專題研究 Week 2 Introduction

Prof. Lin-Shan Lee
TA: Chung-Ming Chien

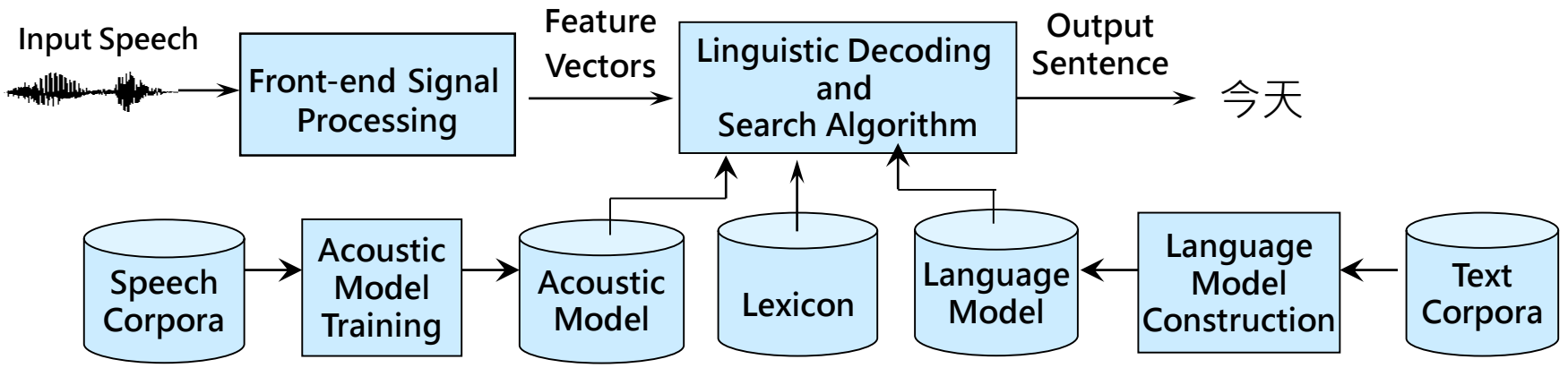
Outline

1. Recap
2. Acoustic modeling
3. Homework

Recap

語音辨識系統

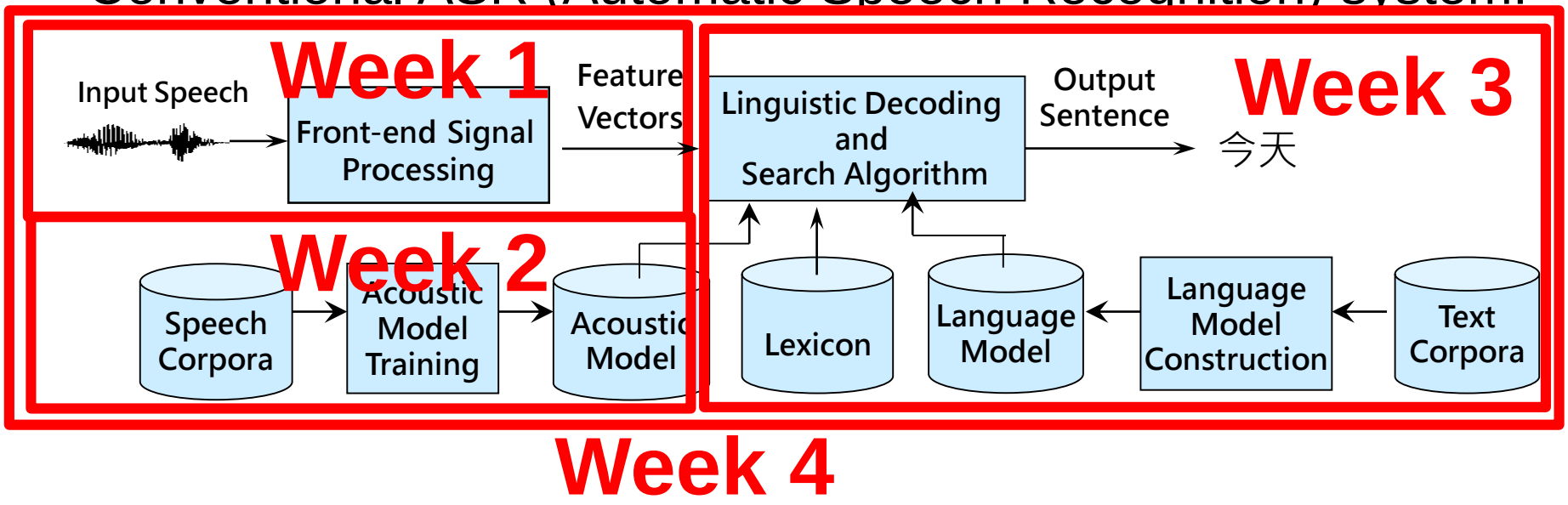
- Conventional ASR (Automatic Speech Recognition) system:



