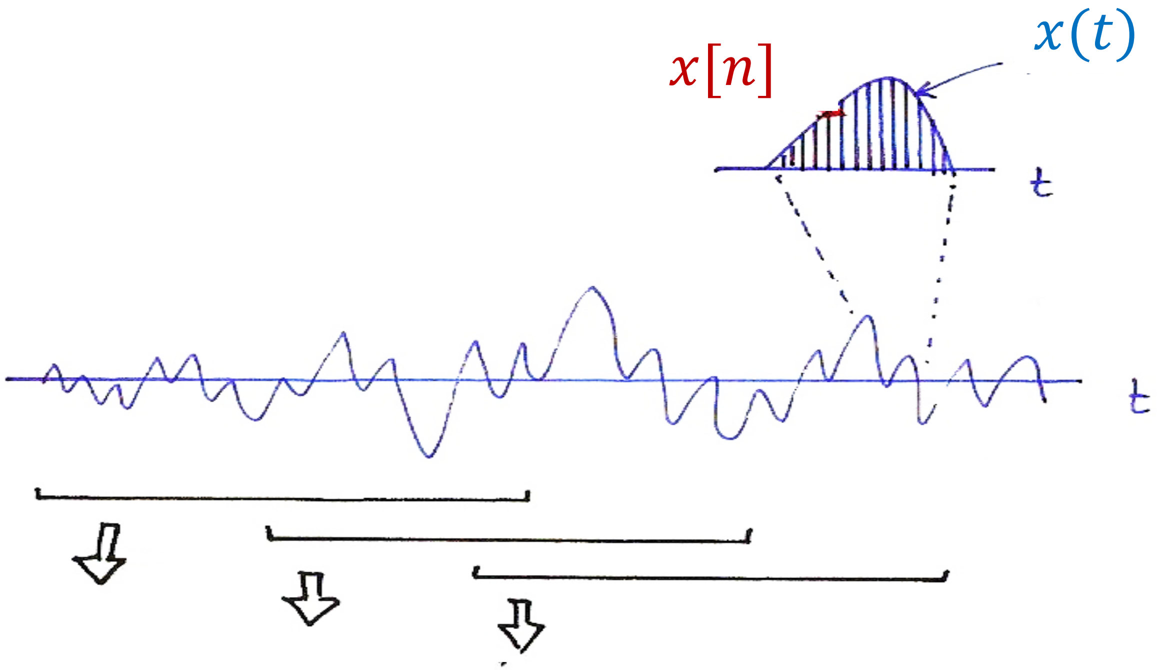
The final process of MFCC generation



Acoustic Models



The input of Acoustic Models



$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_D \end{bmatrix} = o_1 \quad \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_D \end{bmatrix} = o_2 \quad \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_D \end{bmatrix} = o_3$$

Research Challenge:

- Not independent:
  The individual data points cannot be assumed to be independent.
- Not one-to-one labeling:
  说话有快有慢，发言有长有短
  The target sequence $W^* = (w_1, w_2, \ldots, w_U)$ is at most as long as the input sequence $O = (o_1, o_2, \ldots, o_T)$, i.e. $|W| = U \leq |O| = T$.
- The Alignments of input and target are not given.

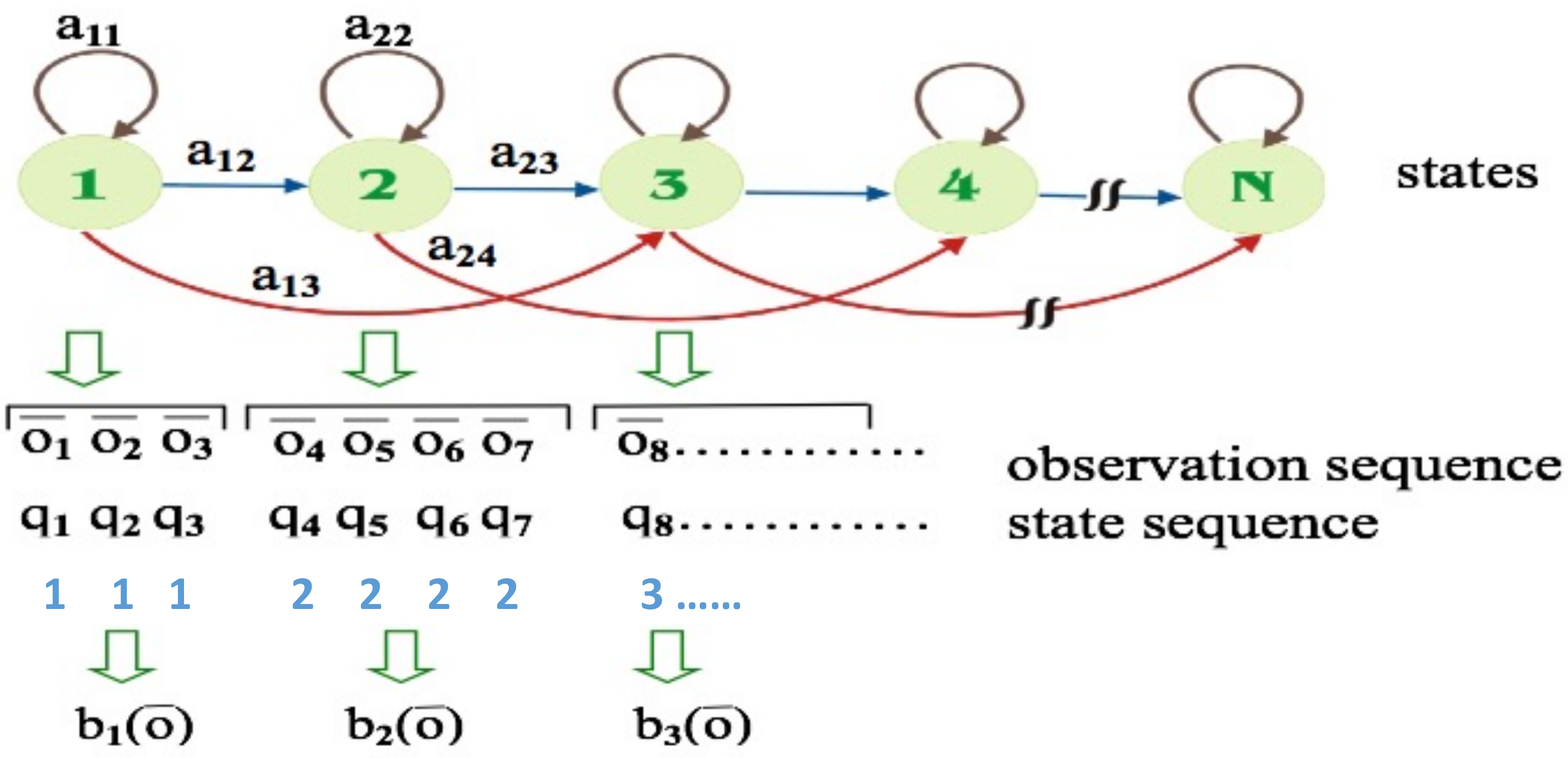
Unit selection for Acoustic Model

- Built acoustic model for each Word:
  - Need to collect enough acoustic data for each word
  - Cannot deal with OOB word
  - There will be a huge number of acoustic models to be learned.
- Built acoustic model for each phoneme:
  - The number of phonemes in each language is limited (44 in English, 普通话有7个元音，22个辅音，4个音调＋轻音)
  - It is not easy to deal with coarticulation (协同发音).
    - One solution: built tri-phone instead. For a same phoneme if it’s prefix and suffix are different, built a corresponding phoneme instead.
- For mandarin may be we can build an acoustic model for each syllable,(中文字都是单音节) each syllable consists of an optional initial consonant + vowel (accompanied by tone) + optional final consonant (*n* or *ng*)

Outline

- Overview of ASR system
- Hidden Markov Models (HMM)
- HMM + Artificial Neural Network
- Connectionist temporal classification

HMM



The HMM have been used in acoustic models because multiple observation sequence can corresponds to a single state. Therefore, the challenge of output dependency and not one-to-one labeling can be tackled.

HMM+GMM

- The HMM has two types of parameters:

  $A = [a_{i,j}]$ is the state transition probability matrix, where $a_{i,j} = P(q_t = j | q_{t-1} = i)$

  $B = [b_j(o), j = 1,2, \dots, N]$ is the observation (emission) probability

- Originally the emission probability is approximated via a Gaussian Mixture Model (GMM). For each state in the HMM we will build a GMM with M number of components.

$$b_j(o) = \sum_{k=1}^{M} c_{jk} b_{jk}(o)$$

  Where $b_{jk}(o)$ is the multi-variance Gaussian distribution for the k-th mixture Gaussian of the j-th state, and c_{jk} is the corresponding weight scalar.

