Chapter 8

음향 모델

김지환

서강대학교 컴퓨터공학과

Table of contents

8. 음향 모델

- 음향 모델용 classifier가 가져야 할 특성
  - 모델의 구분 단위를 정할 수 있어야 함 (예: 음소)
  - 모델이 주어졌을 때 인식 결과 생성이 가능해야 함
  - 학습 자료가 주어졌을 때 모델 학습이 가능해야 함
  - 대용량 음성코퍼스로 부터 모델 구분 단위별 학습 자료를 자동 생성할 수 있어야 함
  - 모델 결합을 통한 문장 인식 확장성

8. 음향 모델

- 현재의 가장 좋은 성능을 보이는 시스템은 DNN-HMM을 기반으로 하고 있다.

특성	HMM	DNN
모델의 구분 단위를 정할 수 있어야 함	- Tri-phone	
모델이 주어졌을 때 인식 결과 생성이 가능해야 함	- Total Likelihood - Viterbi	-Multi-layer Feedforward Network
학습 자료가 주어졌을 때 모델 학습이 가능해야 함	- Baum Welch - Viterbi training algorithm	- Back Propagation
대용량 음성 코퍼스로부터 모델 구분 단위별 학습 자료를 자동 생성할 수 있어야 함	- Viterbi Segmentation Algorithm	
모델 결합을 통한 문장 인식 확장성	- HMM 기반 음성인식 디코딩 네트워크 사용	

8. 음향 모델

- HMM 기반 음성인식 대비, DNN 기반 음성인식의 성능은 relative improvement 기준 약 20% 좋은 것으로 정리되고 있음 [Hinton, 2012]
- DNN은 추정해야 하는 파라미터가 많아 학습 시간이 많이 소요됨
- DNN에서 모델의 구분 단위, 대용량 음성 코퍼스로부터 모델 구분 단위별 학습 자료 자동 생성 및 모델 결합을 통한 문장 인식 확장성 부분은 HMM의 해결 방안을 그대로 사용함

8.1.1 HMM 개요

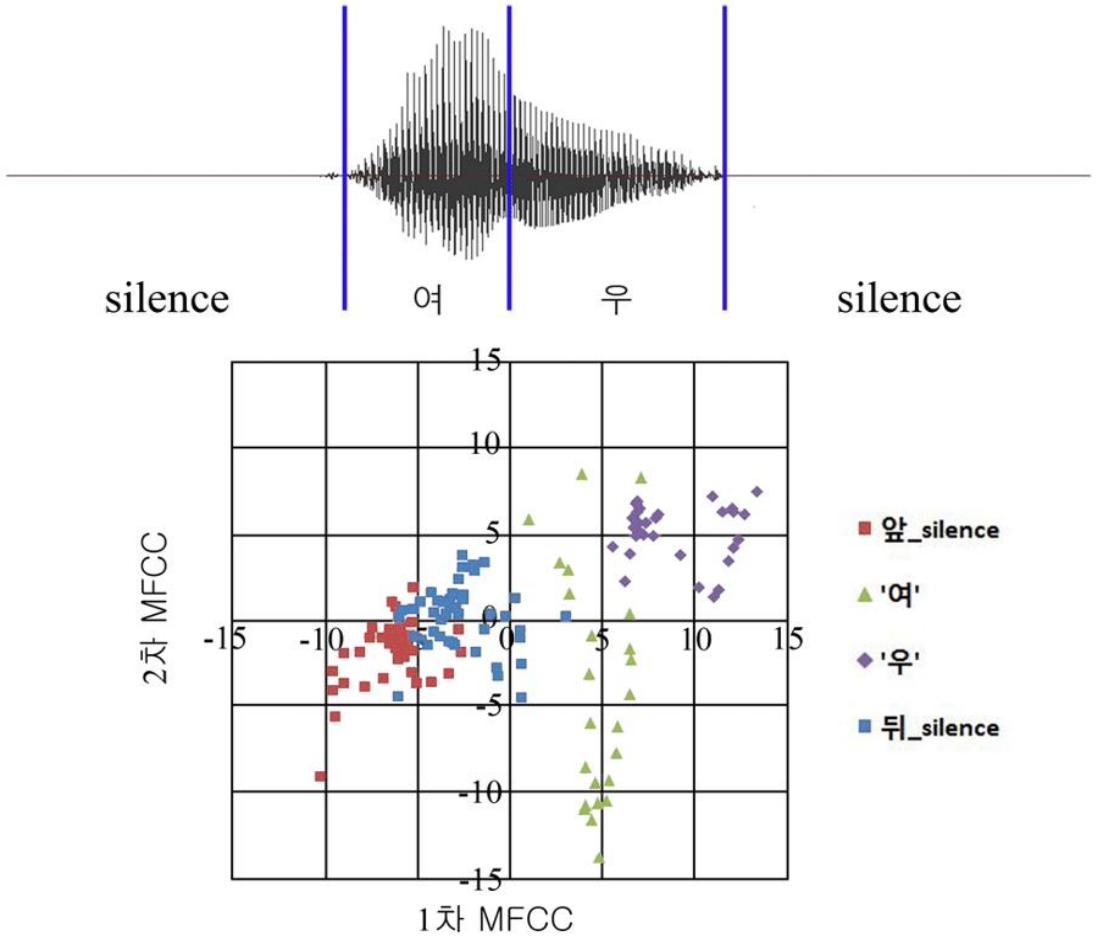
- HMM 예제 (Urn-and-Ball Model)

Observed
Ball Sequence

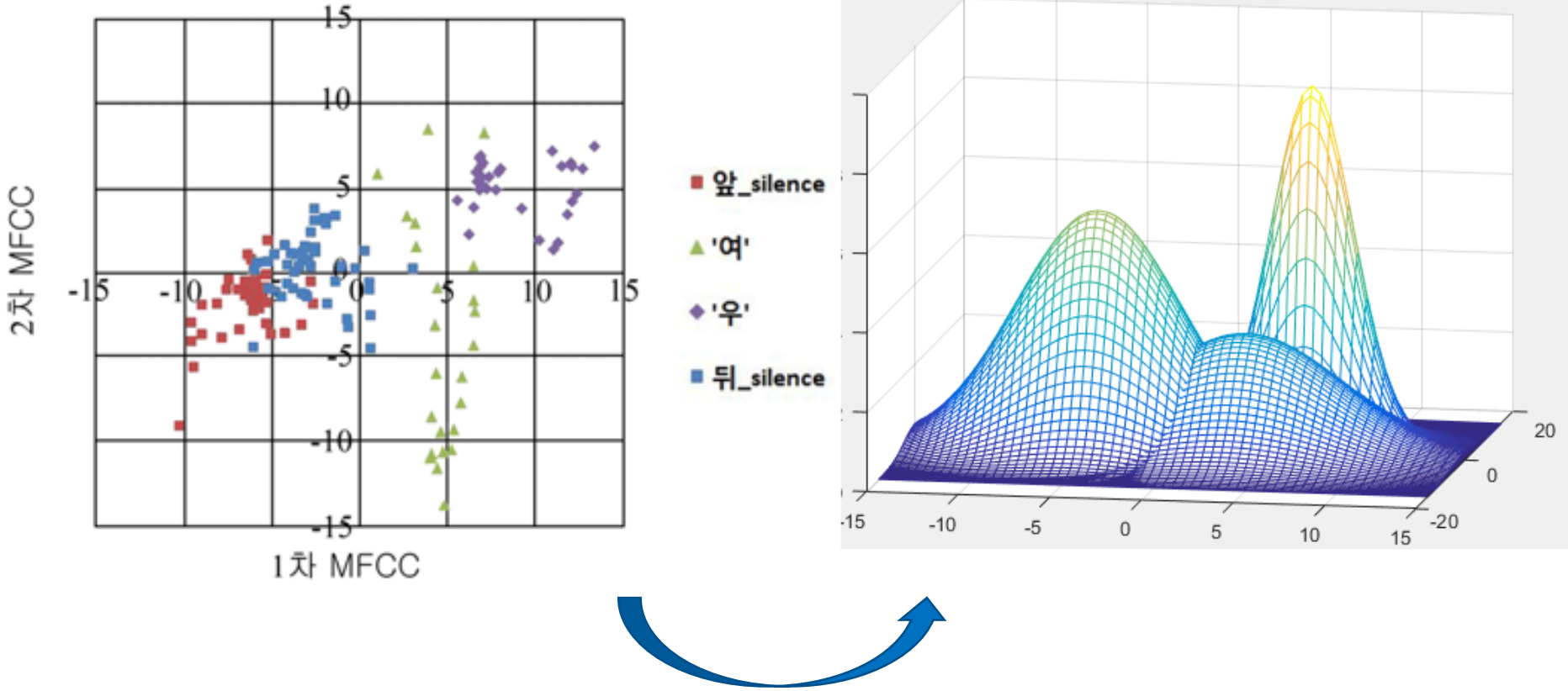
- 흰색, 노란색, 빨간색, 하얀색 공이 3개의 바구니에 담겨있다.
- 각 바구니별로 공들의 구성비율은 서로 다르다.
- 임의의 바구니를 선택한다.
- 바구니에서 공을 확인한다.
- 공을 다시 넣는다.
- 위와 같은 과정을 반복한다.
- Ball sequence를 보고 바구니를 선택한 순서를 판단한다.

8.1.1 HMM 개요 (구성요소)

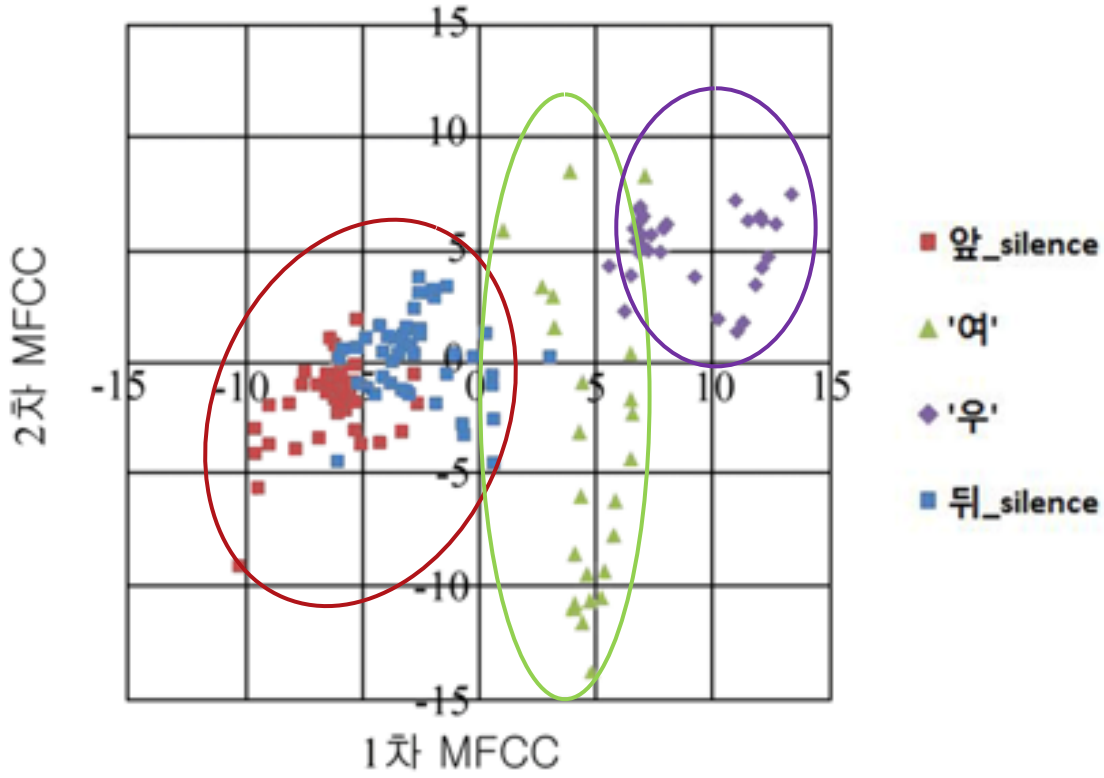
- 적절한 상태의 수는?