- Deep learning based ASR system

語音辨識系統

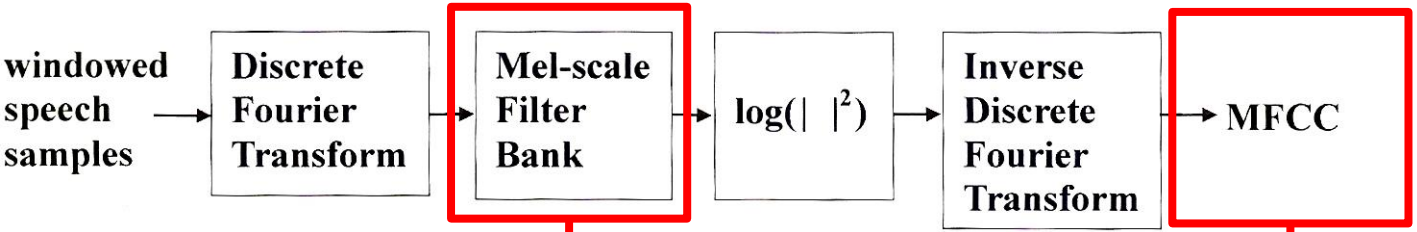
- Conventional ASR (Automatic Speech Recognition) system:

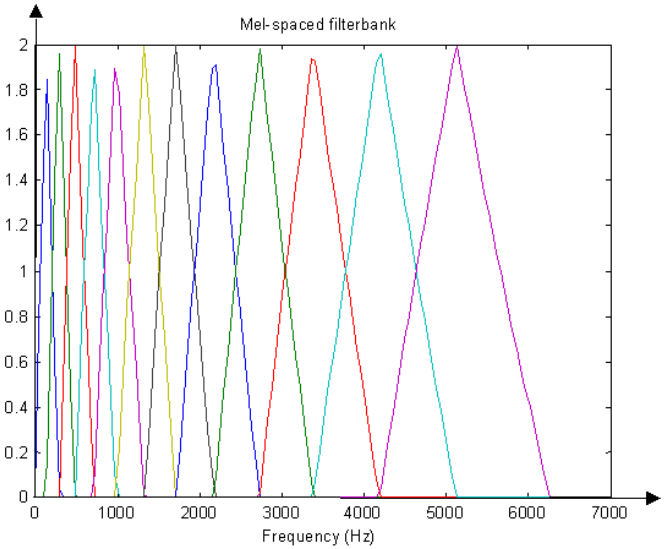


- Deep learning based ASR system

Week 5

MFCC (Mel-frequency cepstral coefficients)





13 dimensions vector

Extract Feature (02.extract.feat.sh)

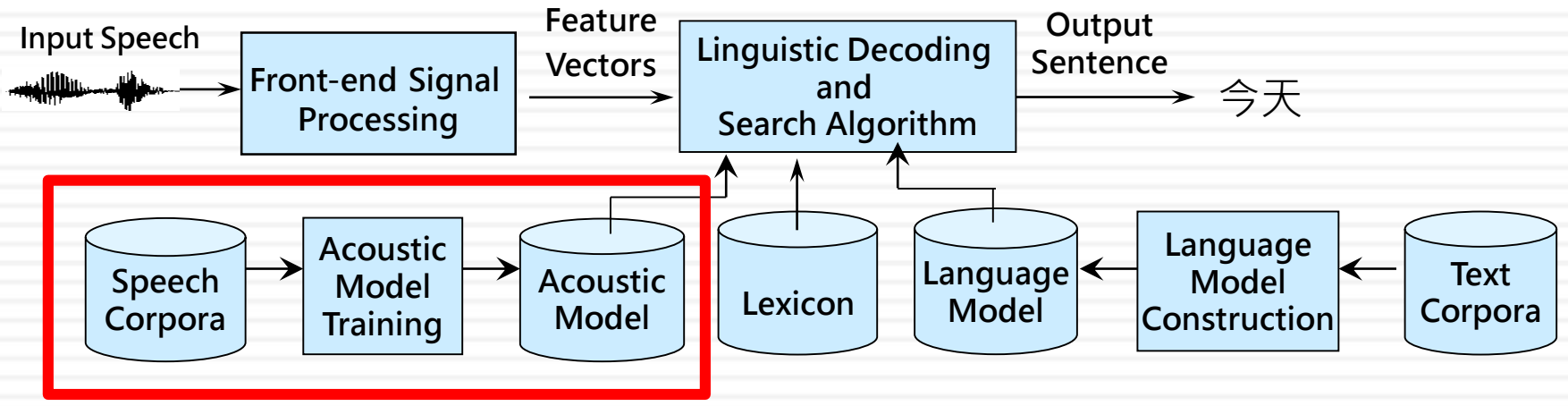
- compute-mfcc-feats
- add-deltas
- compute-cmvn-stats
- apply-cmvn

Acoustic Modeling

03.mono.train.sh

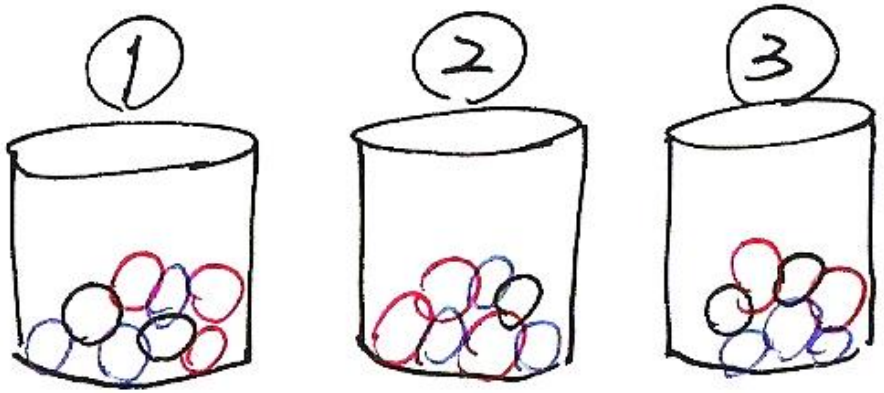
05.tree.build.sh

06.tri.train.sh



Hidden Markov Model(HMM)