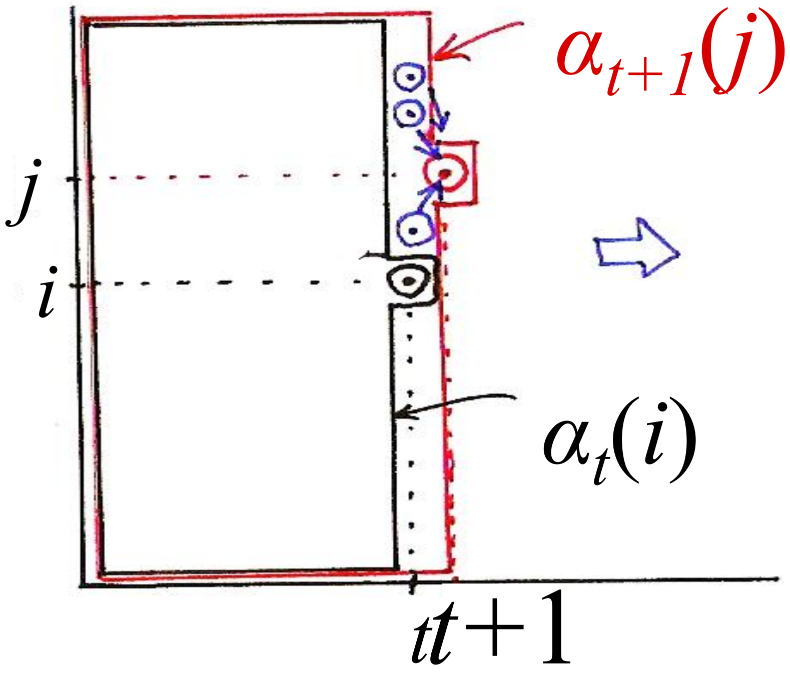
  Totally $N \times M$ number of Gaussian models will be build.

Three Basic Problems in HMM

- Model Evaluation
  - Given the model parameter {A, B} of an acoustic model of a certain phoneme and the input sequence $O = (o_1, o_2, \ldots, o_T)$ find $P(O \mid \text{phoneme})$
- Decoding
  - Given the model parameter {A, B} of an acoustic model of a certain phoneme and the input sequence $O = (o_1, o_2, \ldots, o_T)$ find a best state sequence $q = (q_1, q_2, \ldots, q_T)$ which generate highest $P^*(O \mid \text{phoneme})$ under the current model.
- Parameter Learning
  - Given $O = (o_1, o_2, \ldots, o_T)$ and the corresponding phoneme, find the best values for parameters {A, B} which maximize $P^*(O \mid \text{phoneme})$

Solution of problem 1: Forward Algorithm

- Let $\alpha_t(i)$ denote the probability that in t-th step the state is i, regardless which path it takes in pervious steps



Forward Algorithm

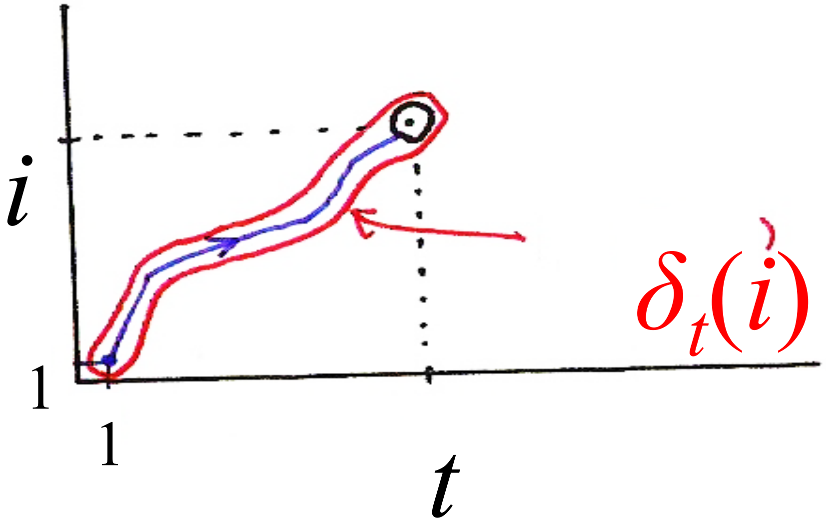
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1})$$

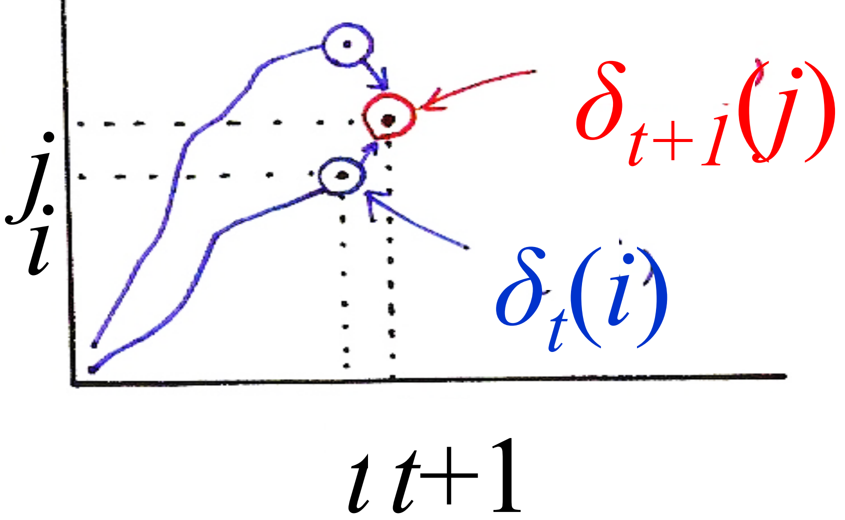
- N is number of distinct HMM state.
- The above function tells us how to efficiently compute $\alpha_{t+1}(j)$
- Fill this table column by column from left to right.
- P(O | phoneme)=$\sum_{i=1}^{N} \alpha_T(i)$ the summation of the probability of all possible solution paths.

Solution of Problem 2: Viterbi Algorithm

- $\delta_t(i)$ the highest probability along a certain single path ending at state i at t-th step

$$\delta_{t+1}(j) = \max_i [\, \delta_t(i)\, a_{ij}\,]\, b_j(O_{t+1})$$

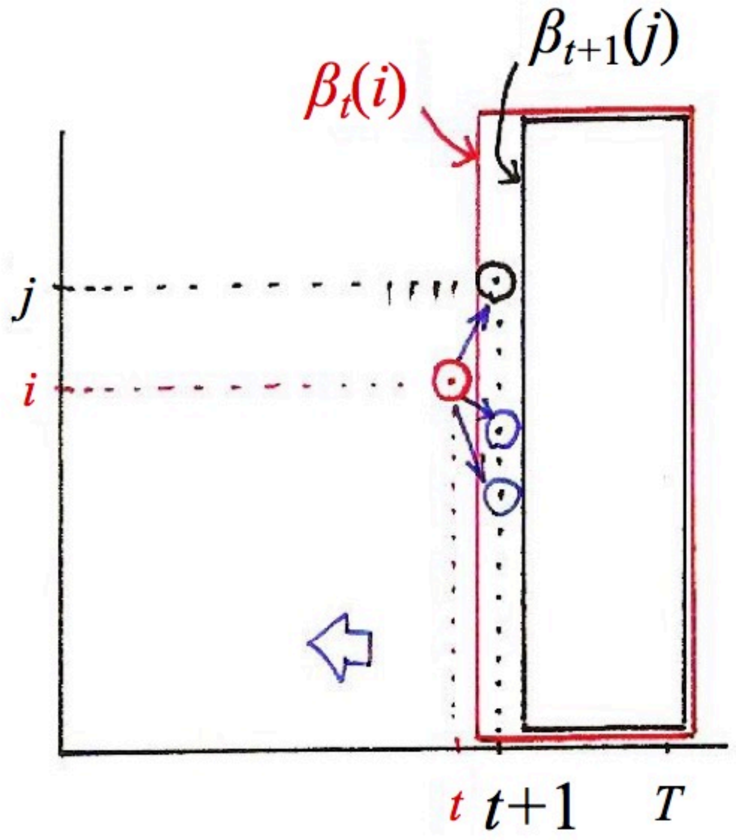




- Note: at each step the algorithm should also record the best state i it selected.
- Fill this table column by column from left to right.
- $P^*(O \mid \text{phoneme}) = \max_i \delta_T(i)$ and q^* can be get through backtracking

Solution of Problem 3: Preparation

- Let $\beta_t(i)$ denote the probability that in t-th step the state is i, regardless which path it takes in remaining steps