8.1.1 HMM 개요 (구성요소)



8.1.1 HMM 개요 (구성요소)



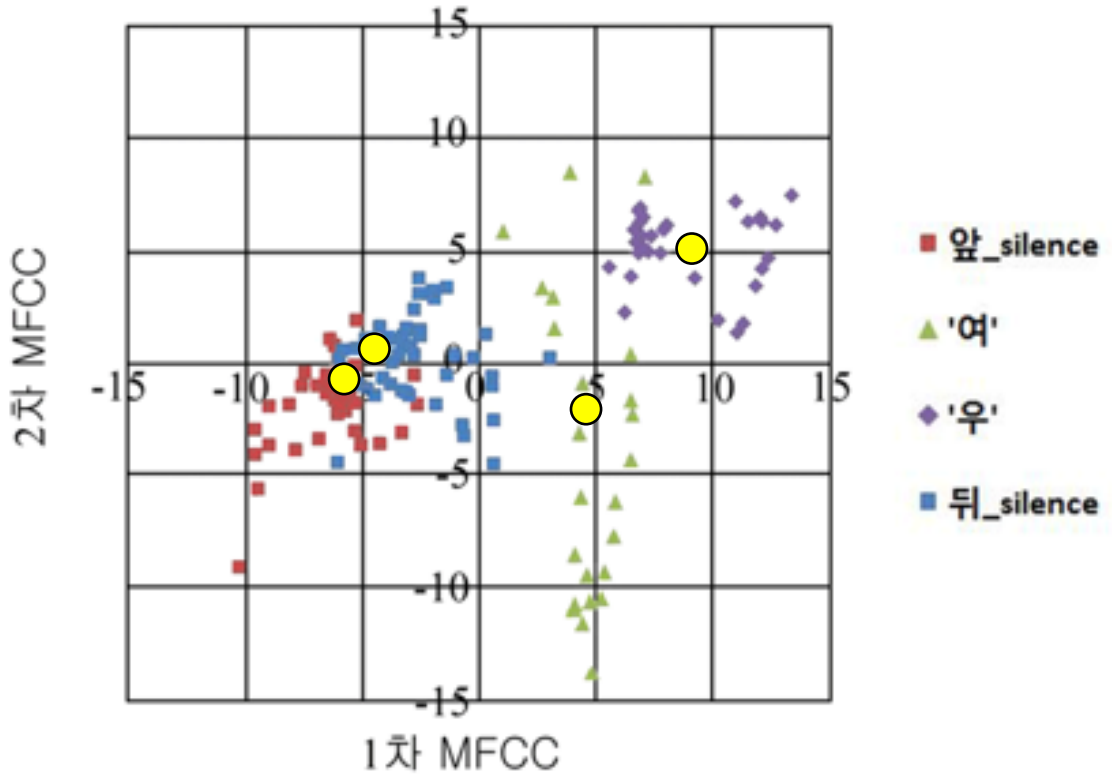
◆ Slience 바구니 :
: 60

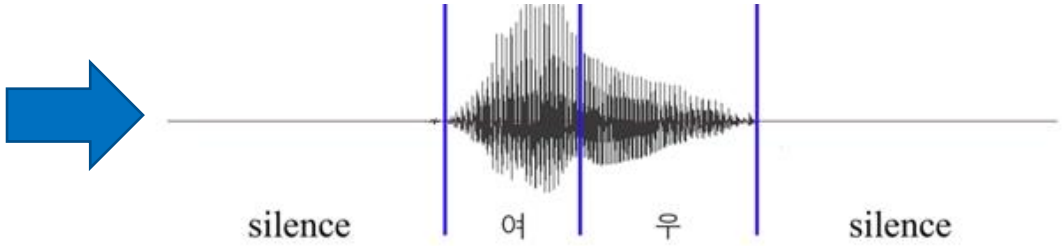
◆ '여' 바구니 :
: 28
: 4
: 5

◆ '우' 바구니 :
: 31
: 1

: 같은 slience 이므로 둘다 으로 취급

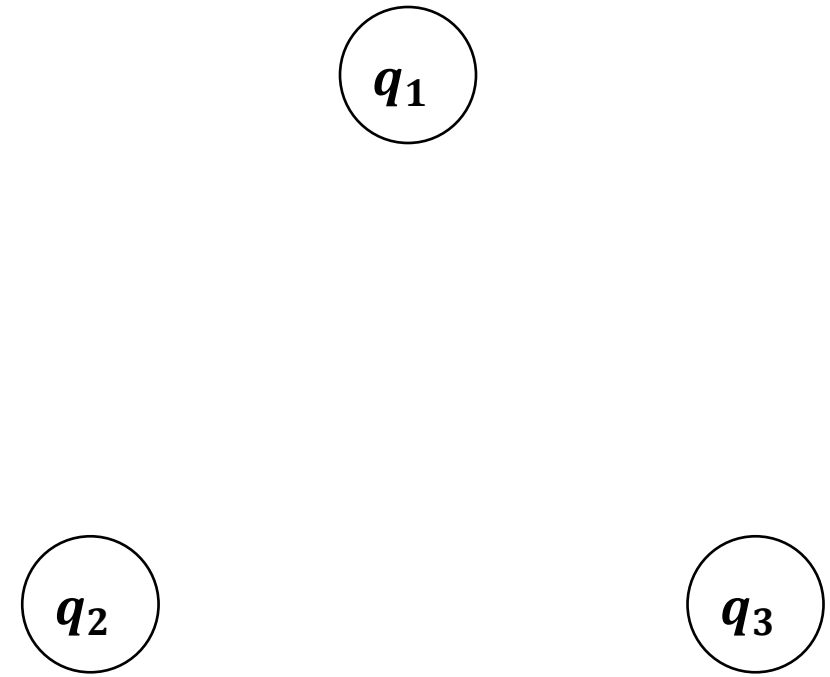
8.1.1 HMM 개요 (구성요소)





8.1.1 HMM 개요 (구성요소)

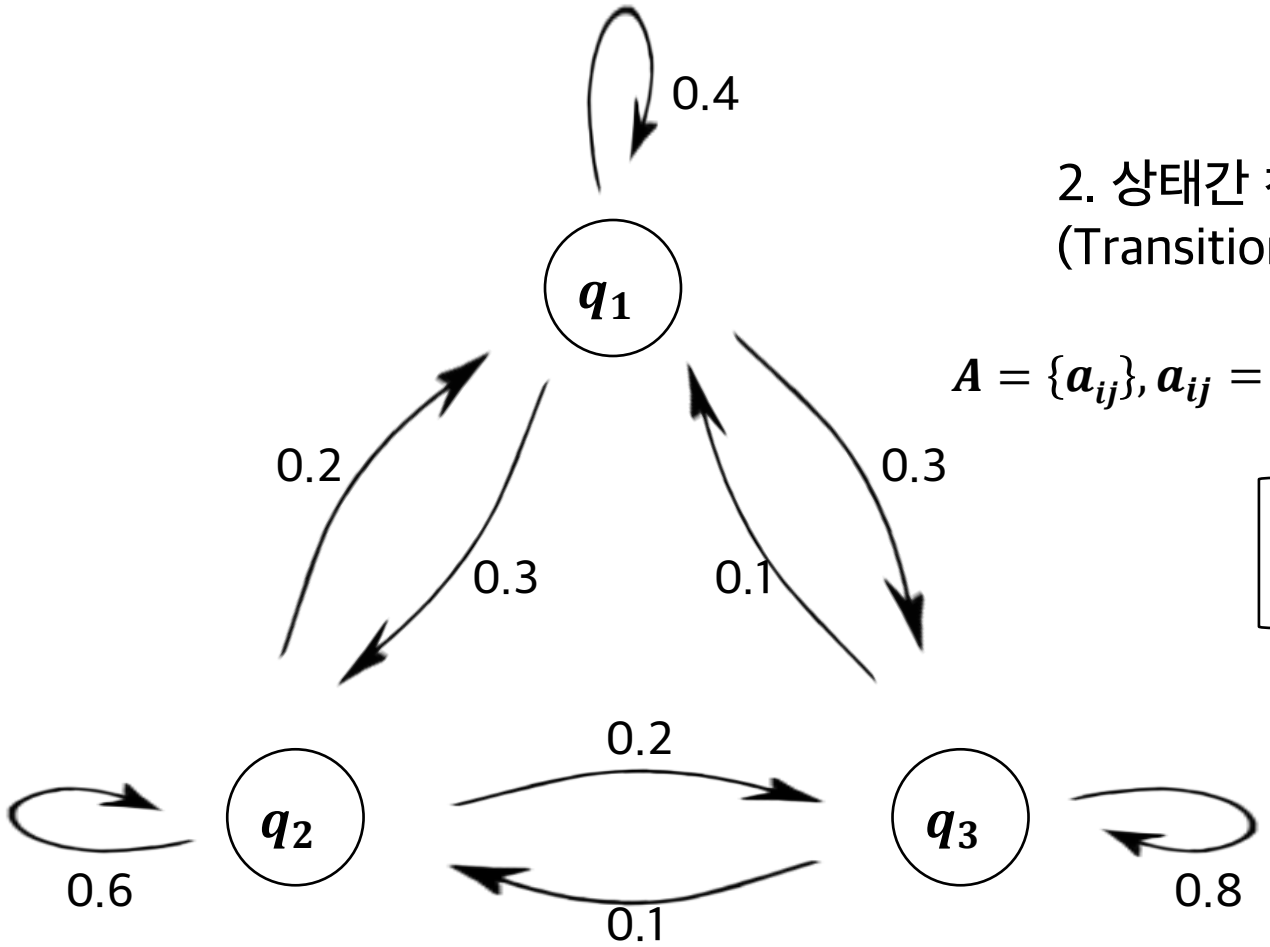
- 크게 4개의 요소로 구성됨



1. N개의 상태 (State)

$$Q = \{q_1, q_2, \ldots, q_N\}$$

8.1.1 HMM 개요 (구성요소)

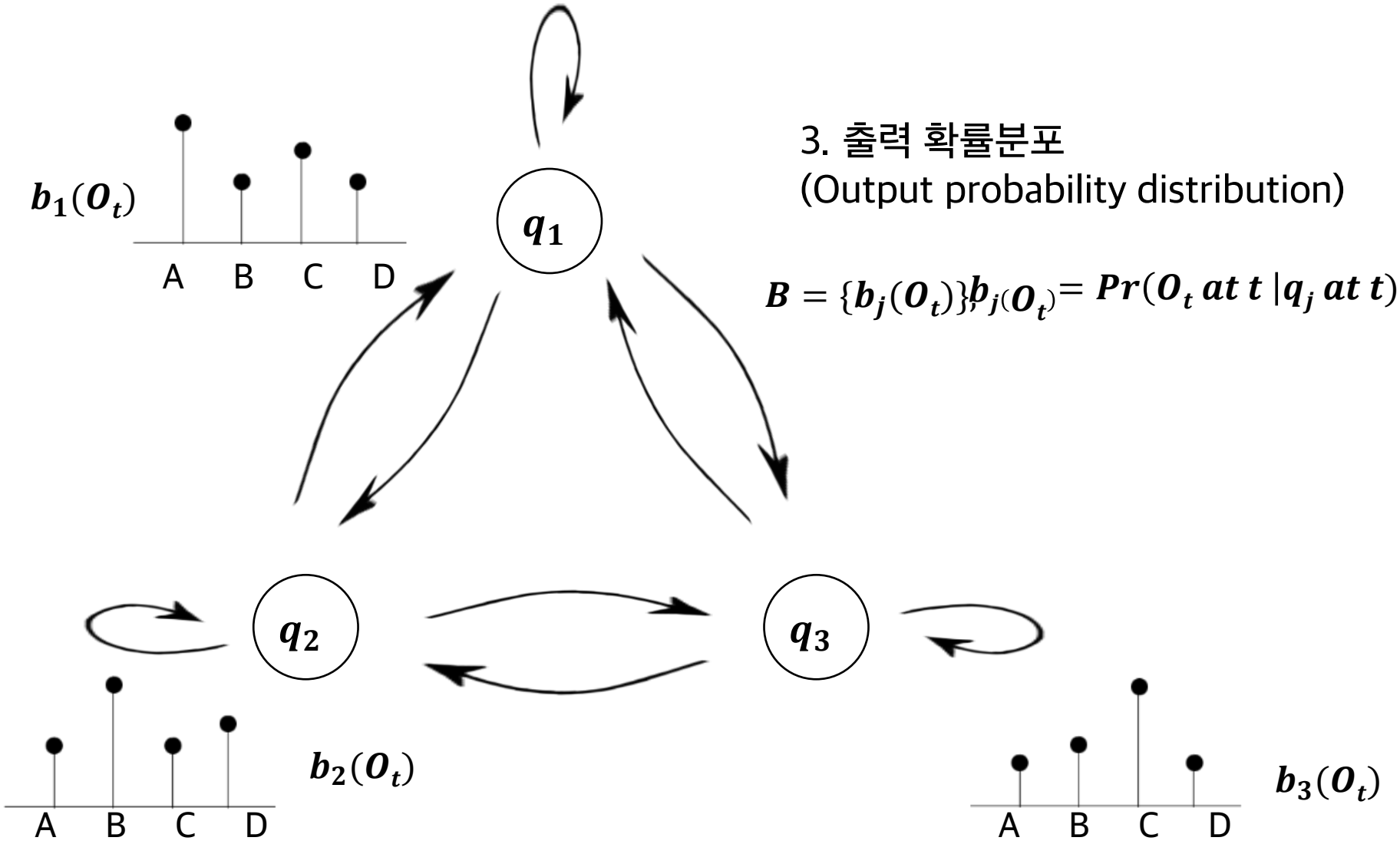


2. 상태간 천이 확률
(Transition probability)

$$A = \{a_{ij}\}, a_{ij} = Pr(q_j at\ t+1 \mid q_i\ at\ t)$$

$$\begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.2 & 0.8 \end{bmatrix}$$

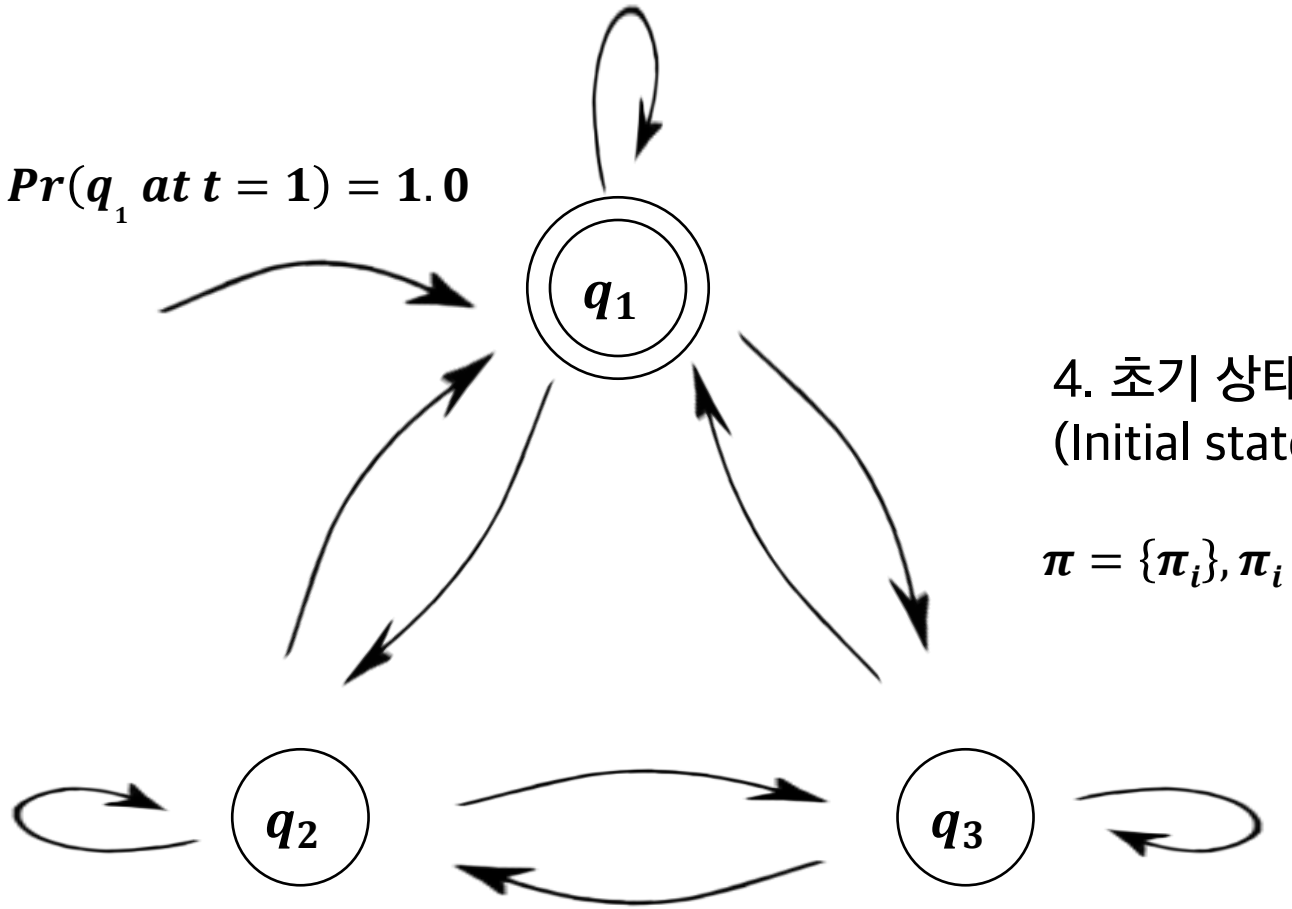
8.1.1 HMM 개요 (구성요소)