- Given
  - Sequence of observations(balls)
  - Hidden Markov model (transition between the baskets and observation probability for each basket)

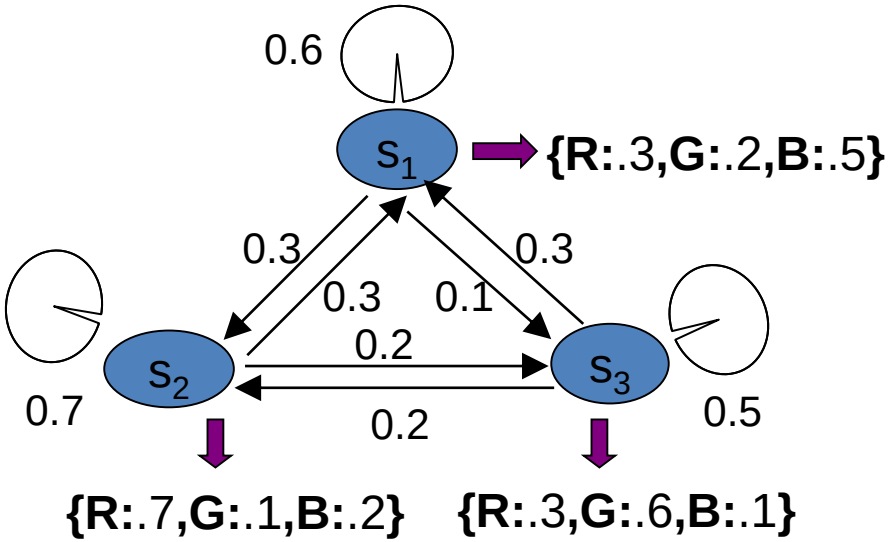


RGBGGBBGRRR.....

- Expected
  - Sequence of states(baskets)

Hidden Markov Model(HMM)

- Elements of an HMM {S,A,B,π}
  - S is a set of *N* states
  - A is the *N* x *N* matrix of state transition probabilities
  - B is a set of *N* probability functions, each describing the observation probability with respect to a state
  - π is the vector of initial state probabilities

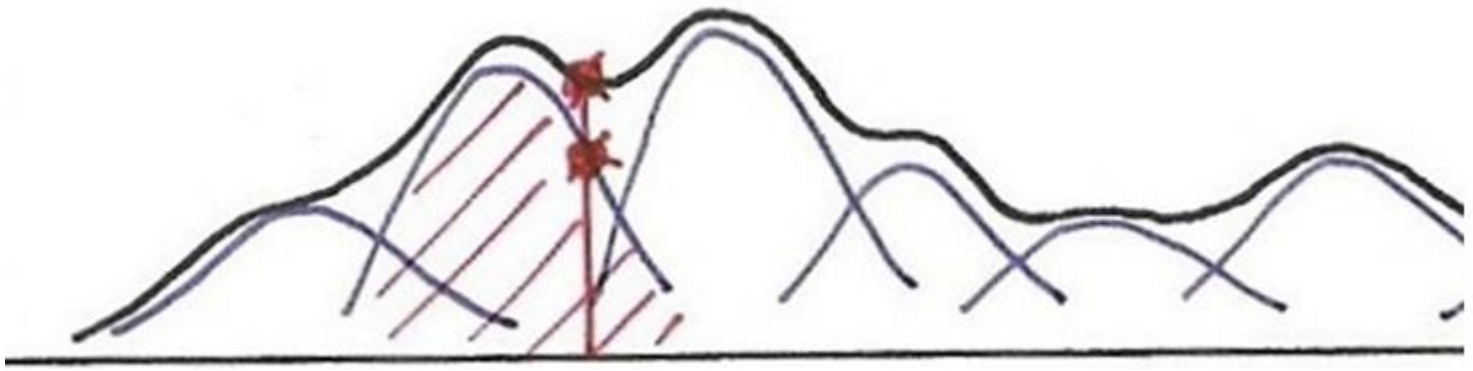


$$\mathbf{A} = \begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 0.1 & 0.7 & 0.2 \\ 0.3 & 0.2 & 0.5 \end{bmatrix}$$