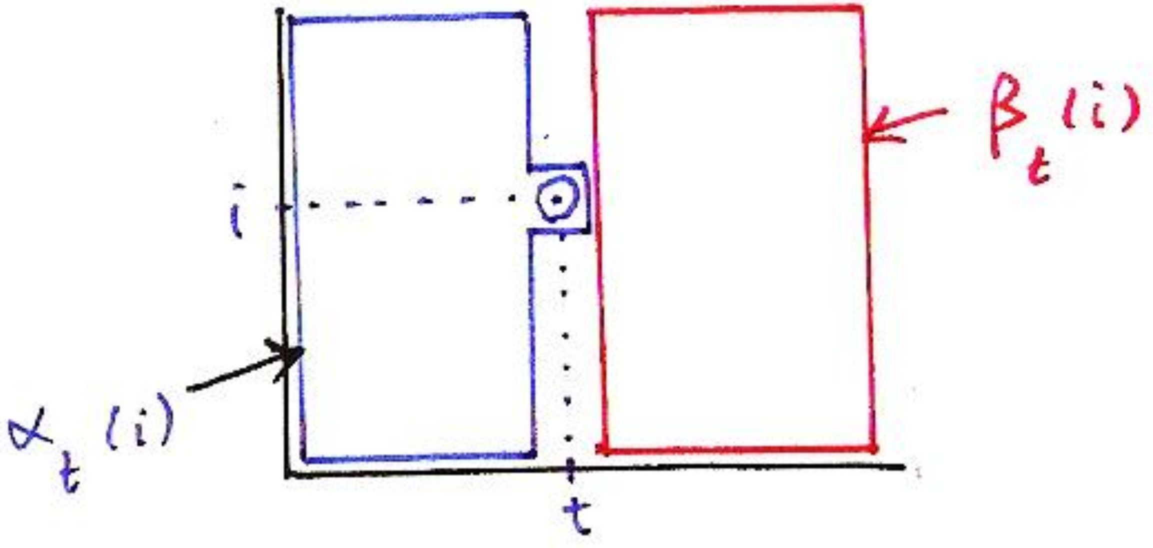
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(O_{t+1})\beta_{t+1}(j)$$

- The above function tells us how to efficiently compute $\beta_t(i)$
- Fill this table column by column from right to left.
- P(O | phoneme)=$\sum_{i=1}^{N} \beta_1(i)$ the summation of the probability of all possible solution paths.
- Initialization: $\beta_T(i) = 1\ \forall\ i = 1 \ldots N$

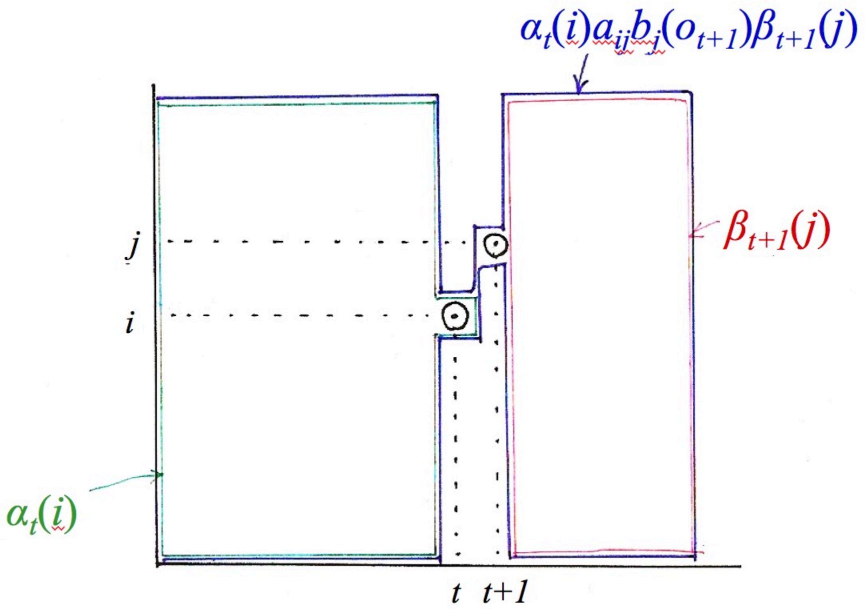
Solution of Problem 3 (cont.)

- $\gamma_t(i) = \frac{P(\bar{O}, q_t = i)}{P(\bar{O})} = \frac{\alpha_t(i)\,\beta_t(i)}{P(\bar{O})}$
  $= P(q_t = i|\bar{O})$

在t时刻对应的state为 i 的概率



- $\varepsilon_t(i,j) = \frac{P(\bar{O}, q_t = i, q_{t+1} = j)}{P(\bar{O})} = \frac{\alpha_t(i)\, a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j)}{P(\bar{O})}$
  $= P(q_t = i, q_{t+1} = j|\bar{O})$



在t时刻对应的state为 i 并且在
t+1时刻对应的state为 j的概率

Solution of Problem 3 (cont.)

- Given $\gamma_t(i)$ and $\varepsilon_t(i,j)$, the state transition probability can be calculated as follows

$$\tilde{a}_{ij} = \frac{\mathbb{E}[P(q_t = i|\bar{O}\ \ )]}{\mathbb{E}[P(q_t = i|\bar{O})]} = \frac{\sum_{t=1}^{T-1} \varepsilon_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

- Using GMM to approximate emission probability

$$b_j(o) = \sum_{k=1}^{M} c_{jk}\, \mathcal{N}\,(o;\ u_{jk}, \Lambda_{jk})$$

Three types of parameters which need to be estimated:

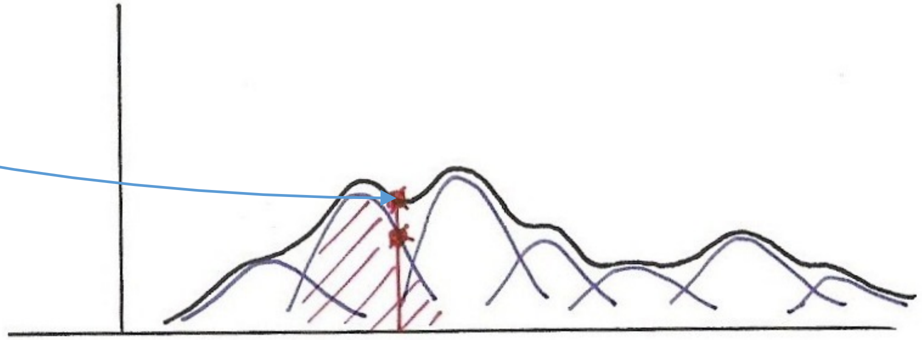
u_{jk}: vector of mean values for the k-th mixture component

Λ_{jk}: covariance matrix for the k-th mixture component

$\sum_{k=1}^{M} c_{jk} \ = 1$: for normalization

Updating parameters for GMM via EM

- $\gamma_t(i, k) = \gamma_t(i) \cdot \frac{c_{jk}\,\mathcal{N}(o; u_{jk}, \Lambda_{jk})}{\sum_{k'=1}^{M} c_{jk'}\,\mathcal{N}(o; u_{jk'}, \Lambda_{jk'})}$ the contribution of k-th mixture component out of the mixture components on $\gamma_t(i)$.



$$\tilde{c}_{ik} = \frac{\sum_{t=1}^{T} \gamma_t(i, \mathrm{k})}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_t(i, \mathrm{k})}$$

$$\tilde{u}_{ik} = \frac{\sum_{t=1}^{T} [\ \gamma_t(i, \mathrm{k}) \cdot o_t]}{\sum_{t=1}^{T} \gamma_t(i, \mathrm{k})}$$