3. 출력 확률분포
(Output probability distribution)

$$B = \{b_j(O_t)\}, b_j(O_t) = Pr(O_t \; at \; t \mid q_j \; at \; t)$$

8.1.1 HMM 개요 (구성요소)

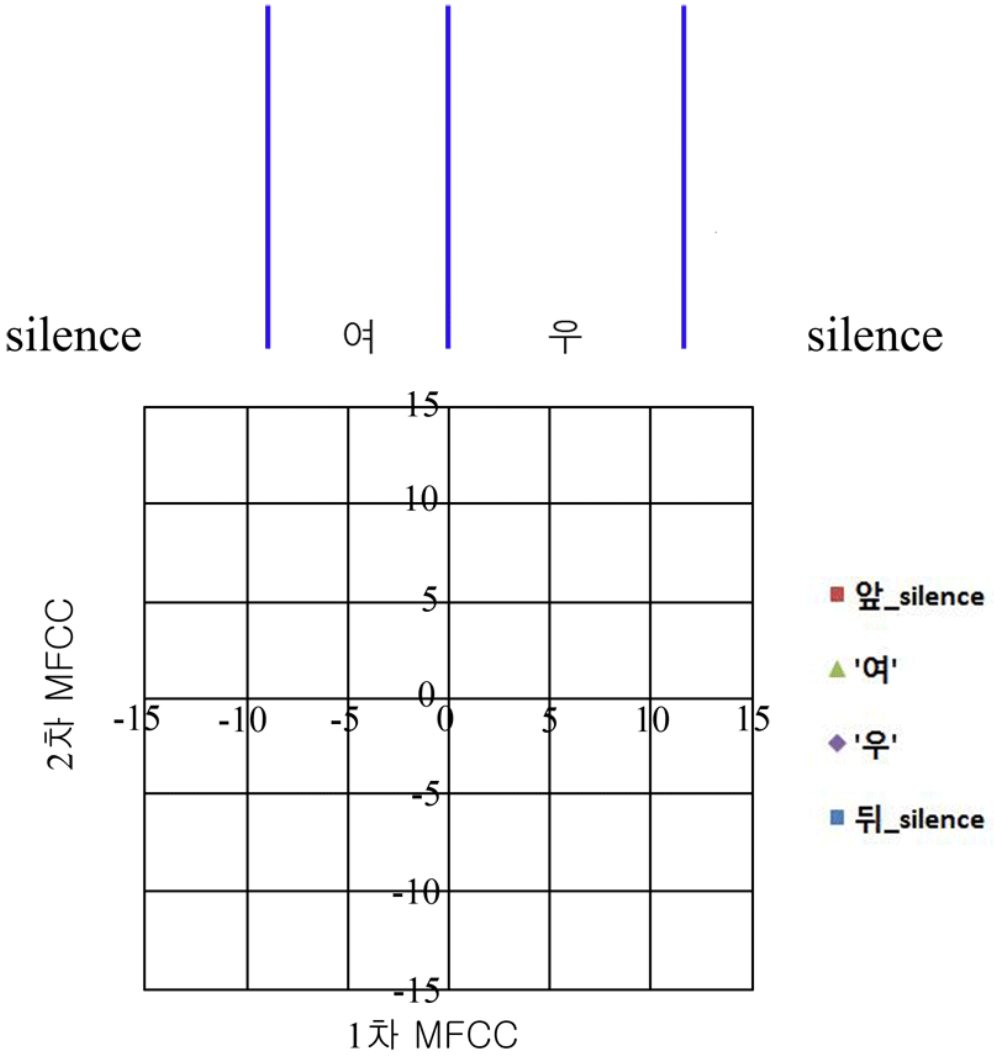


4. 초기 상태
(Initial state distribution)

$\pi = \{\pi_i\}, \pi_i = Pr(q_i\ at\ t = 1)$

8.1.1 HMM 개요 (구성요소)

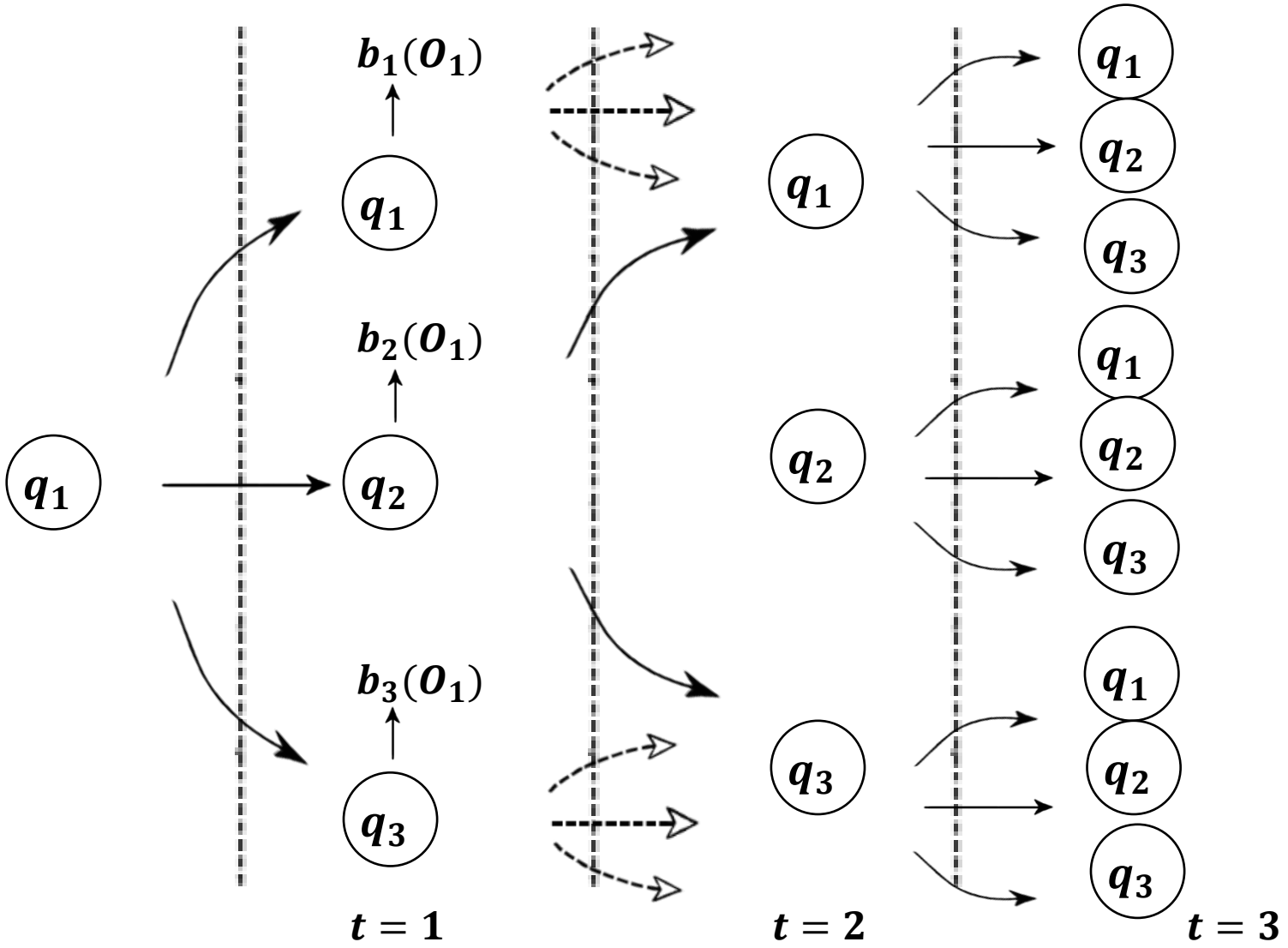
- 적절한 상태의 수는?



8.1.1 HMM 개요

- HMM으로부터 관측열을 발생 시키는 과정
  1. π에 따라 첫 상태 i가 결정됨
  2. $T=1$
  3. 상태 i에서 $b_i(o_1)$에 따라 o_1이 출력됨
  4. 상태 i에 대한 상태천이확률 a_{ij}에 따라 다음 상태 j로의 천이가 일어남
  5. 시간을 하나 증가시키고 $(t=t+1)$, 최종 시간에 도달하지 못했으면 $(t < T)$ 3번 과정으로 돌아가고, 도달했으면 관측열 생성을 종료한다.