$$\pi = \begin{bmatrix} 0.4 & 0.5 & 0.1 \end{bmatrix}$$

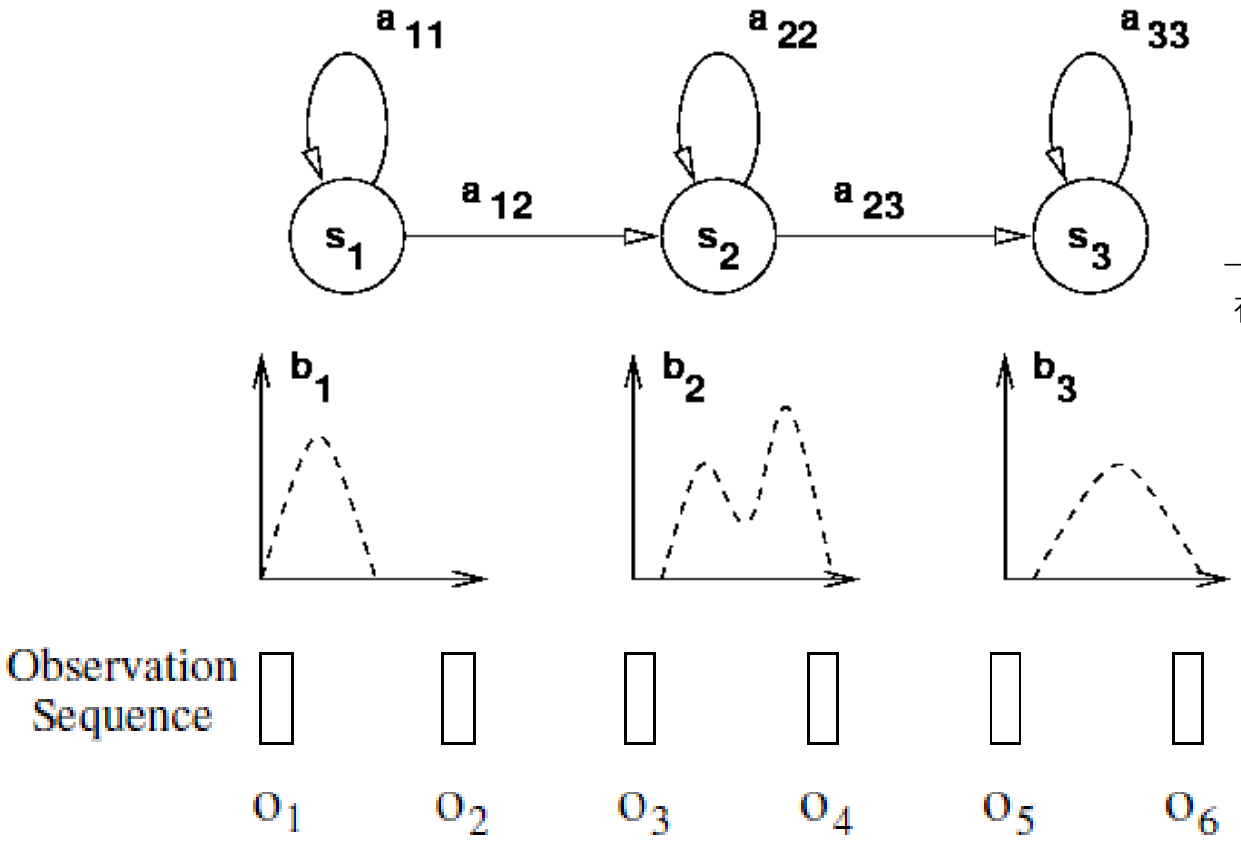
Gaussian Mixture Model(GMM)

- Observation may be continuous. (e.g., mfcc)
- Use GMM to model continuous prob. density function.



Acoustic Model: P(O | λ)

- Model of a phone



Markov Model

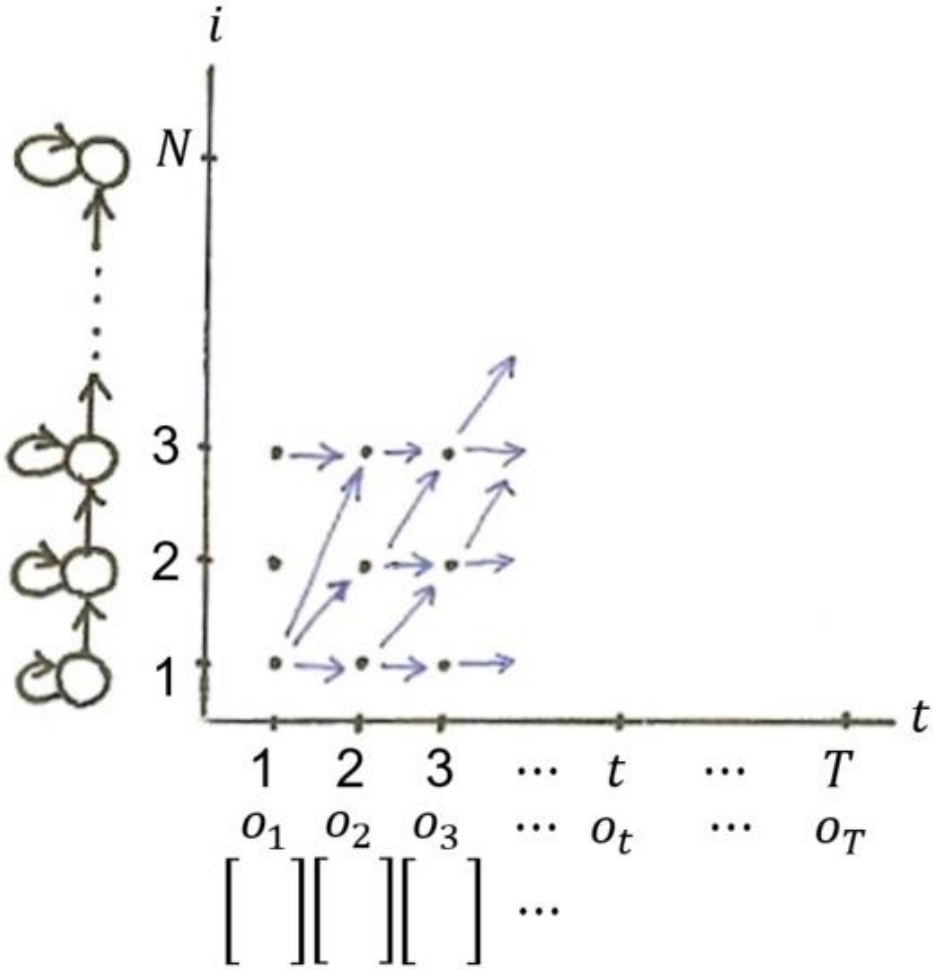
一般的HMM不必有方向性，但用在Acoustic model的都是單向的

Gaussian Mixture Model

Acoustic Model: P(O | λ)