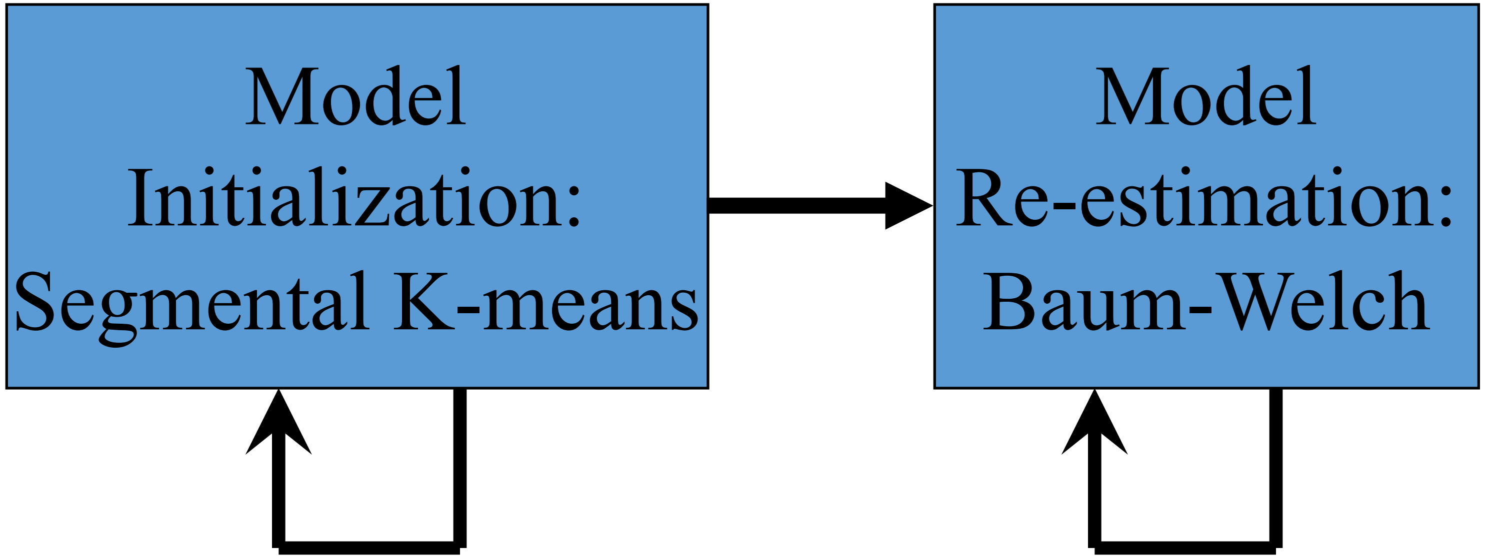
$$\int_{-\infty}^{\infty} x\ \boxed{f_X(x)}\ dx = \bar{x}$$

$$\tilde{\Lambda}_{ik} = \frac{\sum_{t=1}^{T} [\ \gamma_t(i, \mathrm{k})\ \boxed{(o_t - u_{ik})(o_t - u_{ik})^T}]}{\sum_{t=1}^{T} \gamma_t(i, \mathrm{k})}$$

$$\int_{-\infty}^{\infty} \boxed{(x - \bar{x})^2}\ f_X(x)\ dx = \sigma_x^2$$

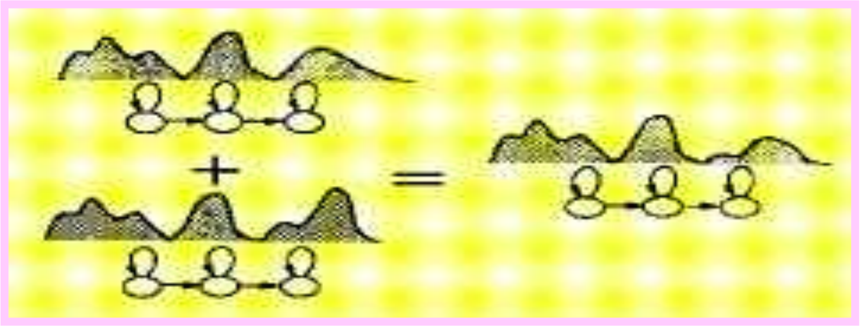
Training framework for HMM+GMM

- No closed-form solution, but approximated iteratively
- An initial model is needed-model initialization
- May converge to local optimal points rather than global optimal point
  - heavily depending on the initialization
- Model training



Triphone is still too large, so we need Sharing of Parameters and Training Data for Triphones

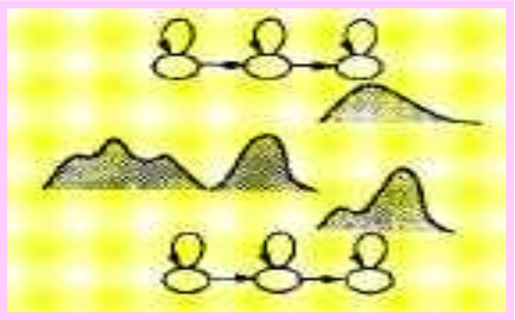
- **Sharing at Model Level**



Generalized Triphone

– clustering similar triphones and merging them together

- **Sharing at State Level**

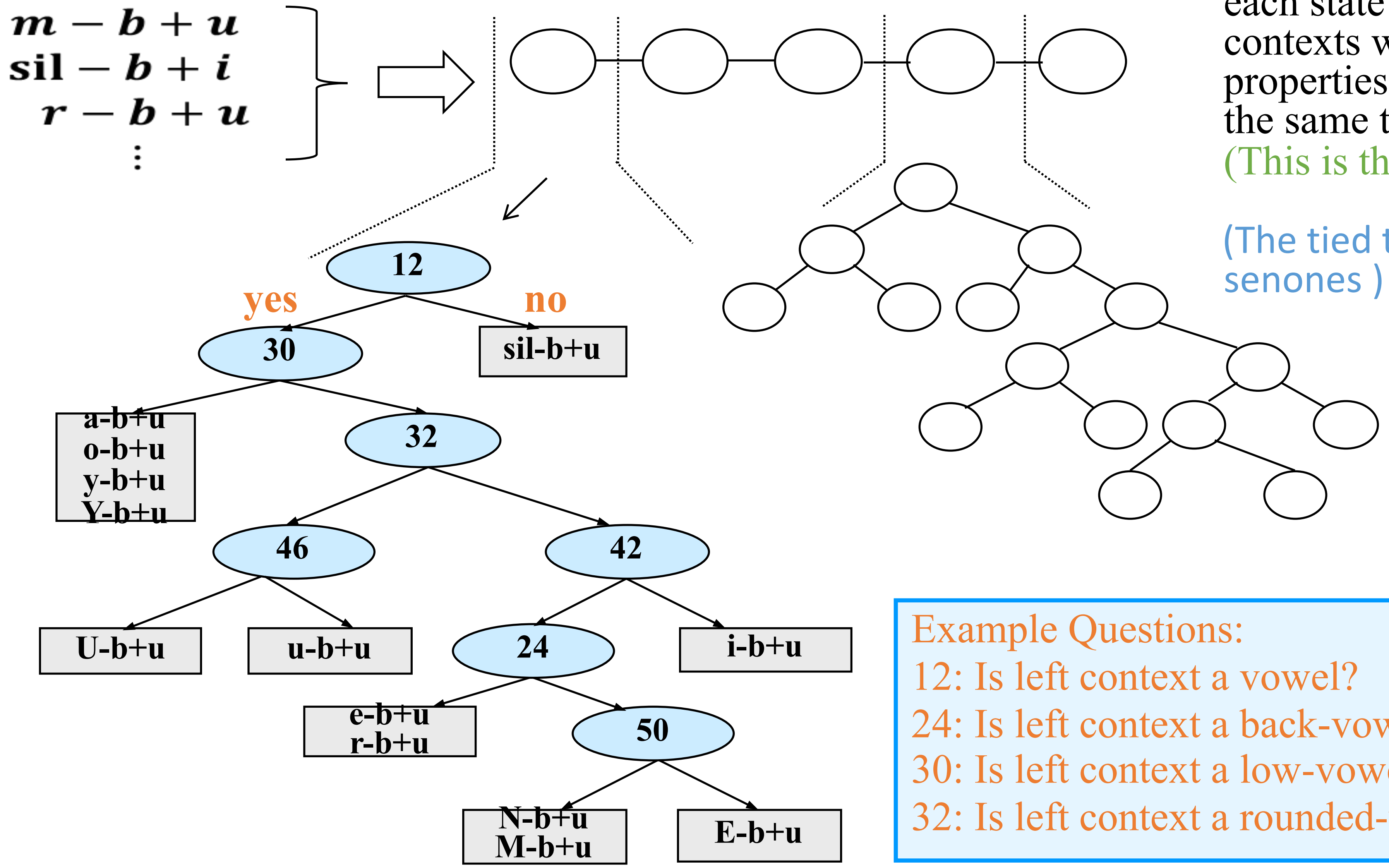


Shared Distribution Model (SDM)

– those states with quite different distributions do not have to be merged

Training Triphone Models with Decision Trees, use tree to group data

- **An Example: “(_ –) b (+_)”**



The Gaussion mixture distribution for each state of a phoneme model for contexts with similar linguistic properties are “tied” together, sharing the same training data and parameters. (This is the goal of using tree)

(The tied triphone states is named as senones)

Example Questions:
12: Is left context a vowel?
24: Is left context a back-vowel?
30: Is left context a low-vowel?
32: Is left context a rounded-vowel?