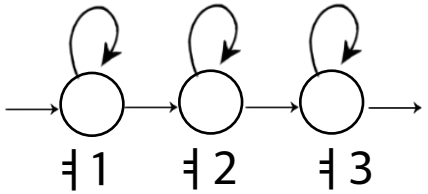
8.1.1 HMM 개요



8.1.1 HMM 개요

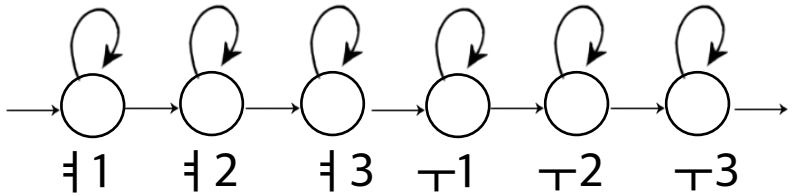
- HMM 의 확장적 구조의 예

'여'의 음소모델



N=3

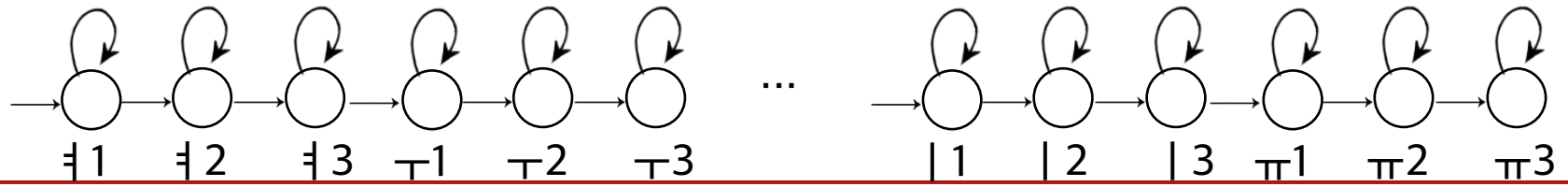
'여우'의 단어모델



N=6

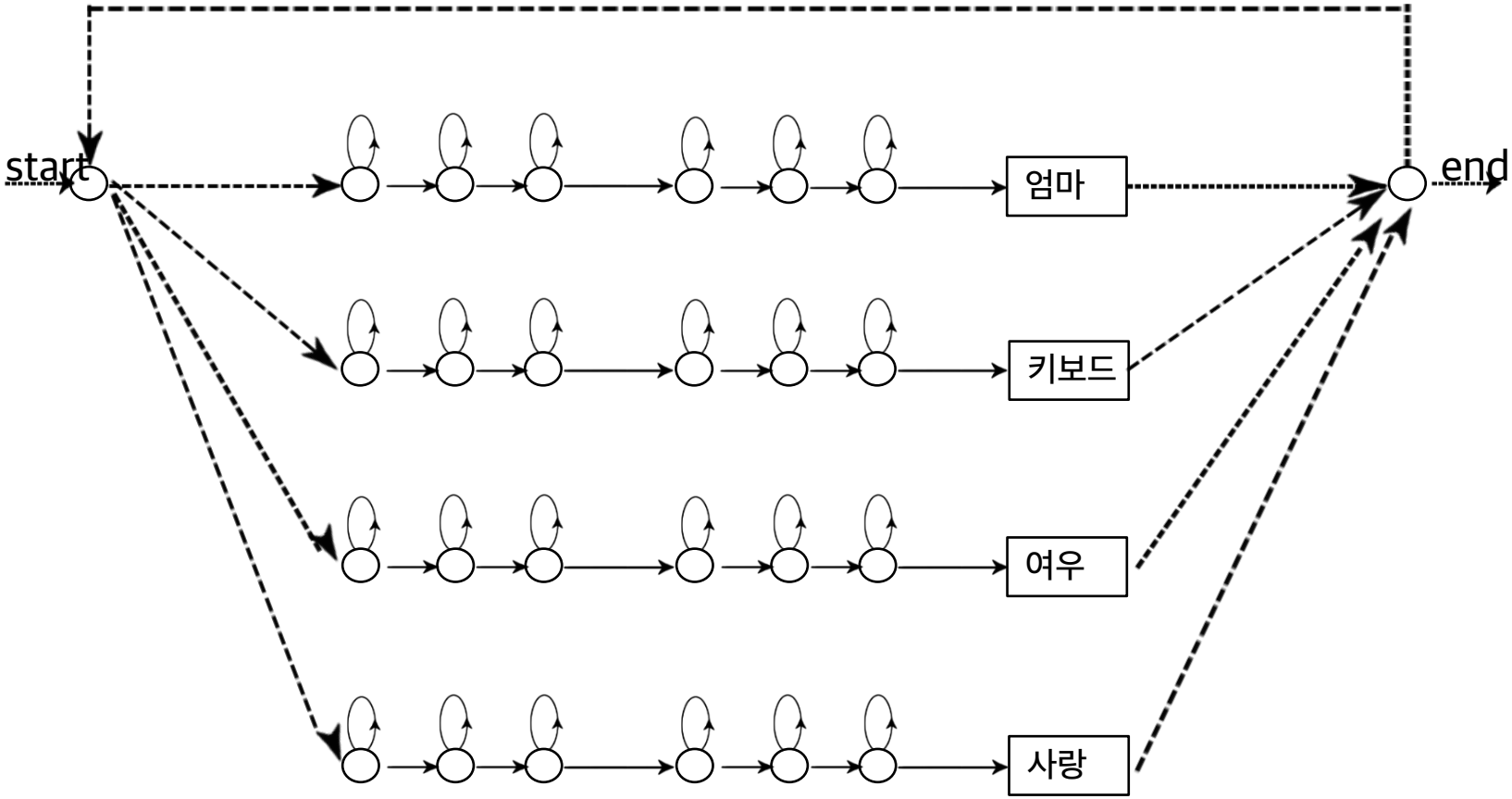
'여우가 멸종위기의 처한 이유'의 문장모델

N=72



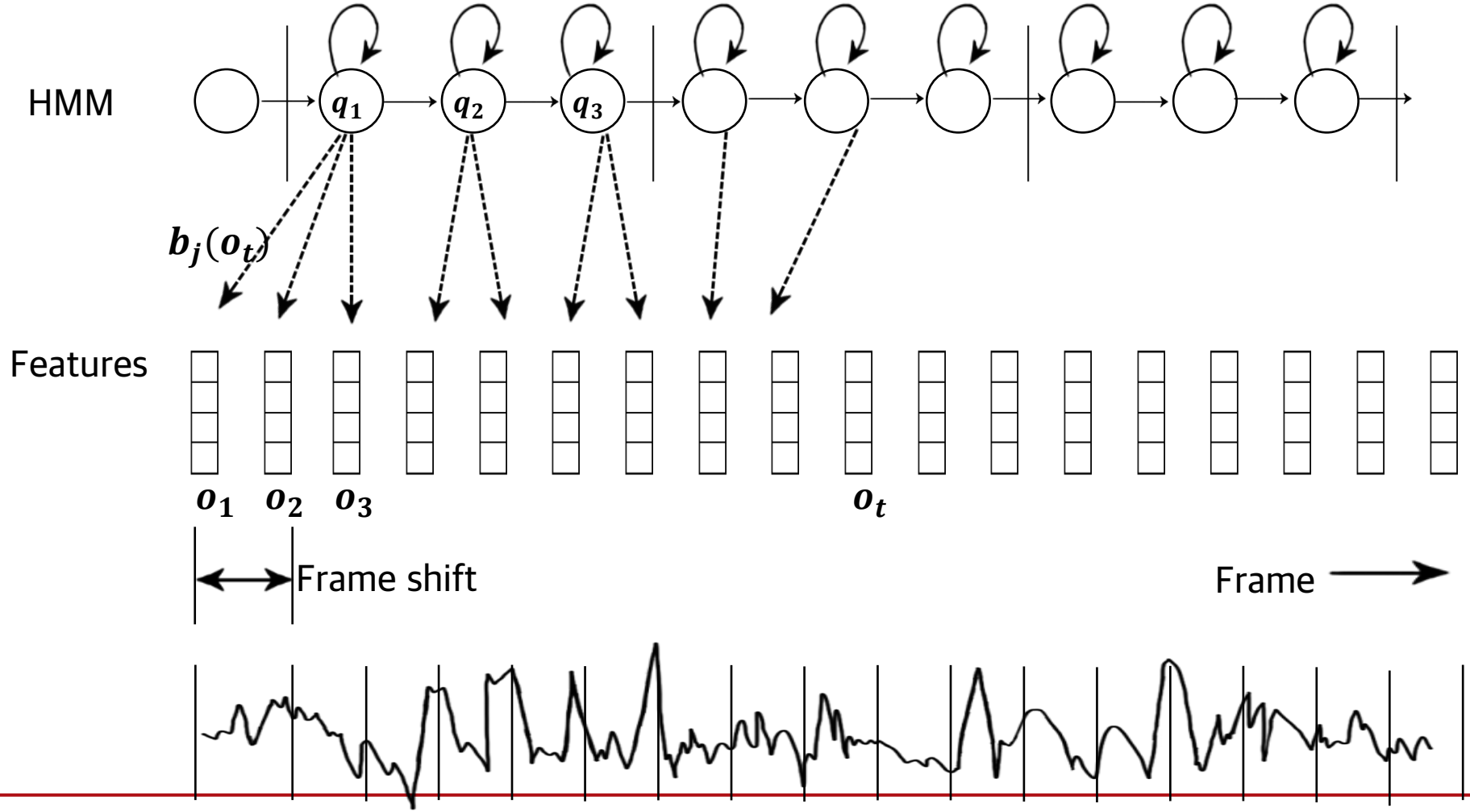
8.1.1 HMM 개요

- 연속 음성인식 네트워크



8.1.1 HMM 개요

- HMM을 이용한 음성 특징 벡터의 생성 예



8.2 HMM의 세가지 기본 문제

- 기본 문제 1: (인식) 관측열이 발생될 확률의 계산
  - 인식결과를 어떻게 결정할 것인가?
  - 모델 파라미터는 주어진 상황에서 관측열의 생성 확률을 최대로 하는 모델 선정
  - 알고리즘: Total likelihood method, forward algorithm

8.2 HMM의 세가지 기본 문제

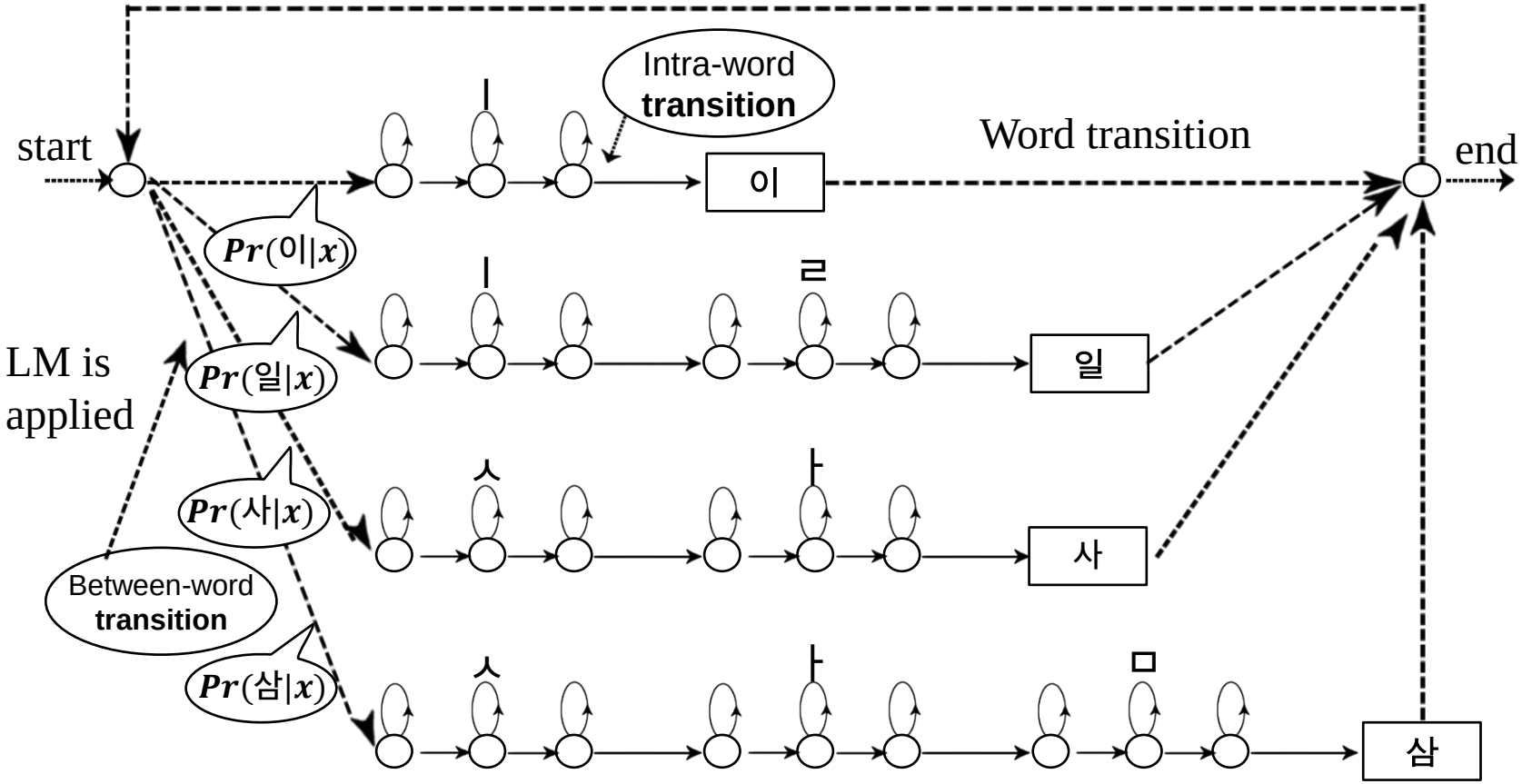
- 기본 문제 2: (Segmentation) 관측열이 어떤 상태천이를 거쳐 발생되었는지를 추정
  - 전체 음성신호 중 특정 단어를 찾아주는 segmentation시 필요
  - 전체 학습자료에서 모델 별 학습에 필요한 자료 자동 추출
  - 알고리즘: Viterbi algorithm

8.2 HMM의 세가지 기본 문제

- 기본 문제 3: (학습) HMM의 파라미터 추정
  - 모델 학습 시 필요
  - 알고리즘: Viterbi training algorithm, Baum-Welch algorithm
  - 본 자료는 expectation-maximization algorithm에 기초하여 작성되어 있음

8.2.1.1 HMM의 세가지 기본 문제 (인식: 결과 생성)

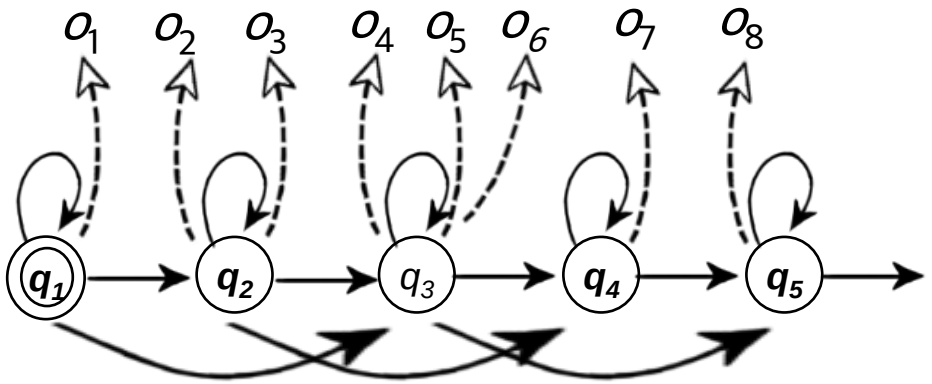
- 숫자 1,2,3,4 인식 네트워크 구조



- Recognized model = $arg_q max\ Pr(O|M_q)$

8.2.1.2 HMM의 세가지 기본 문제 (인식: 관측열 생성확률 계산 예제)

- 조건: 상태 이동 경로가 주어진 경우



- State 1으로 이동 후 o1을 생성하고, state 2로 이동 후 o2를 생성하고, state 2로 이동 후 o3 생성…..

$$Pr(\mathbf{O}, S|M) = \pi_1 b_1(o_1) a_{12} b_2(o_2) a_{22} b_2(o_3) a_{23} b_3(o_4) a_{33} b_3(o_5) a_{33} b_3(o_6) a_{34} b_4(o_7) a_{45} b_5(o_8)$$

$$Pr(\mathbf{O}, S|M) = \pi_{s(1)} b_{s(1)}(o_1) \prod_{t=2}^{T} a_{s(t-1),s(t)} b_{s(t)}(\boldsymbol{o_t})$$

8.2.1.3 HMM의 세가지 기본 문제 (인식: Total Likelihood Method)

- HMM에서 상태 열(state sequence)은 hidden되어 있음

- 따라서, $\Pr(O|M)$은 가능한 모든 상태열에 대해, 각각의 $\Pr(O,S|M)$을 누적함으로써 구할 수 있음

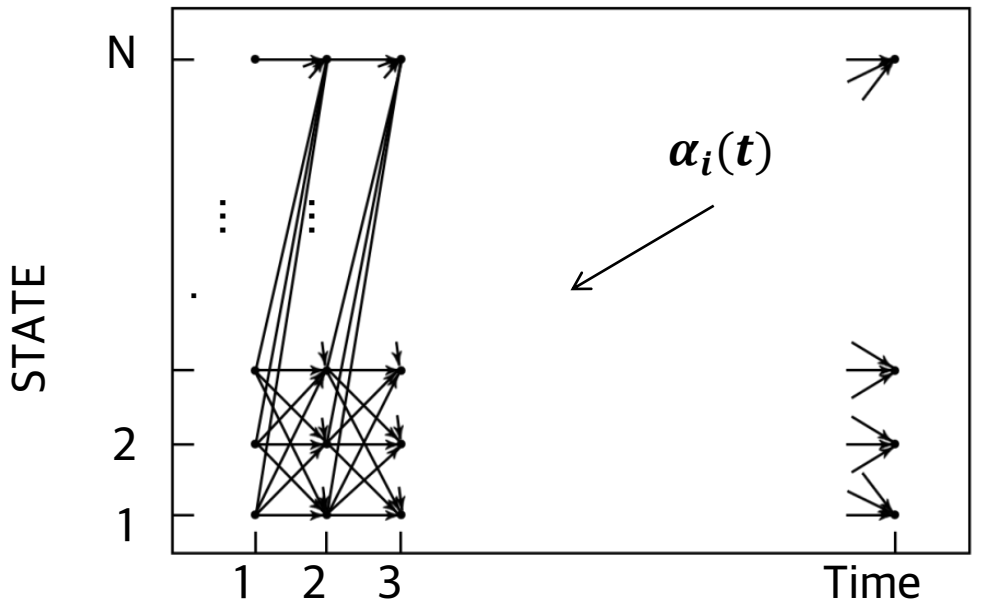
$$Pr(\mathbf{O}|\boldsymbol{M}) = \sum_{S} Pr(\mathbf{O},S|M)$$

- 가능한 모든 상태열의 개수가 N^T가 되어, 시간 복잡도가 $O(2\mathrm{T}N^{\mathrm{T}})$가 됨
  - 입력 음성의 길이에 따른 시스템의 반응 속도가 예측되지 않음
    - 1초의 음성에 대해서 0.5초에 인식 결과를 한 경우, 2초의 음성에 대해 필요한 시간은?

8.2.1.4 HMM의 세가지 기본 문제 (인식: Forward algorithm)

- Dynamic programming기법을 이용해서 total likelihood method의 시간 복잡도를 $O(N^2T)$로 줄이는 방법
- 기본 idea
  - 2차원 상에서의 $N * T$개의 각 격자점의 $\boldsymbol{\alpha_i(t)}$값을 저장

$$\boldsymbol{\alpha_i(t) = Pr(o_1 o_2 \ldots o_t, s(t) = i|\mathcal{M})}$$



- Algorithm
  - 초기화

$$\boldsymbol{\alpha_1(1) = 1}$$

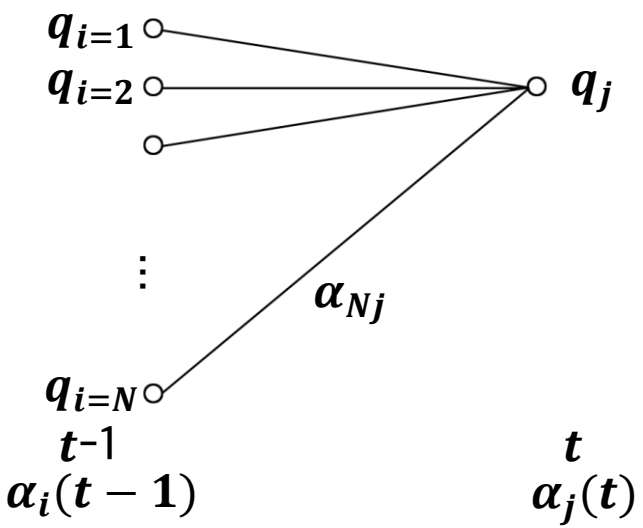
$$\boldsymbol{\alpha_1(t) = 0 \quad t > 0}$$

$$\boldsymbol{\alpha_i(1) = 0 \;\; 1 < i \leq N}$$

8.2.1.4 HMM의 세가지 기본 문제 (인식: Forward algorithm)

- Recursion

$$\alpha_j(t) = \left[\sum_{i=1}^{N} \alpha_i(t-1)a_{ij}\right] b_j(o_t)$$



8.2.2.1 HMM의 세가지 기본 문제 (Segmentation: Viterbi Algorithm)

◀ A N C ▶

교차로에서 좌회전할 때 지금은 신호등에 녹색 화살표가 표시되죠.
앞으로는 달라집니다.
지금도 괜찮은데 왜 바꾸냐는 목소리도 있는데요.
조영익 기자가 보도합니다.