- Forward algorithm (4.0 - basic problem 1)
  - $\alpha_t(i) = P(o_1 o_2 \ldots\ldots o_t , q_t = i|\lambda)$
  - $= P(\text{observing } o_1 o_2 \ldots o_t , \text{ state i at time t}|\lambda)$
  - $\alpha_1(i) = \pi_i b_i(o_1) , \qquad 1 \le i \le N$
  - $\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)\ a_{ij}]\ b_j(o_{t+1})$
  - $P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

Acoustic Model: P(O | λ)



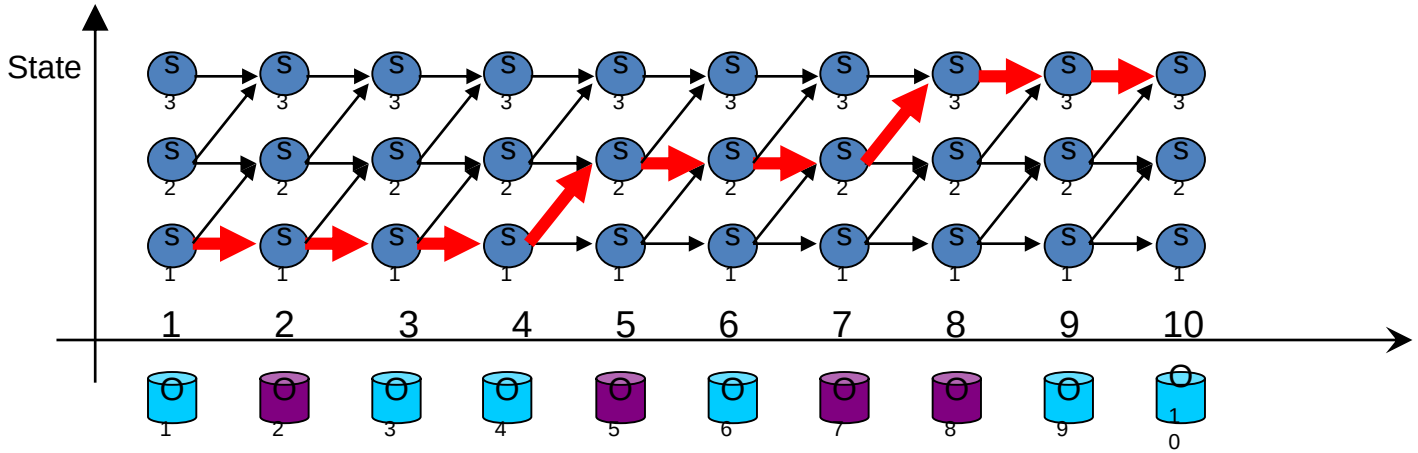
Acoustic model: Best State Seq.

- Viterbi algorithm (4.0 - basic problem 2)
  - Goal: find the optimal state sequences $\boldsymbol{q} = q_1 q_2 \ldots q_T$
  - P(O|λ) can also be computed based on $\boldsymbol{q}$.

Acoustic model: Training

- Baum-Welch algorithm (4.0 - basic problem 3)
  - Goal: Adjust parameters (π, A, B) to maximize P(O| λ)

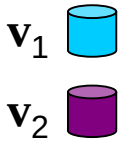
Acoustic model: Training



$b_1(\mathbf{v}_1)=3/4,\ b_1(\mathbf{v}_2)=1/4$

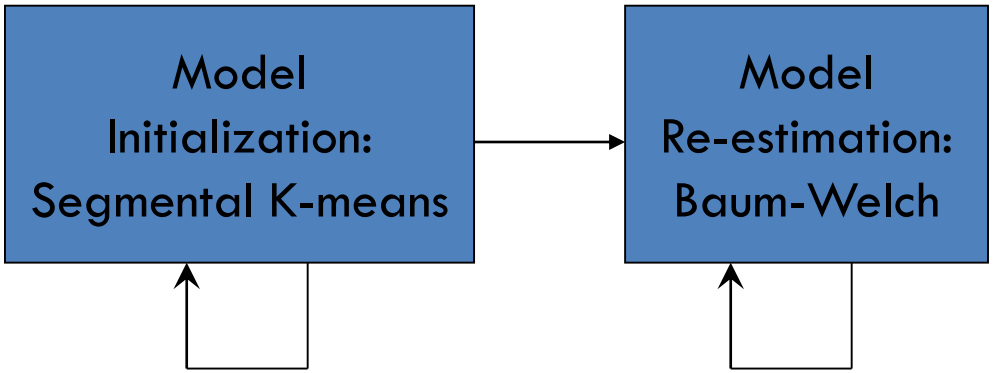
$b_2(\mathbf{v}_1)=1/3,\ b_2(\mathbf{v}_2)=2/3$