Advantage of HMM

- Handle the output dependency.
- Do not need to align the input with the out put at each time point in advance. 说话有快有慢，发言有长有短. HMM is able to deal with this via introducing hidden state, and each state corresponds to a sequence of observation with various length.
- HMM model can be easily concatenated, therefore, the acoustic model for the word can be built via connect a serious of acoustic models for phonemes. Similarly, the acoustic model for the sentence can be built via connect a serious of acoustic models for words.

Drawback of HMM+GMM

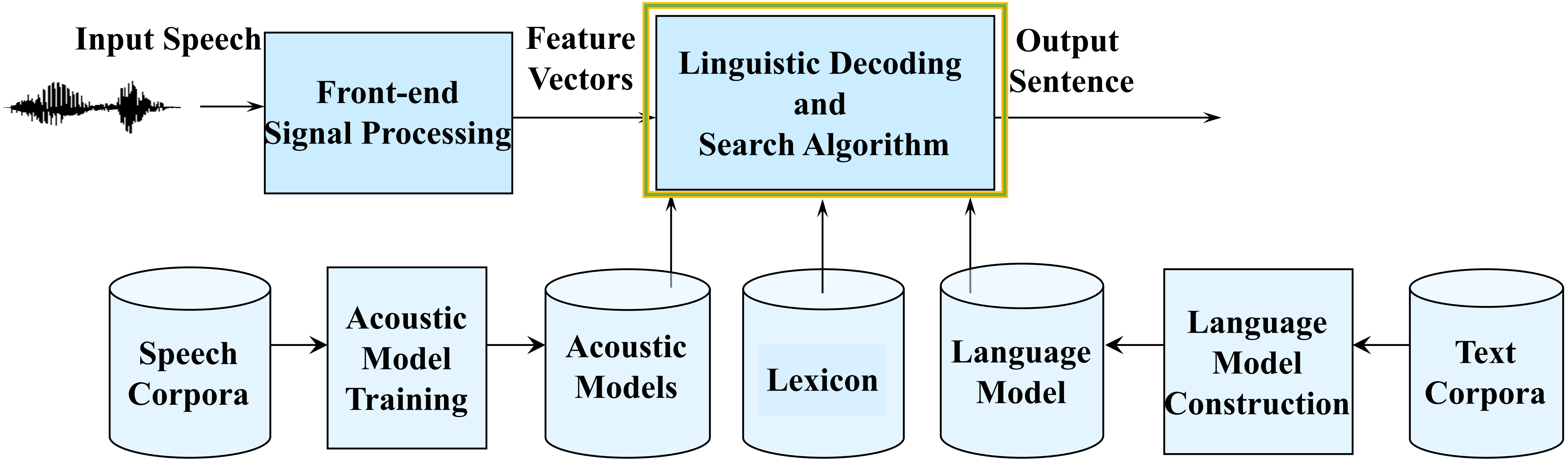
- This is a generative based model. The model is trained with only positive samples, and may only focus on the characteristic of the objective. Therefore, it is noisy sensitive.
  - Eg. To train a acoustic model for "一" in Chinese, a lot of acoustic data of "一" will be collected. And the model will be trained accordingly. However, "七" in some case is also sounds like "一". The acoustic might only figure out the characteristic of each word, but can not distinguish them very well.

We need Discriminative training for ASR

Heigold, Georg, et al. "Discriminative training for automatic speech recognition: Modeling, criteria, optimization, implementation, and performance." *IEEE Signal Processing Magazine* 29.6 (2012): 58-69.

How to add Discriminative training?

1. The label is the word sentence of the corresponding speech.
2. Using HMM+GMM to build acoustic models for each target training unit.
3. Taking the above acoustic models as initialization and combined with the language model to jointly update both acoustic models and language model.

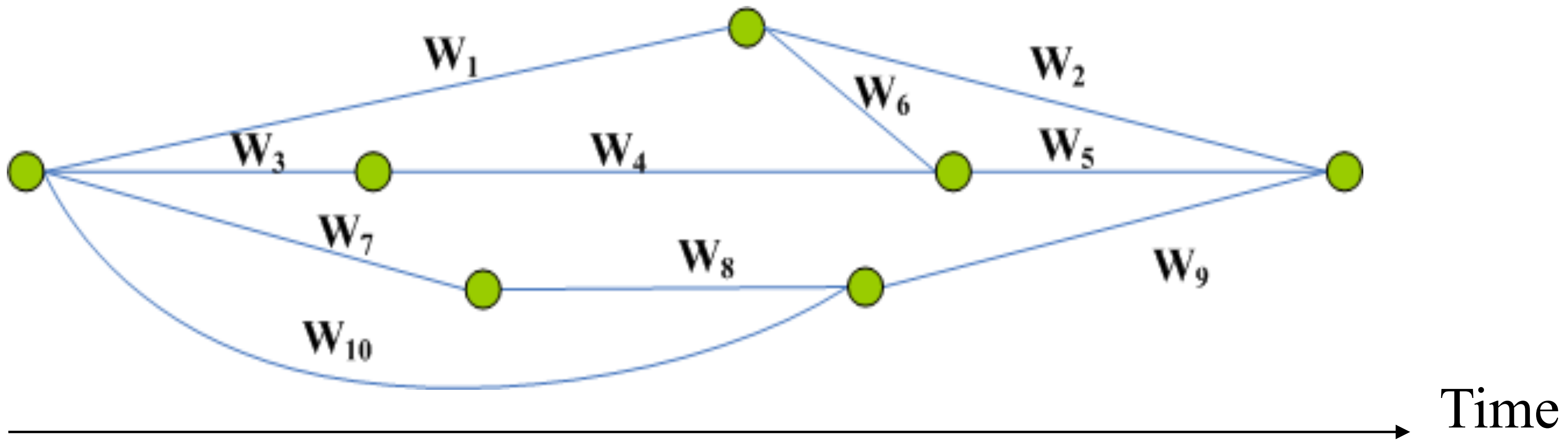


Discriminative Training for ASR

- **Minimum Bayesian Risk (MBR)**
  - $(\Lambda, \Gamma) = \arg\min_{\Lambda,\Gamma} \sum_r R(W_r^* | O_r)$ adjusting all model parameters to minimize the Bayesian Risk
    - Λ: $\{\lambda_i, i=1,2,\ldots\ldots N\}$ acoustic models
    - Γ: Language model parameters
    - O_r : r-th training utterance
    - W_r^*: correct transcription of O_r
  - $R(W_r^* | O_r) = \sum_u P_{\Lambda,\Gamma}(u | O_r) L(u, W_r^*)$ Bayesian Risk
    - u: a possible recognition output found in the lattice
    - $L(u,W_r)$: Loss function
    - $P_{\Lambda,\Gamma}(u | O_r)$: posteriori probability of *u* given O_r based on Λ,Γ
    - $L(u, W_r^*) = \begin{cases} 0, u = W_r^* \\ 1, u \neq W_r^* \end{cases}$ → *MAP principle*
    - Other definitions of $L(u, W_r^*)$ possible

Discriminative Training for ASR (cont.)

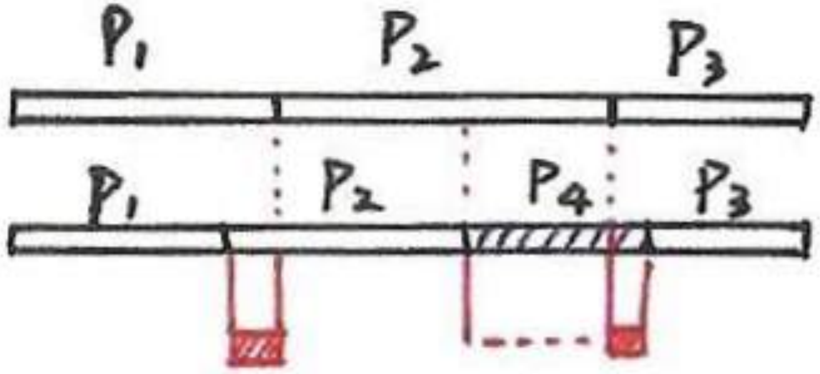
- Lattice is u, the correct answer W_r^* is just one path of lattice



Minimum Phone Error Rate (MPE) Training

- $(\Lambda, \Gamma) = \arg\min_{\Lambda,\Gamma} \sum_r \sum_u P_{\Lambda,\Gamma}(u|O_r)\, Acc(u, W_r^*)$
  - $Acc(u, W_r^*)$: Phone Accuracy

Phone Accuracy



Reference phone sequence

Decoded phone sequence for a path in the lattice

Povey, Daniel, and Philip C. Woodland. "Minimum phone error and I-smoothing for improved discriminative training." *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*. Vol. 1. IEEE, 2002.