◀ V C R ▶

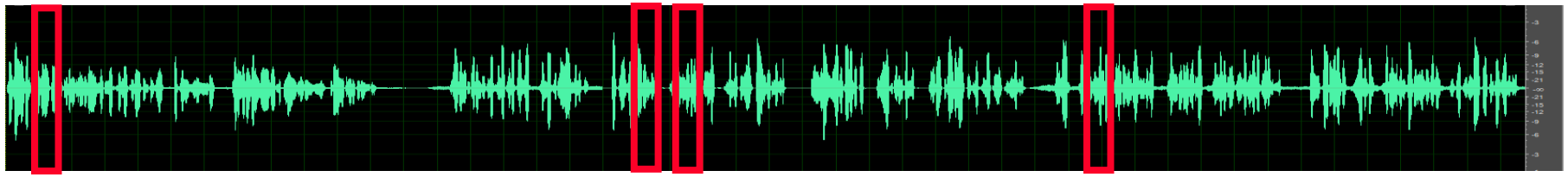
좌회전을 뜻하는 녹색 화살표가
포함된 현재 교차로의 4색등.

앞으로 이 좌회전 신호가
기존 신호등에서 분리돼
3색 등으로 따로 설치됩니다.

정지를 의미하는 빨간색과
주의를 뜻하는 황색, 그리고
좌회전의 녹색 화살표로
구분됩니다.

경찰은 광화문 삼거리 등
서울 도심 11곳에 새 신호등을 설치해
단계적으로 시범 운영에 들어갔습니다.

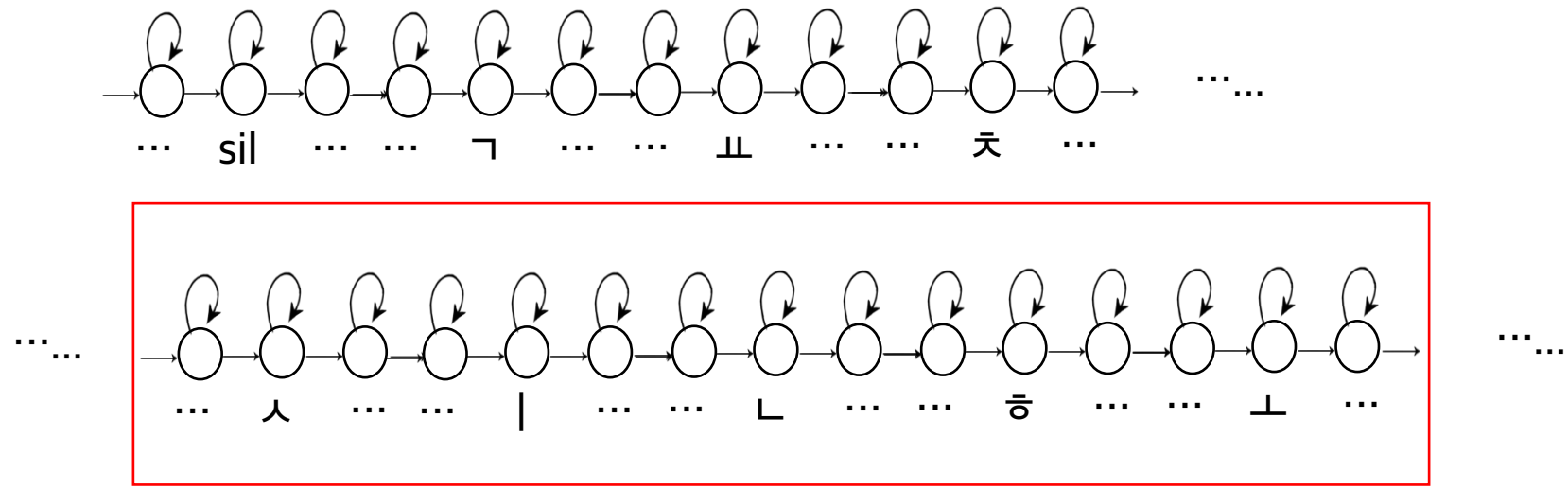
영국과 네덜란드, 스웨덴 등
서구 일부 국가에서 시행되고 있는
국제 표준과 맞추기 위해서입니다.



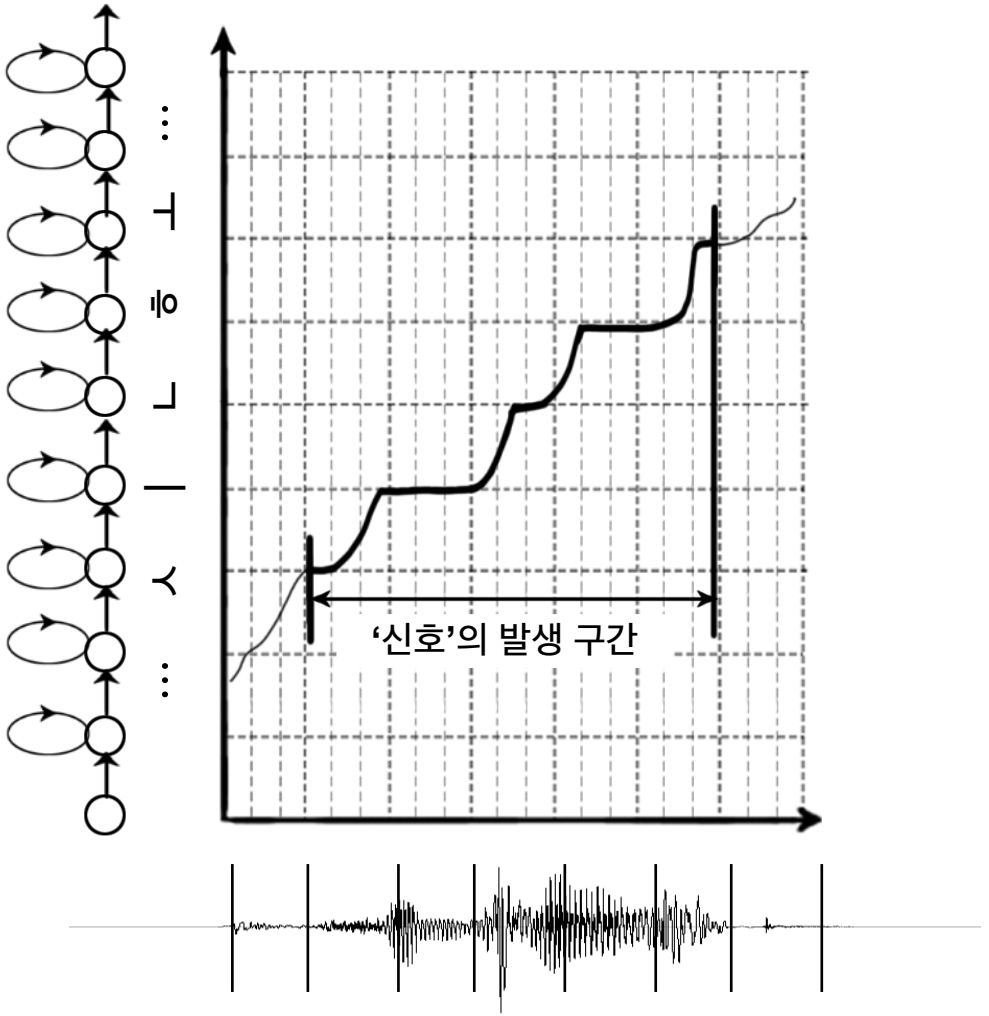
8.2.2.1 HMM의 세가지 기본 문제 (Segmentation: Viterbi Algorithm)

"교차로에서 좌회전할 때 … 좌회전 신호가 …"

sil ㄱㅛㅊㅏㄹㅗㅇㅔㅅㅓ ㅈㅘㅎㅚㅈㅓㄴㅎㅏㄹ ㄸㅐ…

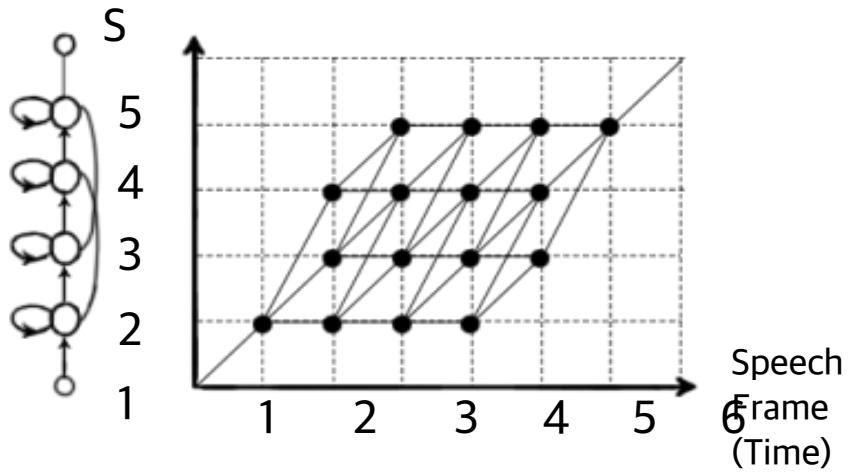


8.2.2.1 HMM의 세가지 기본 문제 (Segmentation: Viterbi Algorithm)



8.2.2.1 HMM의 세가지 기본 문제 (Segmentation: Viterbi Algorithm)

- Forward 알고리즘의 각 격자점에서 partial path에 대한 local decision을 수행하여, 최적의 상태열의 결정이 가능
- 알고리즘
  - Φj(t)=o1…ot를 생성하면서 s(t) = j에 도달하는 가장 적합한 path의 likelihood



8.2.2.1 HMM의 세가지 기본 문제 (Segmentation: Viterbi Algorithm)

- Dynamic programming 기법으로 $\Phi(t-1)$을 이용하여, $\Phi(t)$를 결정

$$\boldsymbol{\Phi_j(t) = \max_{1 \le i \le N} \left[\Phi_i(t-1) a_{ij} \right] b_j(o_t)}$$

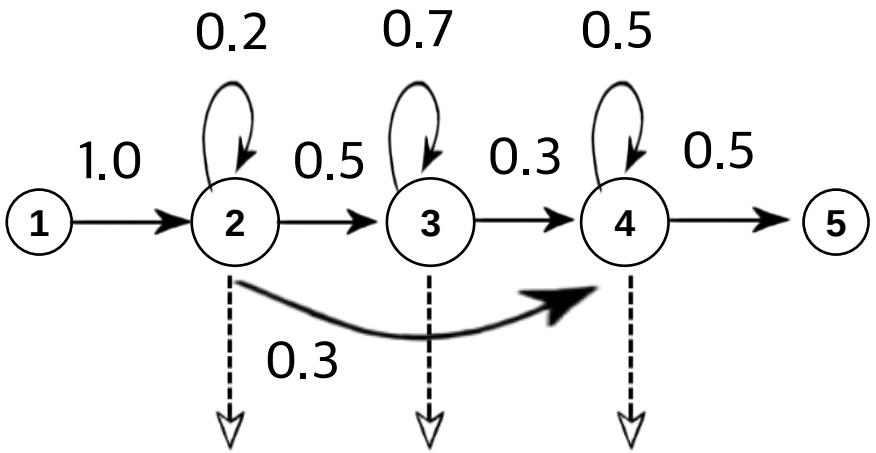
  - 최적 상태열에서의 likelihood 값은 아래와 같이 계산됨

$$\boldsymbol{Pr_{max}(\mathbf{O}|\mathcal{M}) = \max_{1 \le i \le N} [\Phi_i(T) a_{iN}]}$$

- 각 격자점에서의 local decision의 결과를 기록했다면, 최적의 상태열은 T로부터 trace back하면서 결정 가능
- 인식의 문제에도 적용 가능함
  - Viterbi algorithm으로 계산한 likelihood로써 모델로부터의 생성 확률을 근사함

8.2.2.2 HMM의 세가지 기본 문제 (HMM Likelihood 계산 예제)

- 아래의 5개의 상태(상태 1과 5는 dummy)로 구성된 HMM을 가정



time	o_t	$b_2(o_t)$	$b_3(o_t)$	$b_4(o_t)$
1	0.9	0.397	0.218	0.115
2	1.9	0.266	0.397	0.171
3	2.4	0.150	0.368	0.191
4	3.3	0.028	0.171	0.197

$$\mu_2 = 1.0 \quad \mu_3 = 2.0 \quad \mu_4 = 3.0$$
$$\sigma_2 = 1.0 \quad \sigma_3 = 1.0 \quad \sigma_4 = 2.0$$

8.2.2.2 HMM의 세가지 기본 문제 (HMM Likelihood 계산 예제)

- Forward algorithm

state					
4	0.000	0.000	0.0204	0.0077	0.0022
3	0.000	0.000	0.0788	0.0242	0.0029
2	0.000	0.397	0.0211	0.0006	0.000003
1	1.000	0.000	0.000	0.000	0.000
	0	1	2	3	4

Total $\boldsymbol{Pr(\mathbf{O}|\mathcal{M}) = 0.0011}$

- Viterbi algorithm

state

4	0.000	0.000	0.0204	0.0045	0.0012
3	0.000	0.000	0.0788	0.0203	0.0024
2	0.000	0.397	0.0211	0.0006	0.000003
1	1.000	0.000	0.000	0.000	0.0000
	0	1	2	3	4

Max $\boldsymbol{Pr(\mathbf{O}|\mathcal{M}) = 0.0006}$, 최적의 상태열은= 1, 2, 3, 3, 4, 5.