$b_3(\mathbf{v}_1)=2/3,\ b_3(\mathbf{v}_2)=1/3$



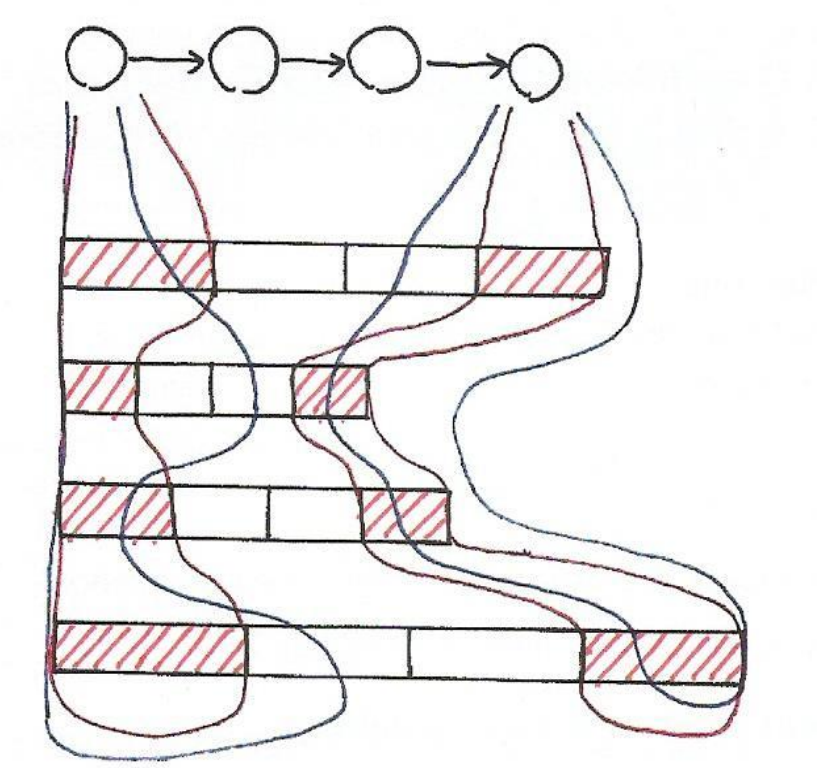
Acoustic model: Training

- Initialization
  - Bad initialization leads to local minimum with higher probability.



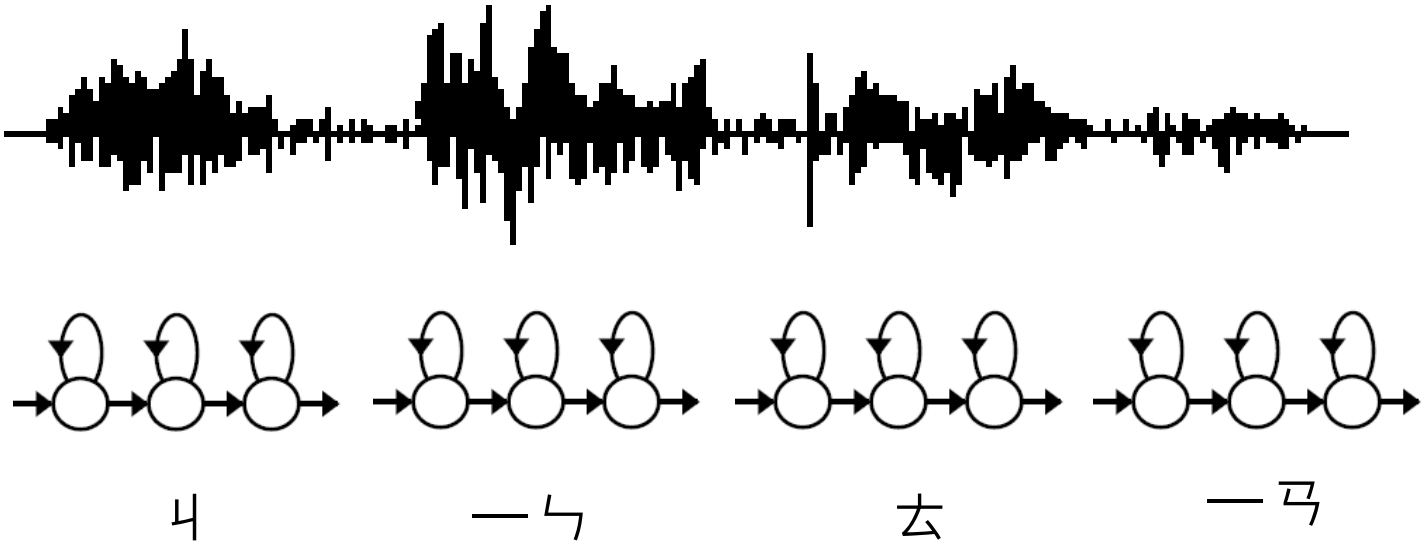
Acoustic model: Training

- 假設有四個人同時發出「ㄅ」這個音



Acoustic Model: P(O|W)

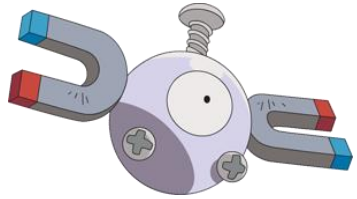
- One acoustic model for a phoneme?