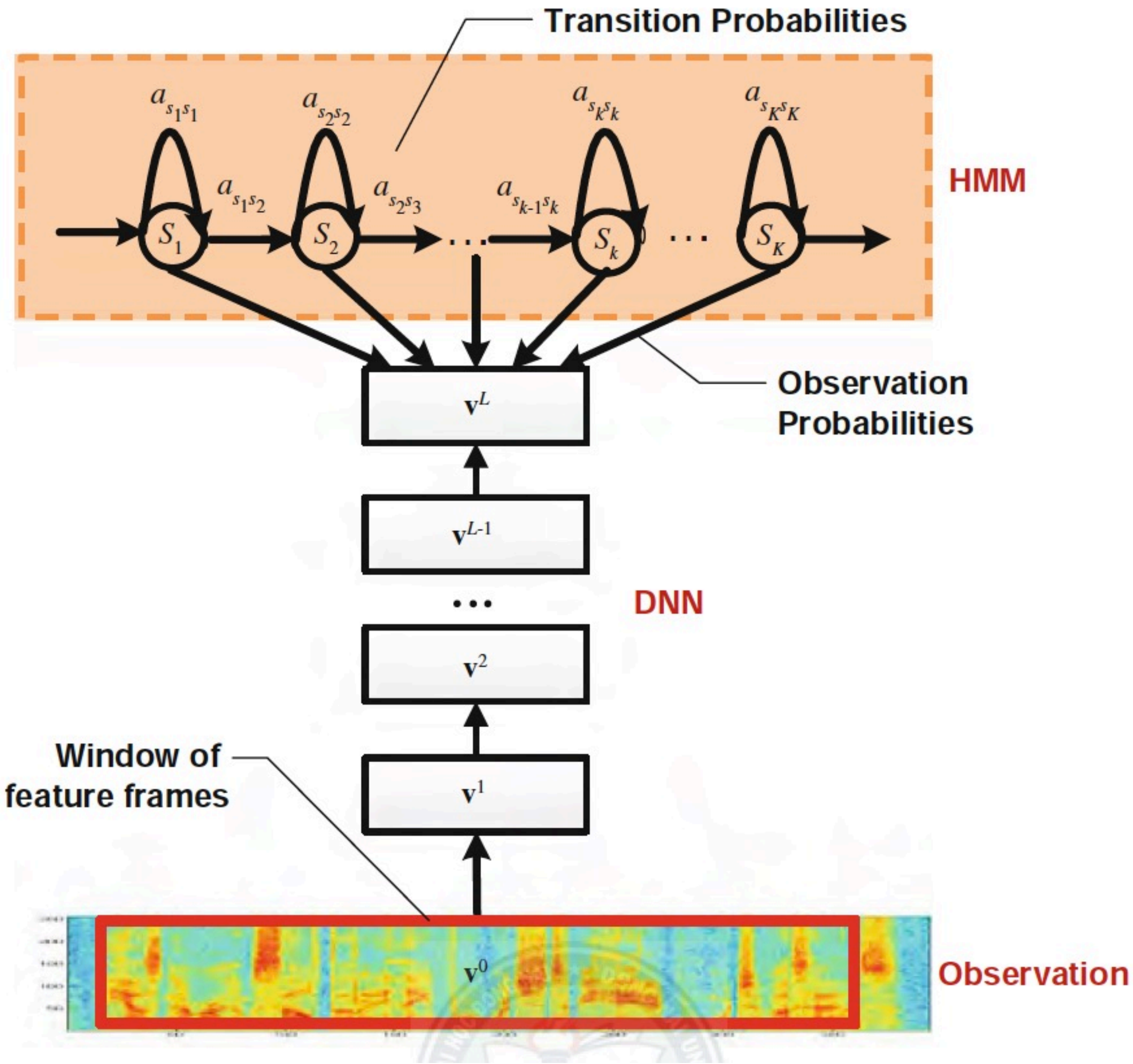
Outline

- Overview of ASR system
- Hidden Markov Models (HMM)
- HMM + DNN
- Connectionist temporal classification

HMM+DNN

- The HMM+DNN hybrid system takes advantage of DNN’s strong representation learning power and HMM’s sequential modeling ability, and outperforms conventional Gaussian mixture model HMM+GMM systems significantly on many large vocabulary continuous speech recognition tasks.
- The HMM+DNN can conduct discriminate training when building acoustic model.
- In the HMM+DNN, a single DNN is trained to estimate the conditional state posterior probability for all state, while in HMM+GMM a different GMM is used to model each different state.

Architecture of HMM+DNN



- Dynamics of the speech signal is modeled with HMMs
- The observation probabilities are estimated through DNNs
- Each output neuron of the DNN is trained to estimate the posterior probability of continuous density HMMs' state given the acoustic observations.

Algorithm 6.1 Main steps involved in training CD-DNN-HMMs

procedure TRAINCD- DNN- HMM($\mathbb{S}$) ▹ $\mathbb{S}$ is the training set

  $hmm0 \leftarrow$ TrainCD-GMM-HMM($\mathbb{S}$); ▹ $hmm0$ is used in the GMM system

  $stateAlignment \leftarrow$ ForcedAlignmentWithGMMHMM($\mathbb{S}$, $hmm0$); 用 Viterbi 找出输入与HMM state的对应

  $stateToSenoneIDMap \leftarrow$ GenerateStateTosenoneIDMap($hmm0$);

  $featureSenoneIDPairs \leftarrow$ GenerateDNNTrainingSet($stateToSenoneIDMap$, $stateAlignment$);

  $ptdnn \leftarrow$ PretrainDNN($\mathbb{S}$); ▹ Optional

  $hmm \leftarrow$ ConvertGMMHMMToDNNHMM($hmm0$, $stateToSenoneIDMap$); ▹ hmm is used in the DNN system

  $prior \leftarrow$ EstimatePriorProbability($featureSenoneIDPairs$)

  $dnn \leftarrow$ Backpropagate($ptdnn$, $featureSenoneIDPairs$);

  Return $dnnhmm = \{dnn, hmm, prior\}$

end procedure

Convert *gmm-hmm* to the corresponding CD-DNN-HMM denoted as *hmm* by borrowing the tri-phone and senone structure as well as the transition probabilities from *gmm-hmm*

HMM+DNN Disadvantage

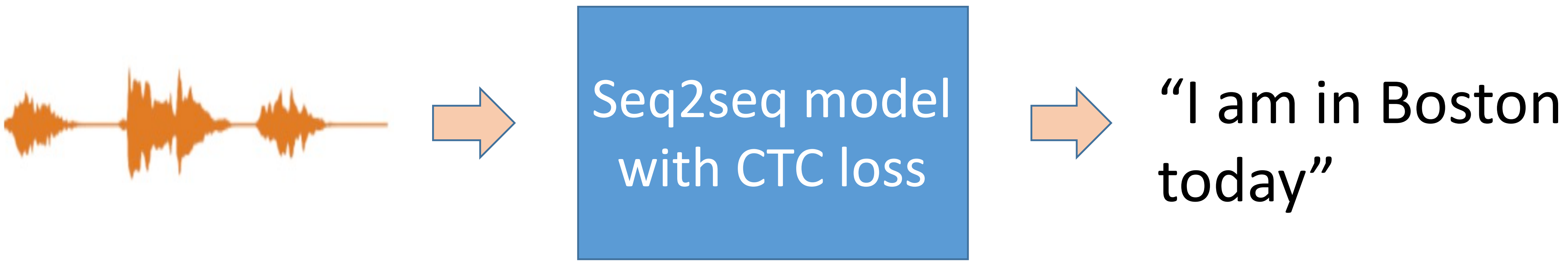
- The label of the DNN (the corresponding HMM state of each observation) is generated via a HMM+GMM model, and the label quality can affect the performance of the DNN system