8.2.3 HMM의 세가지 기본 문제 (학습)

- Expectation maximization algorithm은 아래의 생성 확률을 최대가 되도록 파라미터 값을 수정함
  - R: 전체 학습자료에서 모델 M에 대응되는 데이터의 수

$$\prod_{r=1}^{R} Pr(\mathbf{O}^r|\mathcal{M})$$

  - 학습자료로부터 모델 파라미터의 업데이트는 자동으로 진행

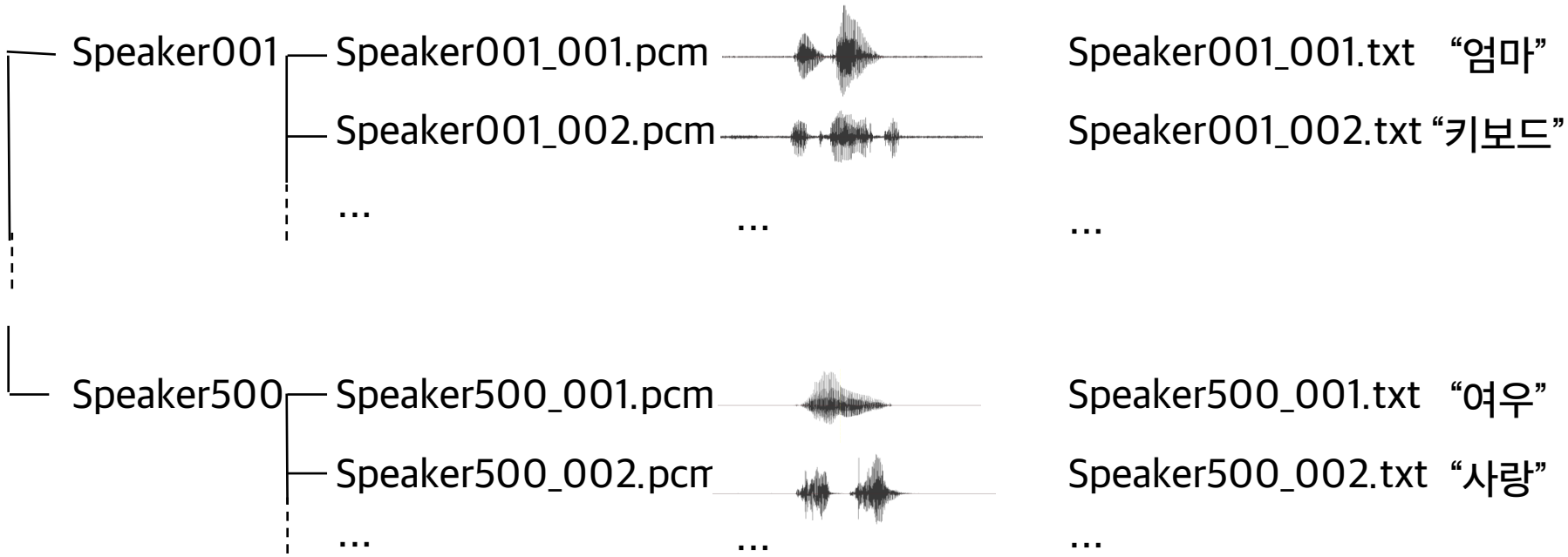
- 알고리즘
  - Viterbi training algorithm: 근사치로 학습진행. 속도가 빨라 실무에서 많이 사용됨
  - Baum-Welch algorithm: 정확한 방법이지만, 수식이 복잡하고, 시간이 많이 소요됨

8.2.3 HMM의 세가지 기본 문제 (학습)

- 학습 자료 구성의 예

  (예: 음성인식 학습용 SiTEC CleanWord01 코퍼스)

  - 화자 수 500명
  - 화자당 평균 417단어 발성



8.2.3 HMM의 세가지 기본 문제 (학습)

- 단어당 평균 음소의 수를 10개로 가정할 경우
  - 코퍼스 전체에서 나타난 음소의 수
    - 500명 x 417단어 x 10개 = 2,085,000 개
  - 인식단위가 음소이고, 전체 음소 수가 40개이면,
    - 평균 R = 2,085,000 / 40 = 52,125

8.2.3 HMM의 세가지 기본 문제 (학습)

- Viterbi training algorithm
  - 현재의 모델로 Viterbi algorithm에 따라 전체 학습자료에 대해 segmentation을 수행
  - Segmentation 결과를 바탕으로 다음 장과 같이 모델 파라미터를 업데이트 함
  - $\prod_{r=1}^{R} Pr(\mathbf{O}^r|\mathcal{M})$ 이 더 증가하지 않을 때 까지 위의 과정을 반복

8.2.3.1 HMM의 세가지 기본 문제 (학습: Viterbi Training Algorithm)

- a_{ij}는 아래와 같이 업데이트됨

$$\hat{a}_{ij} = \frac{\text{Estimated number of transitions state } i \rightarrow \text{state } j}{\text{Estimated number of transitions from state } i}$$

- $b_j(o_t)$가 하나의 정규분포로 모델링 된 경우정규분포의 mean과 variance는 아래와 같이 업데이트 됨.

$$\hat{\mu}_j = \frac{\text{Estimated sum of vectors emitted from } j}{\text{Estimated number of vectors emitted from } j}$$

$$\hat{\sum}_j = \frac{\text{Estimated sum of}(\boldsymbol{o}_t - \hat{\mu}_j)(\boldsymbol{o}_t - \hat{\mu}_j)'}{\text{Estimated number of vectors emitted from } j}$$

8.2.3.2 HMM의 세가지 기본 문제 (학습: Baum-Welch Algorithm)

- Viterbi training algorithm은 특정 시점에 특정 상태에 있는 확률을 0 또는 1로 hard decision
- Baum-Welch algorithm에서는 특정 시점에 특정 상태에 $L_j(t)$의 확률로 있을 수 있다고 가정
  - Viterbi algorithm은 Baum-Welch algorithm의 특수한 경우
- 파라미터 값들은 가중 평균값으로 추정됨

$$\hat{\mu}_j = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T^r} L_j^r(t)\boldsymbol{o}_t^r}{\sum_{r=1}^{R}\sum_{t=1}^{T^r} L_j^r(t)}$$

- Viterbi training algorithm과 같이, Baum-Welch algorithm도 반복적인 모델 업데이트로 학습

8.2.3.2 HMM의 세가지 기본 문제 (학습: Baum-Welch Algorithm)

- $L_j(t)$

$$Pr(s(t) = j, \mathbf{O}|\mathcal{M}) = \alpha_j(t)\beta_j(t)$$

$$L_j(t) = Pr(s(t) = j|\mathbf{O}, \mathcal{M})$$
$$= \frac{1}{Pr(\mathbf{O}|\mathcal{M})}\alpha_j(t)\beta_j(t)$$

8.2.3.2 HMM의 세가지 기본 문제 (학습: Baum-Welch Algorithm)

- 비슷한 방법으로

$$Pr(s(t)=i, s(t+1)=j, \mathbf{O}|\mathcal{M}) = \alpha_i(t)a_{ij}b_j(\mathbf{o}_{t+1})\beta_j(t+1)$$

$$Pr(s(t)=i, s(t+1)=j|\mathbf{O}, \mathcal{M}) = \frac{1}{Pr(\mathbf{O}|\mathcal{M})}\alpha_i(t)a_{ij}b_j(\mathbf{o}_{t+1})\beta_j(t+1)$$

- 따라서, a_{ij}에 대한 재추정 식은 아래와 같다

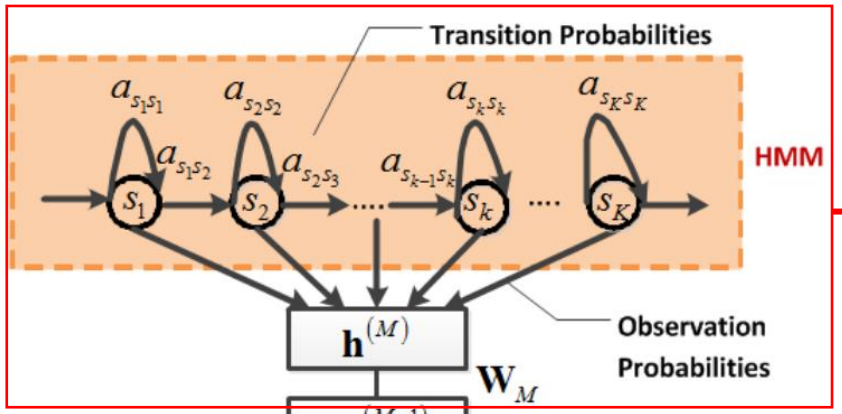
$$\hat{a}_{ij} = \frac{\sum_{r=1}^{R}\frac{1}{Pr(\mathbf{O}^r|\mathcal{M})}\sum_{t=1}^{T^r}\alpha_i^r(t)a_{ij}b_j(\boldsymbol{o}_{t+1}^r)\beta_j^r(t+1)}{\sum_{r=1}^{R}\sum_{t=1}^{T^r}L_i^r(t)}$$

- $1 \le i,\ j < N$인 경우, final state transition은 다음과 같다

$$\hat{a}_{iN} = \frac{\sum_{r=1}^{R}L_i^r(T^r)}{\sum_{r=1}^{R}\sum_{t=1}^{T^r}L_i^r(t)}$$

8.3 Deep Neural Network (DNN)을 이용한 음향모델

- Gaussian Mixture Model (GMM)의 역할을 DNN이 대체 [Hinton, 2013]
  - DNN 모델로 Deep Belief Network (DBN) 사용 [Hinton, 2006]



Network output

$$P(o_i|s^k) = \frac{P(s^k|o_i)P(o_i)}{P(s^k)} \approx \frac{P(s^k|o_i)}{P(s^k)}$$

Emission prob.

[Dahl, 2012]

8.3 Deep Neural Network (DNN)을 이용한 음향모델

- DNN 기반 음향모델 시스템 구성 [Hinton 2010]
  - 입력 layer unit의 개수: 600개 (= 15frames X 40 filterbank outputs)
  - Hidden layer의 개수: 5~8 개 (Layer 당 unit의 개수: 2,048 units)
  - 출력 layer unit의 개수: 6,000 ~ 10,000개 (Tied state 수)

8.3.1 개요 - 음소(Phoneme)

- **음소**(Phoneme): 다른 소리와 구별되어 언어 사용자가 인식하는 소리의 최소 단위
- 예: 영어단어 발음에 나타나는 발음기호 하나하나
  - **phoneme**[foʊni:m]
  - CMU dictionary에서 사용하는 영어 phoneme set

index	Phoneme	Example	Translation	index	Phoneme	Example	Translation	index	Phoneme	Example	Translation
1	AA	odd	AA D	15	G	green	G R IY N	28	R	read	R IY D
2	AE	at	AE T	16	HH	he	HH IY	29	S	sea	S IY
3	AH	hut	HH AH T	17	IH	it	IH T	30	SH	she	SH IY
4	AO	ought	AO T	18	IY	eat	IY T	31	T	tea	T IY
5	AW	cow	K AW	19	JH	gee	JH IY	32	TH	theta	TH EY T AH
6	AY	hide	HH AY D	20	K	key	K IY	33	UH	hood	HH UH D
7	B	be	B IY	21	L	lee	L IY	34	UW	two	T UW
8	CH	cheese	CH IY Z	22	M	me	M IY	35	V	vee	V IY
9	D	dee	D IY	23	N	knee	N IY	36	W	we	W IY
10	DH	thee	DH IY	24	NG	ping	P IH NG	37	Y	yield	Y IY L D
11	EH	Ed	EH D	25	OW	oat	OW T	38	Z	zee	Z IY
12	ER	hurt	HH ER T	26	OY	toy	T OY	39	ZH	seizure	S IY ZH ER
13	EY	ate	EY T	27	P	pee	P IY	40	SI	silence	silence
14	F	fee	F IY								

www.speech.cs.cmu.edu/cgi-bin/cmudict

8.3.1 개요 - 음소(Phoneme)

- 국립국어원 한글맞춤법에 따른 한국어 phoneme set

ㄱ(기역)	ㄴ(니은)	ㄷ(디귿)	ㄹ(리을)	ㅁ(미음)
ㅂ(비읍)	ㅅ(시옷)	ㅇ(이응)	ㅈ(지읒)	ㅊ(치읓)
ㅋ(키읔)	ㅌ(티읕)	ㅍ(피읖)	ㅎ(히읗)	
ㅏ(아)	ㅑ(야)	ㅓ(어)	ㅕ(여)	ㅗ(오)
ㅛ(요)	ㅜ(우)	ㅠ(유)	ㅡ(으)	ㅣ(이)

  - 위 자모로써 적을 수 없는 소리는 두 개 이상의 자모를 어울려서 적되, 그 순서와 이름은 다음과 같이 정한다.

ㄲ(쌍기역)	ㄸ(쌍디귿)	ㅃ(쌍비읍)	ㅆ(쌍시옷)	ㅉ(쌍지읒)	
ㅐ(애)	ㅒ(얘)	ㅔ(에)	ㅖ(예)	ㅘ(와)	ㅙ(왜)
ㅚ(외)	ㅝ(워)	ㅞ(웨)	ㅟ(위)	ㅢ(의)	

- 동일한 언어에 대해서도 적용하는 방법에 따라 phoneme set은 다를 수 있음

8.3.1 개요 - 음소(Phoneme)

- 음소열 변환 예제

  - 문장 예: ‘교육에 관해서 이와 같은 의견을 말하는 것은…’
  - 음소열 변환결과: ㄱ ㅛ ㅇ ㅠ ㄱ ㅞ ㄱ ㅘ ㄴ ㅎ ㅐ ㅅ ㅓ # ㅇ ㅣ ㅇ ㅘ ㄱ ㅏ ㅊ ㅡ ㄴ ㅇ ㅢ ㄱ ㅕ ㄴ ㅡ ㄹ # ㅁ ㅏ ㄹ ㅎ ㅏ ㄴ ㅡ ㄴ ㄱ ㅓ ㅅ ㅡ ㄴ …
  - 음소열 변환결과에 따른 발음대로 작성한 문장: 교유게 관해서 이와가튼 의겨늘 말하는 거슨 …

8.3.1 개요 - Tri-phone

- 음성인식에서는 일반적으로 앞뒤 context phoneme에 따라 음소를 다르게 모델링함

- 이를 tri-phone이라 함
  - 왼쪽 context phoneme, 해당 phoneme, 오른쪽 context phoneme으로 모델링함
  - green(G R IY N)의 IY의 경우
    - 왼쪽 phoneme R, 해당 phoneme IY, 오른쪽 phoneme N 으로 모델링함

8.3.1 개요 - Tri-phone

- Tri-phone모델
  - ‘좌측 phoneme – 해당phoneme + 우측 phoneme’ 의 형태의 notation으로 표기
    - 따라서 앞장의 IY는 ‘R – IY + N’ 의 형태로 표기함

- Phoneme하나당 모델링 되는 tri-phone의 (이론상)개수?
  - Phoneme이 40개인 경우, 좌측 context phoneme이 40개, 오른쪽 context phoneme이 40개가 위치 가능
  - Phoneme 하나당 (40*40)=1600개의 모델이 가능
  - 따라서 모든 phoneme은 (40*40)*40=64000개의 모델이 가능

- ‘R – IY + N’ 은 64000개 모델 중 몇 번째 tri-phone모델인가?