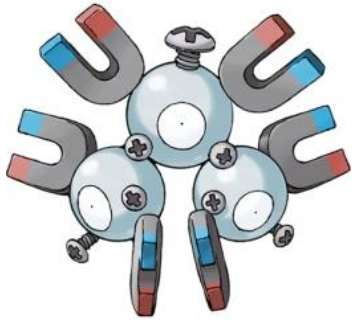
- The Pronounce of a phoneme may be affected by its neighbors!

Monophone v.s. Triphone

- Monophone
  - Consider only one phone information per model
  - Ex. ㄧ, ㄨ, ㄩ



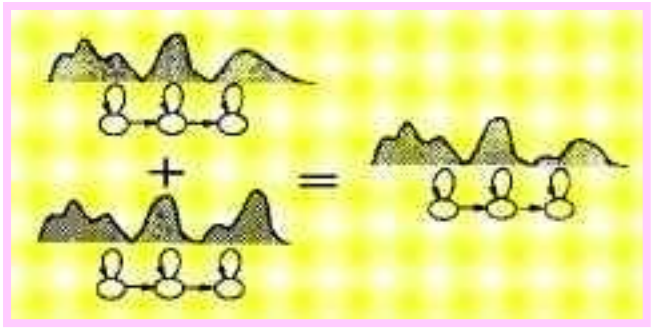
- Triphone
  - Consider both left and right neighboring phones
    $(60)^3 \rightarrow$ 216,000
  - Ex. ㄇ+ㄧ+ㄠ



Triphone

- Too much (216000) model to train?
- Share!

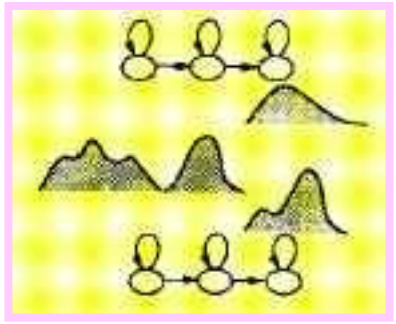
• **Sharing at Model Level**



Generalized Triphone

• **Sharing at State Level**

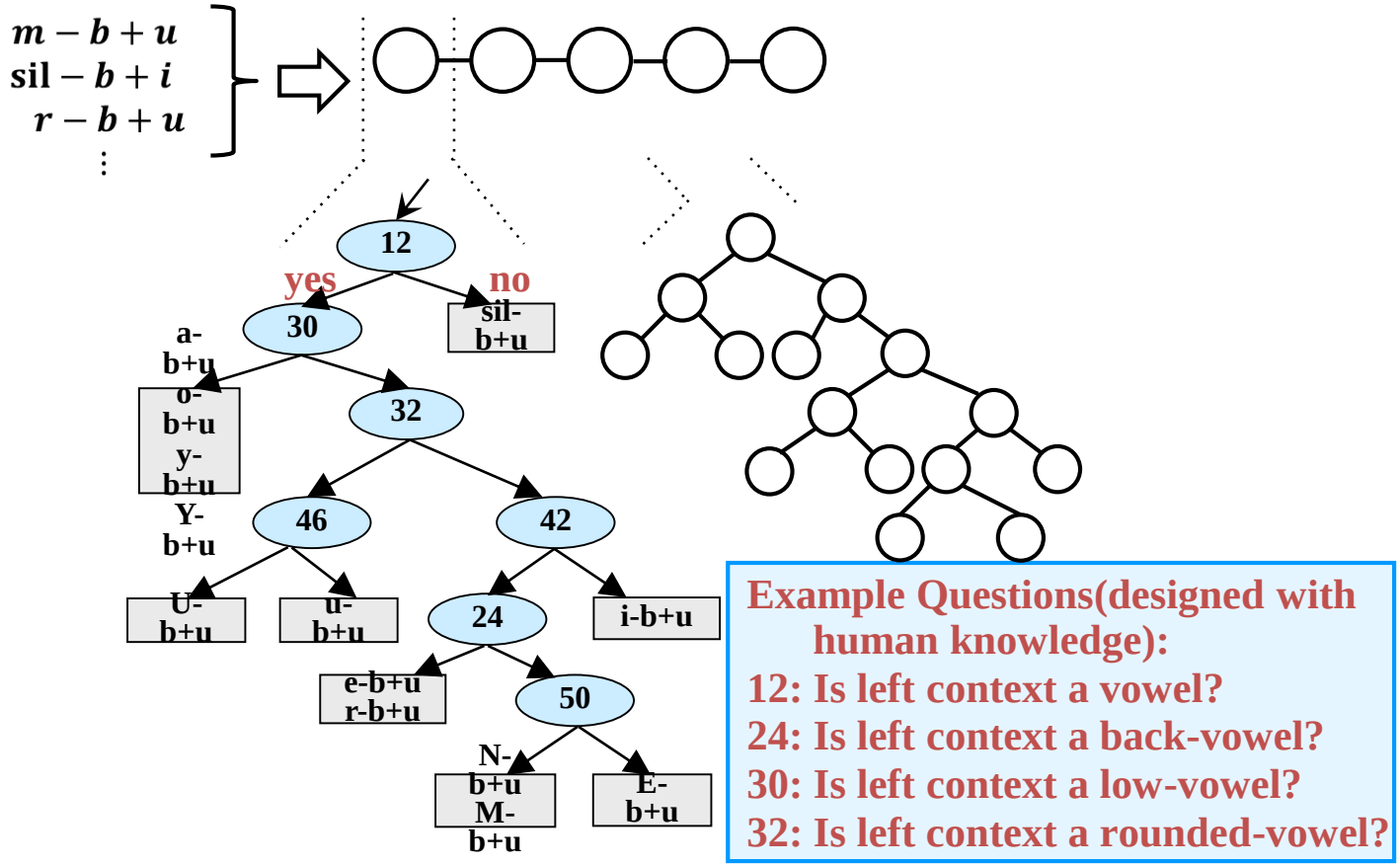
OOV的概念?