Outline

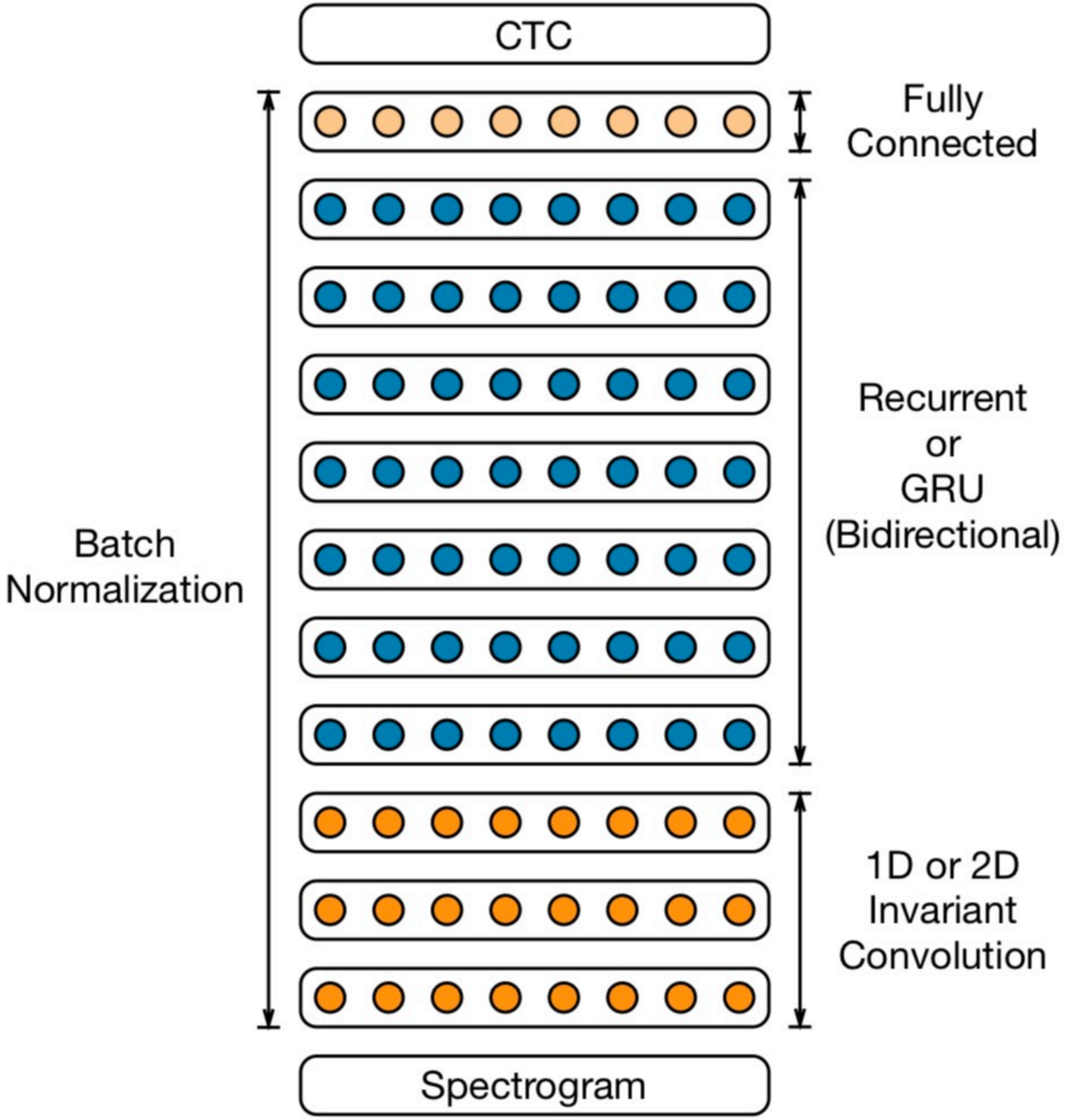
- Overview of ASR system
- Hidden Markov Models (HMM)
- HMM + Artificial Neural Network
- Connectionist temporal classification

End-to-End ASR

- ASR is a sequence-to-sequence learning problem
- A simpler paradigm with a single model (and training stage)



Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on* (pp. 6645-6649). IEEE.



Architecture of the DS2 system used to train on both English and Mandarin speech. The authors explore variants of this architecture by varying the number of convolutional layers from 1 to 3 and the number of recurrent or GRU layers from 1 to 7 .

Amodei, Dario, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper et al. "Deep speech 2: End-to-end speech recognition in english and mandarin." In *International Conference on Machine Learning*, pp. 173-182. 2016.

Connectionist Temporal Classification

- CTC is a sequence-to-sequence learning technique

$$L_{CTC} = -log\,P(\mathrm{W}*|O)$$

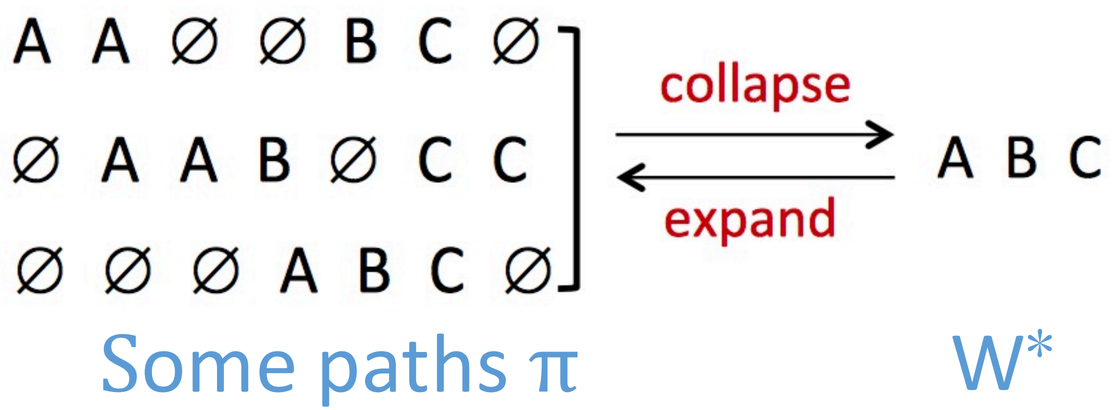
  - The target sequence W* = (w_1,w_2,…,w_U) is at most as long as the input sequence O = (o_1,o_2,…,o_T), i.e. |W| = U ≤ |O| = T.
- CTC paths bridge frame-level labels with the label sequence
  - A CTC path π is a sequence of labels on the frame level, the π has same length as input sequence.
  - The likelihood of a CTC path is decomposed onto the frames:

$$p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y^t_{\pi_t},\ \forall \pi \in L'^T.$$

  - $y^t_{\pi_t}$ is the probability of observing label π_t at time t, is the output of the RNN network. $L' = L \cup \{\emptyset\}$

CTC Paths

- CTC paths differ from labels sequences in that:
  - Add the blank as an additional label, meaning no (actual) labels are emitted
  - Allow repetitions of non-blank labels



- Many-to-one mapping from CTC paths **Φ**(W*) to the label sequence W*

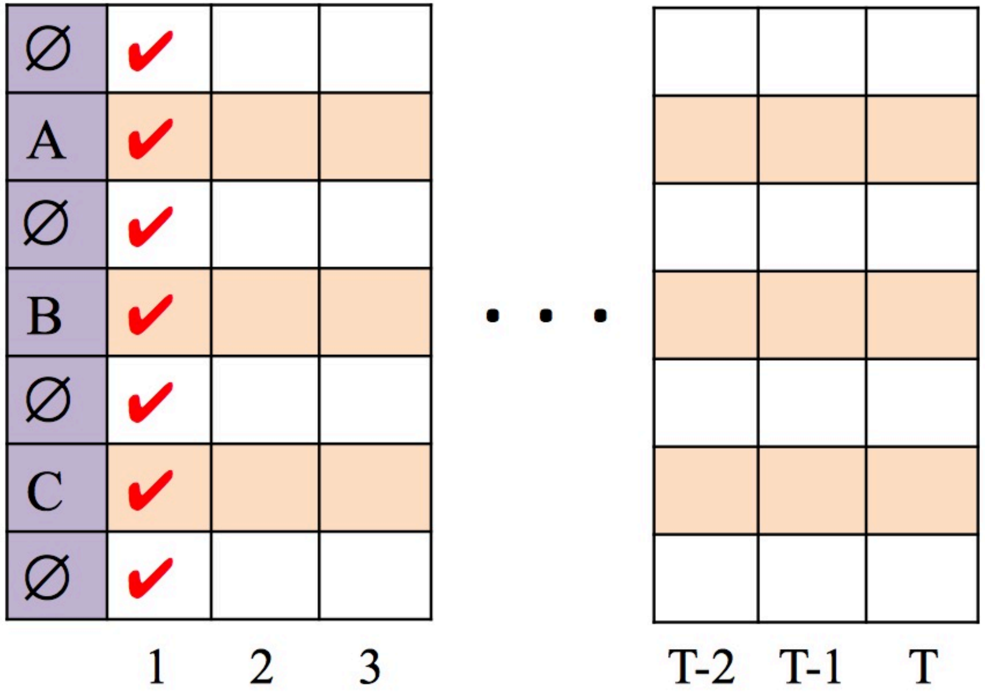
$$P(W^*|O) = \sum_{\pi \in \Phi(W^*)} P(\pi|O) = \sum_{\pi \in \Phi(W^*)} \sum_{t=1}^{T} y_{\pi_t}^{t}$$

Computationally Intractable !!