8.3.1 개요 - Tri-phone

- ‘R – IY + N’
- Phoneme index
  - Center: 18 >> 1600*(18-1)=27200
  - Left : 28 >> 40*(28-1)=1080
  - Right: 23

index	Phoneme	index	Phoneme	index	Phoneme
1	AA	15	G	28	R
2	AE	16	HH	29	S
3	AH	17	IH	30	SH
4	AO	18	IY	31	T
5	AW	19	JH	32	TH
6	AY	20	K	33	UH
7	B	21	L	34	UW
8	CH	22	M	35	V
9	D	23	N	36	W

27200+1080+23=
28303rd

1	AA	AA	AA
2	AA	AA	AE
3	AA	AA	AH
27201	AA	IY	AA
27202	AA	IY	AE
27203	AA	IY	AH
28281	R	IY	AA
28282	R	IY	AE
28283	R	IY	AH
28302	R	IY	M
28303	R	IY	N
28304	R	IY	NG

8.3.1 개요 - Tri-phone

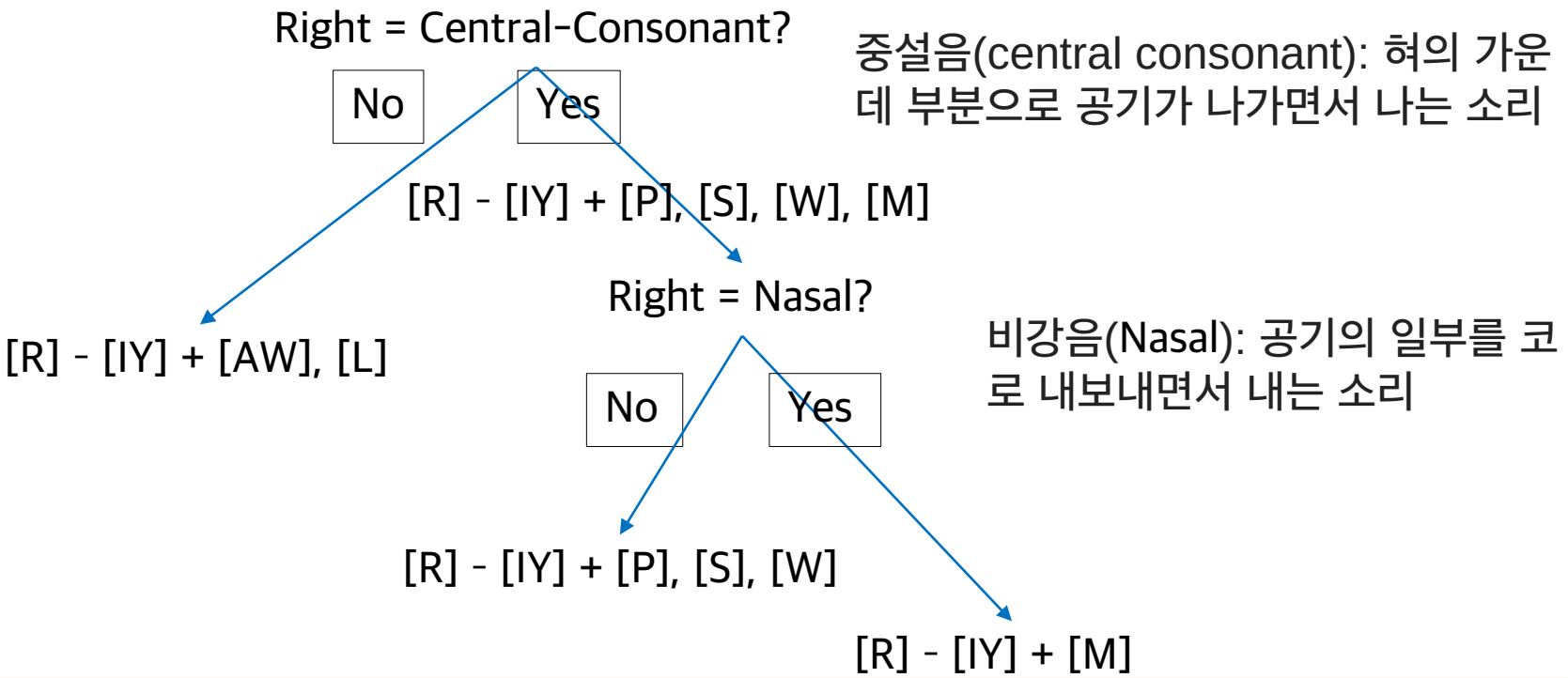
- Corpus에서 실제로 나타는 tri-phone은 약 1만개임
  - ㄱ-ㄱ+ㄱ 등 은 불가능 함
  - 이를 seen tri-phone 이라 함

- Seen tri-phone각각은 3 state-HMM으로 모델링 함
  - 각각의 state에 대한 output probability가 적절히 추정되기 위해서는 학습자료 중 해당 state에 대응되는 vector가 일정량 이상 이어야 함
  - 상당수의 state는 대응되는 vector의 수가 적음
  - 유사한 state를 클러스터링하여 unique한 state의 수를 줄여주어야 함
  - Decision-tree 기반의 state clustering 기법이 많이 사용됨

8.3.1 개요 - Tri-phone

- Decision-Tree based clustering

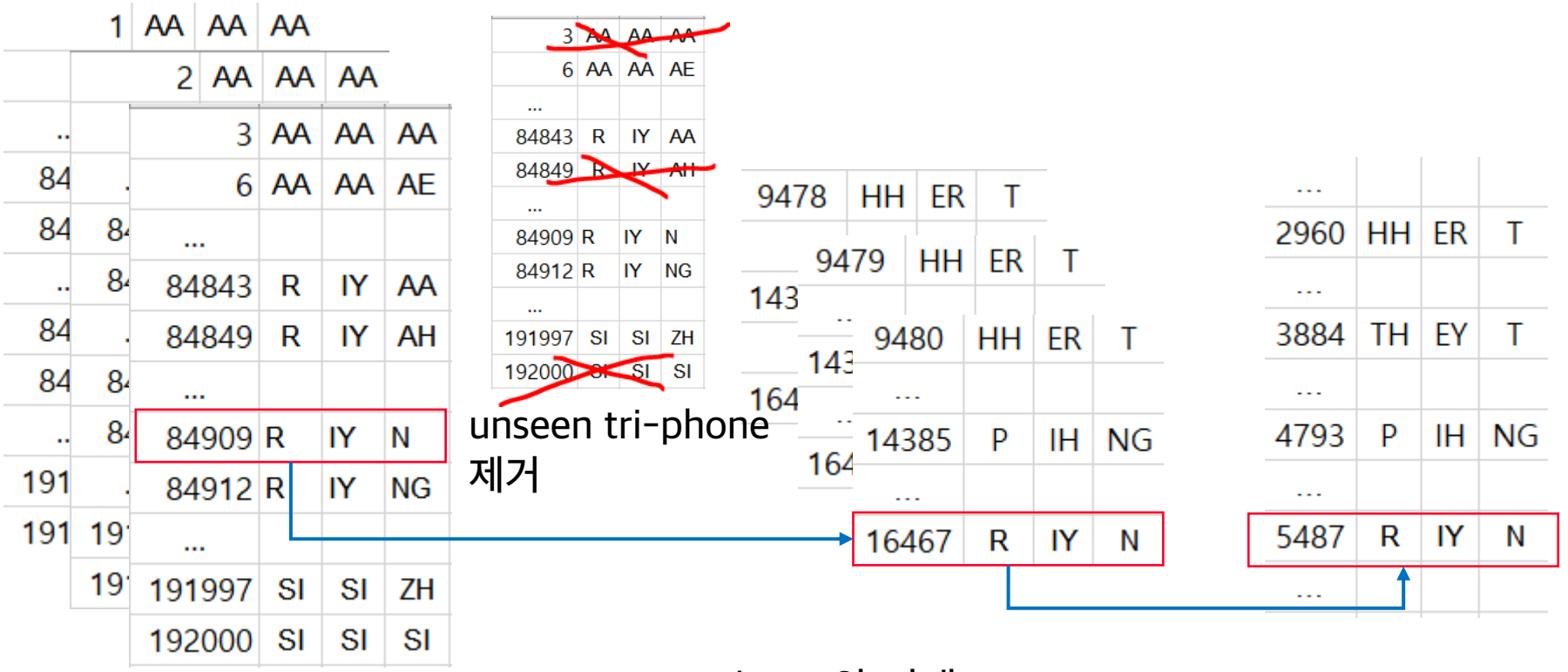
[R] - [IY] + [P] , [R] - [IY] + [S]
[R] - [IY] + [W] , [R] - [IY] + [M]
[R] - [IY] + [AW] , [R] - [IY] + [L]



중설음(central consonant): 혀의 가운데 부분으로 공기가 나가면서 나는 소리

비강음(Nasal): 공기의 일부를 코로 내보내면서 내는 소리

8.3.1 개요 - Tri-phone



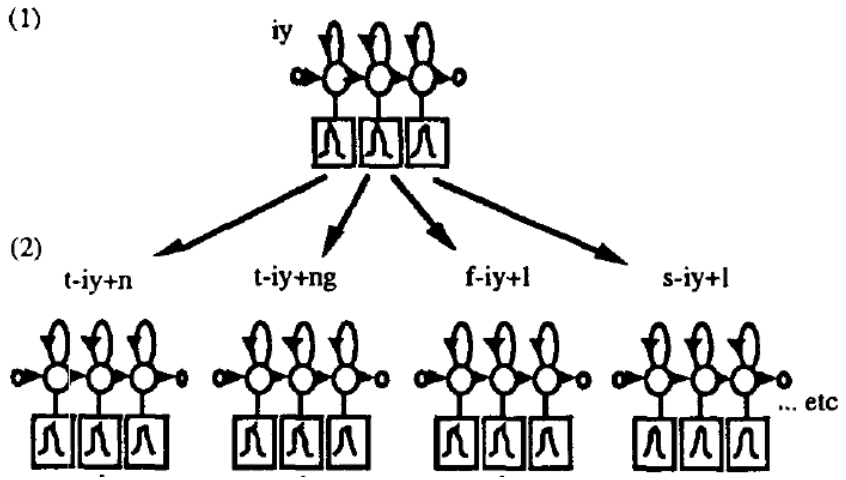
이론상 tri-phone 64000개
각 phone모델 별 3개의 state
64000*3 = 192000 states

Seen tri-phone 약1만개
각 phone모델 별 3개의 state
1만*3 = 3만 states
학습자료가 부족한 state가 많다

States를 Unique한 Cluster로 Mapping
6천~1만개의 clustered states

8.3.1.1 Tree-based state clustering

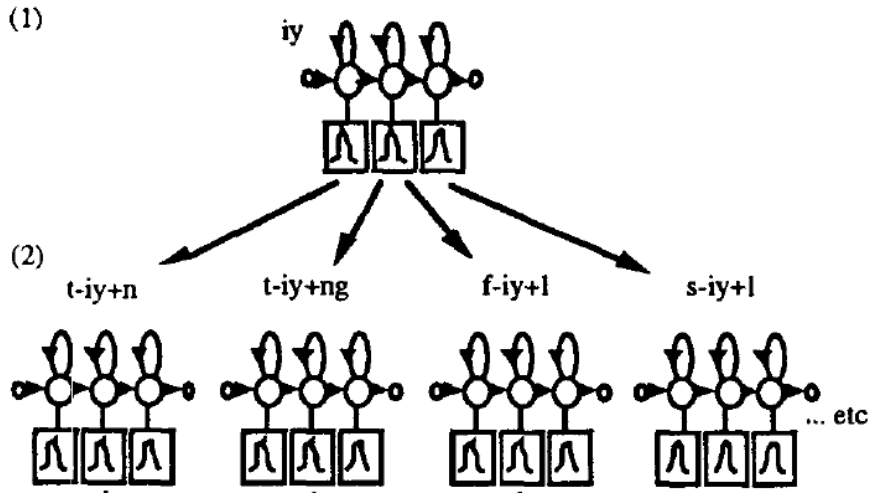
- Tree-based state clustering algorithm (1)
  - 주어진 학습 자료에 대하여 3 state left-right mono-phone model을 구성함
  - Mono-phone model을 training을 거쳐 각 states 마다 single Gaussian output probability density functions을 구함



[young, 1994]

8.3.1.1 Tree-based state clustering

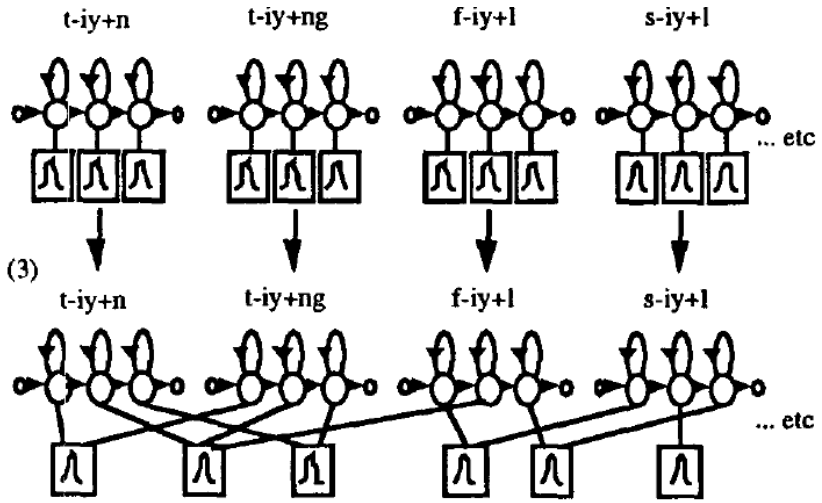
- Tree-based state clustering algorithm (2)
  - 학습된 각 3 state mono-phone 모델을 복제하여 context dependent tri-phone model로 확장함
  - 확장된 tri-phone model을 Viterbi training algorithm (or Baum-Welch)에 의하여 re-estimation을 진행함



[young, 1994]

8.3.1.1 Tree-based state clustering

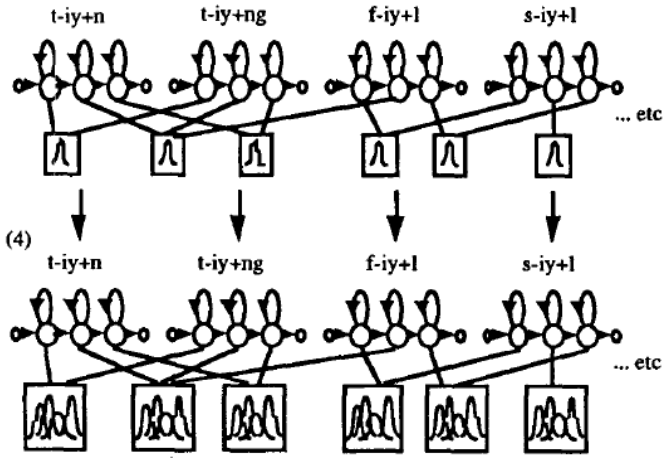
- Tree-based state clustering algorithm (3)
  - 각 tri-phone set들을 decision tree에 의해 clustering을 진행함
  - 생성된 cluster를 기반으로 state tying을 함
  - 하나의 state가 exemplar로 선택되고 다른 모든 cluster member들이 연결됨



[young, 1994]

8.3.1.1 Tree-based state clustering

- Tree-based state clustering algorithm (4)
  - State tying에 따라 각 state들의 mixture들도 incrementation됨
  - State clustering이 끝난 후 해당 모델은 다음 조건을 만족할 때까지 re-estimation을 수행함
    - development test-set에 대하여 성능이 더 이상 올라가지 않는 경우
    - 충분한 mixture 개수에 도달하였을 경우



[young, 1994]