Shared Distribution Model (SDM)

Triphone

- Decision tree decides which triphones should be combined



Acoustic Model: Training Steps

- Get features(previous section)
- Train monophone model
- Use previous model to build decision tree for triphone
- Train triphone model

Acoustic Model: Training Steps

- Get features(previous section)
- Train monophone model
  - a. gmm-init-mono Initialize monophone model
  - b. compile-train-graphs Get train graph
  - c. align-equal-compiled model -> decode & align
    (**gmm-align-compiled** instead when looping)
  - d. gmm-acc-stats-ali EM training: E step
  - e. gmm-est EM training: M step
  - f. numgauss = numgauss + incgauss
  - g. Goto step c. Train several times
- Use previous model to build decision tree for triphone
- Train triphone model

Acoustic Model: Training Steps

- Get features(previous section)
- Train monophone model
- Use previous model to build decision tree for triphone
- Train triphone model
  - a. gmm-init-model Initialize GMM (from decision tree)
  - b. gmm-mixup Gaussian merging (increase #gaussian)
  - c. convert-ali Convert alignments(model <-> decisoin tree)
  - d. compile-train-graphs get train graph
  - e. gmm-align-compiled model -> decode&align
  - f. gmm-acc-stats-ali EM training: E step
  - g. gmm-est EM training: M step
  - h. numgauss = numgauss + incgauss
  - i. Goto step e. train several times

align-equal-compiled

- Write an equally spaced alignment (for getting training started)
- Usage: align-equal-compiled <graphs-rspecifier> <features-rspecifier> <alignments-wspecifier>
  e.g.
  align-equal-compiled 1.fsts 1.fsts scp:train.scp ark:equal.ali

gmm-align-compiled

- Performing re-alignment
- Usage: gmm-align-compiled [options] <model-in> <graphs-rspecifier> <feature-rspecifier> <alignments-wspecifier>
  e.g.
  align-equal-compiled 1.mdl ark:graphs.fsts1.fsts scp:train.scp ark:equal.ali

- gmm-align-compiled $scale_opts --beam=$beam --retry-beam=$[$beam*4] <hmm-model*> ark:$dir/train.graph ark,s,cs:$feat ark:<alignment*>
  - For first iteration(in monophone) beamwidth = 6, others = 10;
  - Only realign at
    mono: $realign_iters="1 2 3 4 5 6 7 8 9 10 12 14 16 18 20 23 26 29 32 35 38"
    tri: $realign_iters="10 20 30"

gmm-acc-stats-ali

- Accumulate stats for GMM training.(E step)
- Usage: gmm-acc-stats-ali [options] <model-in> <feature-rspecifier> <alignments-rspecifier> <stats-out>
  e.g.
  gmm-acc-stats-ali 1.mdl scp:train.scp ark:1.ali 1.acc

- gmm-acc-stats-ali --binary=false <hmm-model*> ark,s,cs:$feat ark,s,cs:<alignment*> <stats>

gmm-est

- Do Maximum Likelihood re-estimation of GMM-based acoustic model
- Usage: gmm-est [options] <model-in> <stats-in> <model-out>
  e.g.
  gmm-est 1.mdl 1.acc 2.mdl

- gmm-est --binary=false --write-occs=<*.occs> --mix-up=$numgauss <hmm-model-in> <stats> <hmm-model-out>
  --write-occs : File to write pdf occupation counts to.
  $numgauss increases every time.

Homework

03.mono.train.sh, 05.tree.build.sh, 06.tri.train.sh

To Do: Acoustic Modeling

- Step1. Execute the following commands.
  - script/03.mono.train.sh | tee log/03.mono.train.log
  - script/05.tree.build.sh | tee log/05.tree.build.log
  - script/06.tri.train.sh | tee log/06.tri.train.log
- Step2. finish code in TODO
  - script/03.mono.train.sh
  - script/06.tri.train.sh
- Step3. Observe the output and results.
- Step4. (opt.) tune #gaussian and #iteration.

Hint(**important!!**)

- Use the variables already defined.

```
numiters=40                                    # Number of iterations of training
maxiterinc=30                                  # Last iter to increase #Gauss on.
numgauss=300                                   # Initial num-Gauss (must be more than #states=3*phones).
totgauss=1000                                  # Target #Gaussians.
incgauss=$[($totgauss-$numgauss)/$maxiterinc] # per-iter increment for #Gauss
realign_iters="1 2 3 4 5 6 7 8 9 10 12 14 16 18 20 23 26 29 32 35 38";
scale_opts="--transition-scale=1.0 --acoustic-scale=0.1 --self-loop-scale=0.1"
```

- Use these formula:

```
x=`printf "%02g" $iter`
y=`printf "%02g" $[$iter+1]`
```

- Pipe for error
  - compute-mfcc-feats … 2> $log