Forward-Backward Algorithm

$$ABC \rightarrow \emptyset A \emptyset B \emptyset C \emptyset$$

expanded labels
l

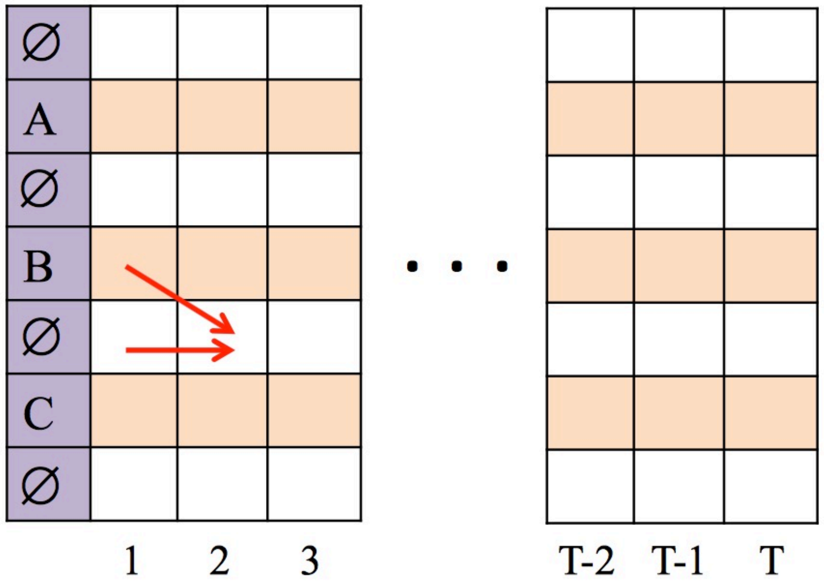


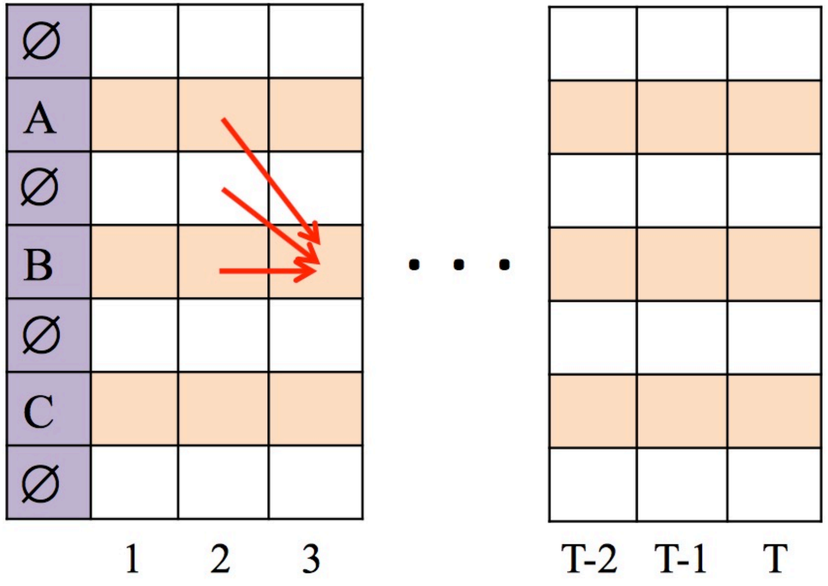
$\alpha_t(s)$

summed probability of all the CTC paths ending at t with s

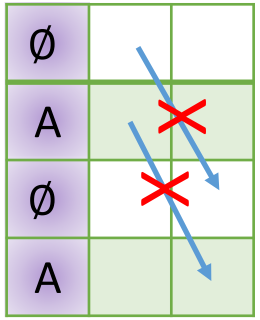
$$\alpha_1(\varnothing) = y_{\varnothing}^{1} \qquad \alpha_1(A) = y_{A}^{1} \qquad \alpha_1(s) = 0,\ \forall s > 2$$

Forward Computation



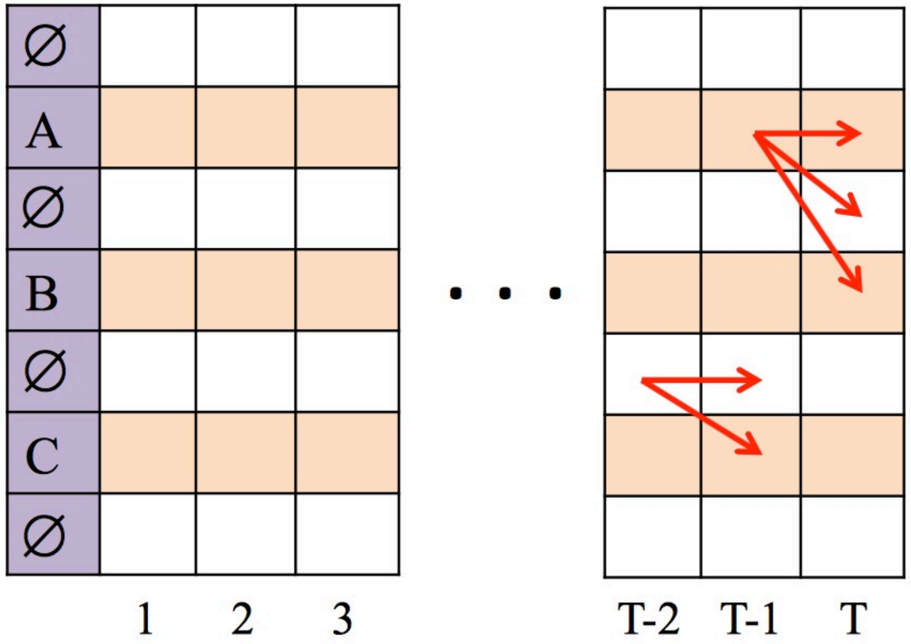


$$\alpha_t(s) = \begin{cases} y^t_{l_s}\left(\alpha_{t-1}(s) + \alpha_{t-1}(s-1)\right) & \text{if } l_s = \varnothing \text{ or } l_s = l_{s-2} \\ y^t_{l_s}\left(\alpha_{t-1}(s) + \alpha_{t-1}(s-1) + \alpha_{t-1}(s-2)\right) & \text{otherwise} \end{cases}$$



Backward Computation

expanded labels
l



$\beta_t(s)$
summed probability of all CTC paths starting at t with s

$$\beta_T(\varnothing) = y_{\varnothing}^{T} \qquad \beta_T(C) = y_C^T \qquad \beta_T(s) = 0,\ \forall s < |l| - 1$$

$$\beta_t(s) = \begin{cases} y_{l_s}^{t}\left(\beta_{t+1}(s) + \beta_{t+1}(s+1)\right) & \text{if } l_s = \varnothing \text{ or } l_s = l_{s+2} \\ y_{l_s}^{t}\left(\beta_{t+1}(s) + \beta_{t+1}(s+1) + \beta_{t+1}(s+2)\right) & \text{otherwise} \end{cases}$$

CTC Training

- The CTC loss function can be effectively calculated via the forward-backward algorithm

$$L_{CTC} = -logP(W^*|O) = -log\sum_{s=1}^{|l|}\alpha_t(s)\beta_t(s)$$

- The gradient of L_{CTC} with respect to the network output y_k^t:

$$\frac{\partial L_{CTC}}{\partial y_k^t} = -\frac{1}{P(W^*|O)}\frac{\partial P(W^*|O)}{\partial y_k^t}$$

$$\frac{\partial P(W^*|O)}{\partial y_k^t} = \frac{1}{y_k^t}\sum_{s\in\Phi(W^*,k)}\alpha_t(s)\beta_t(s)$$

$\Phi(W^*,k) = \{\mathrm{s}, l_s = k\}$ the set of positions where label k occurs in l.

Advantage of CTC

- End to end learning, not like the aforementioned two types of models.
- The deep learning methods is involved automatically in CTC

Reference

- Yu, Dong, and Li Deng. *AUTOMATIC SPEECH RECOGNITION*. SPRINGER LONDON Limited, 2016.
- Heigold, Georg, et al. "Discriminative training for automatic speech recognition: Modeling, criteria, optimization, implementation, and performance." *IEEE Signal Processing Magazine* 29.6 (2012): 58-69
- Povey, Daniel, and Philip C. Woodland. "Minimum phone error and I-smoothing for improved discriminative training." *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*. Vol. 1. IEEE, 2002.
- Dahl, George E., Dong Yu, Li Deng, and Alex Acero. "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition." *IEEE Transactions on audio, speech, and language processing* 20, no. 1 (2012): 30-42.
- Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on* (pp. 6645-6649). IEEE.
- Amodei, Dario, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper et al. "Deep speech 2: End-to-end speech recognition in english and mandarin." In *International Conference on Machine Learning*, pp. 173-182. 2016.