8.3.1.1 Tree-based state clustering

- Decision tree 목적
  - 각 tri-phone set들을 decision tree를 사용하여 state clustering을 수행함
  - Clustering 이후, 각 tied state들의 incremented Gaussian mixture들의 log likelihood를 maximize하는 최적의 decision tree를 구성하는 것이 목표임
    - $L(S)$: log likelihood of S generating the set of training frames F
    - 계산과정에서 Transition probabilities은 생략됨
    - $r_s(o_f)$: a prosteriori probability of the observed frame o_f generated by states s

$$L(\mathbf{S}) = \sum_{f \in F} \sum_{s \in S} \log(Pr(\mathbf{o}_f; \mu(\mathbf{S}), \Sigma(\mathbf{S})) \gamma_s(\mathbf{o}_f)$$

$$L(\mathbf{S}) = -\frac{1}{2}(\log[(2\pi)^n |\Sigma(\mathbf{S})|] + n) \sum_{s \in S} \sum_{f \in F} \gamma_s(\mathbf{o}_f)$$

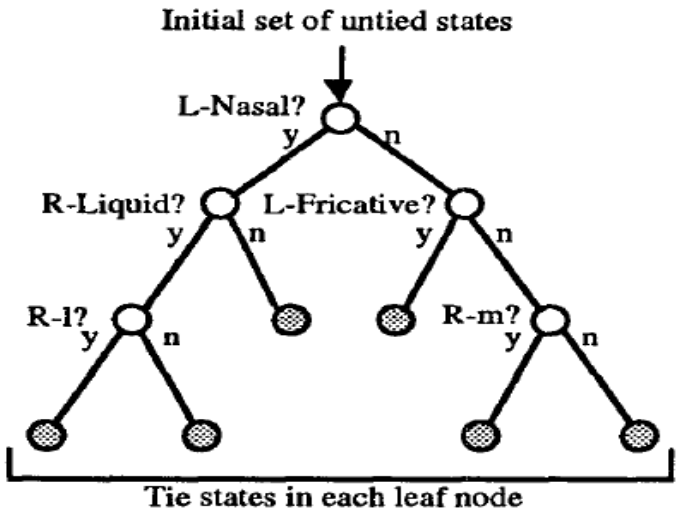
8.3.1.1 Tree-based state clustering

- Decision tree 목적
  - Decision tree의 output states들의 log likelihood를 maximize하기 위해서는 tree의 각 question node의 결정이 중요함
  - State S 가 question node q에 의해서 subset인 $S_y(q)$와 $S_n(q)$으로 나뉨
    - ΔL_q를 maximize하도록 각 question node를 결정함
    - 정해진 threshold 값 이하로 떨어질 때까지 node를 생성하면서 수행함

$$\Delta L_q = L(\mathbf{S}_y(q)) + L(\mathbf{S}_n(q)) - L(\mathbf{S})$$

8.3.1.1 Tree-based state clustering

- Decision tree 구성
  - Top-down 방식으로 phonetic decision tree를 구성함
  - Traditional decision tree 방식에서는 linguistic 기반 question을 사용함
    - Left - Nasal ? (왼쪽에 위치한 phone은 비음인가?)
    - Right - Fricative ? (오른쪽에 위치한 phone은 마찰음인가?)



[young, 1994]

8.3.1.1 Tree-based state clustering

- Decision tree 구성
  - 각 question node마다 가능한 모든 경우의 수의 question에 대하여 log likelihood를 계산 gain을 측정
  - 가장 높은 gain을 가진 question을 각 node에 적용함
    - WSJ 코퍼스에 대한 ranking of useful question 예시

Condition	Question	Total Gain
All states of all models	R-Vowel	25.9
	L-Vowel	23.3
	R-Unrounded	19.7
	L-UnFortisLenis	19.5
	R-UnFortisLenis	18.3
	R-r	17.1
Entry state of all models	L-UnFortisLenis	18.3
	L-Vowel	16.9
	L-Nasal	10.3
	L-CentralFront	7.7
	L-Unrounded	7.4
	L-Fortis	6.2
Exit state of all consonants	R-Vowel	15.2
	R-Unrounded	8.6
	R-High	4.7
	R-ee	3.9
	R-Rounded	3.7
	R-Syllabic	3.6

[young, 1994]

8.3.1.1 Tree-based state clustering

- Toy problem

Input data

	skin	color	size	flesh	class
1	hairy	brown	large	hard	safe
2	hairy	green	large	hard	safe
3	smooth	red	large	soft	dangerous
4	hairy	green	large	soft	safe
5	hairy	red	small	hard	safe
6	smooth	red	small	hard	safe
7	smooth	brown	small	hard	safe
8	hairy	green	small	soft	dangerous
9	smooth	green	small	hard	dangerous
10	hairy	red	large	hard	safe
11	smooth	brown	large	soft	safe
12	smooth	green	small	soft	dangerous
13	hairy	red	small	soft	safe
14	smooth	red	large	hard	dangerous
15	smooth	red	small	hard	safe
16	hairy	green	small	hard	dangerous

Query: hairy skin, red color, large with soft flesh.
The question is: is it safe to eat this unknown animal?

[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

- Toy problem

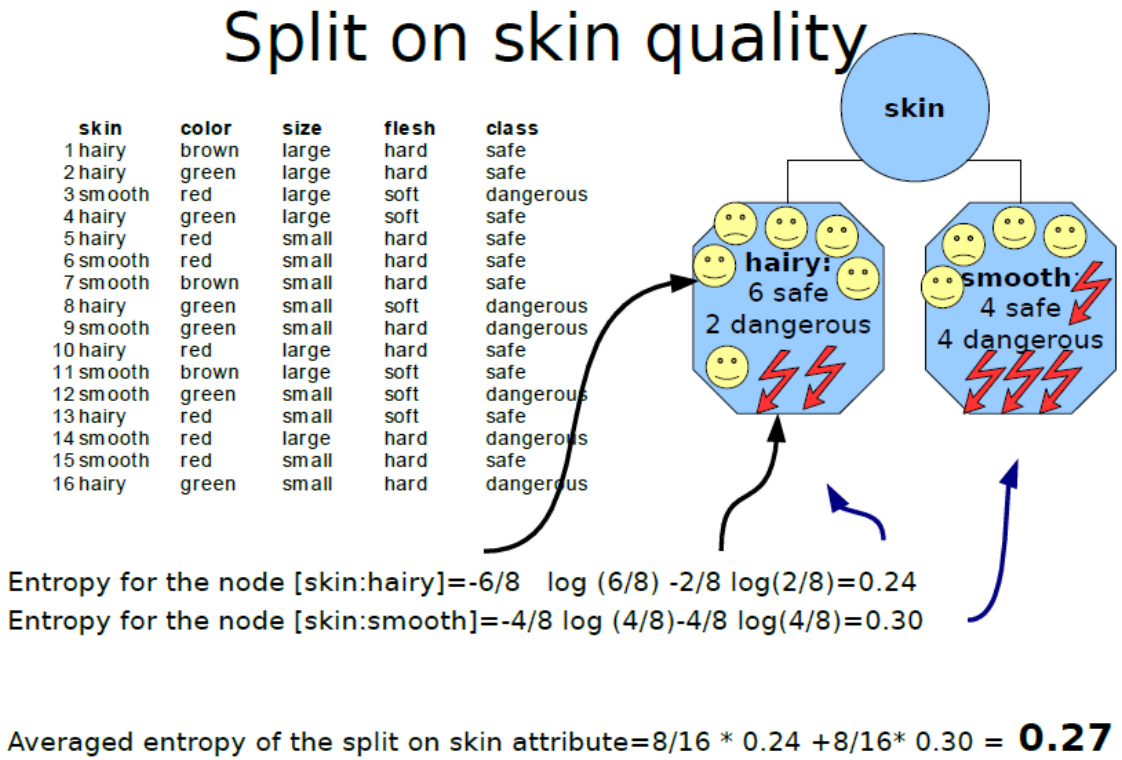
Decision tree classifier

- We can split the data based on any given nominal (non-numeric) attribute
- We compute the entropy (purity) of the tree nodes after the split and we choose the split wich gives the smallest average entropy for all the tree nodes created after the split

[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

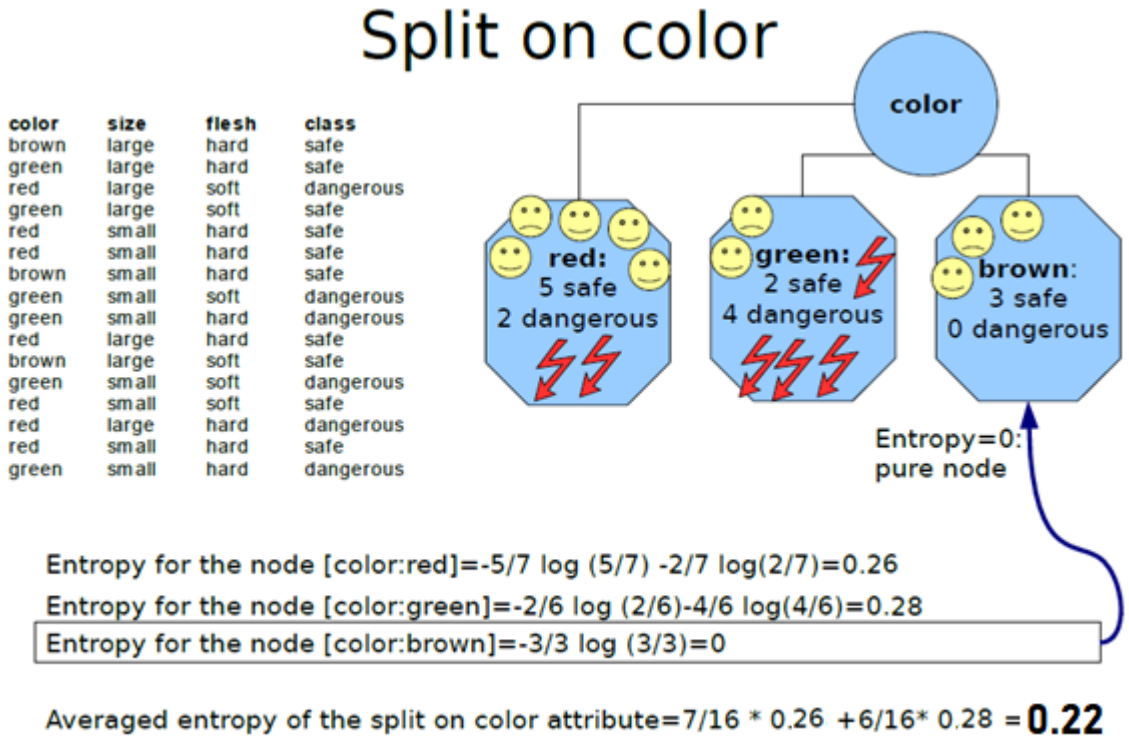
- Toy problem



[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

- Toy problem



color	size	flesh	class
brown	large	hard	safe
green	large	hard	safe
red	large	soft	dangerous
green	large	soft	safe
red	small	hard	safe
red	small	hard	safe
brown	small	hard	safe
green	small	soft	dangerous
green	small	hard	dangerous
red	large	hard	safe
brown	large	soft	safe
green	small	soft	dangerous
red	small	soft	safe
red	large	hard	dangerous
red	small	hard	safe
green	small	hard	dangerous

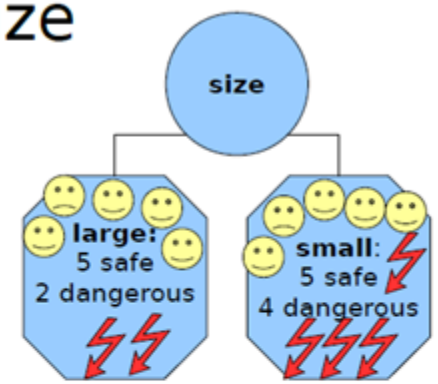
[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

- Toy problem

Split on size

size	flesh	class
large	hard	safe
large	hard	safe
large	soft	dangerous
large	soft	safe
small	hard	safe
small	hard	safe
small	hard	safe
small	soft	dangerous
small	hard	dangerous
large	hard	safe
large	soft	safe
small	soft	dangerous
small	soft	safe
large	hard	dangerous
small	hard	safe
small	hard	dangerous



Entropy for the node [size:large]=-5/7 log (5/7) -2/7 log(2/7)=0.26
Entropy for the node [size:small]=-5/9 log (5/9)-4/9 log(4/9)=0.30

Averaged entropy of the split on size=7/16 * 0.26 +9/16* 0.30 = **0.28**

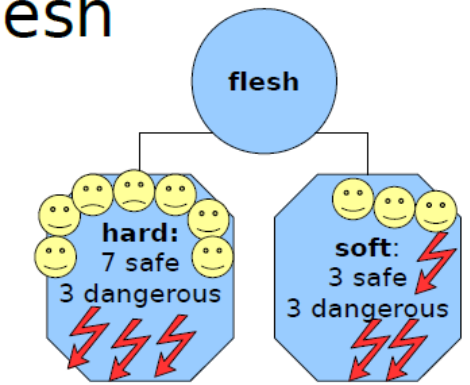
[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

- Toy problem

Split on flesh

flesh	class
hard	safe
hard	safe
soft	dangerous
soft	safe
hard	safe
hard	safe
hard	safe
soft	dangerous
hard	dangerous
hard	safe
soft	safe
soft	dangerous
soft	safe
hard	dangerous
hard	safe



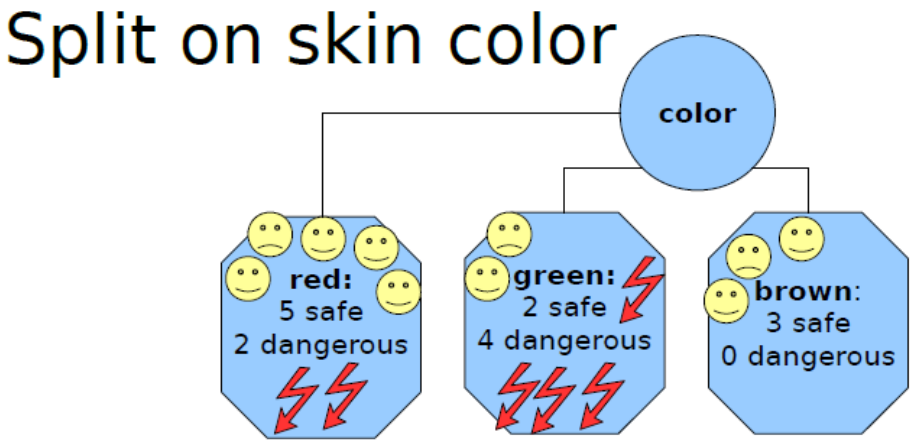
Entropy for the node [flesh:hard]=-7/10 log (7/10) -3/10 log(3/10)=0.27
Entropy for the node [flesh:soft]=-3/6 log (3/6)-3/6 log(3/6)=0.33

Averaged entropy of the split on flesh=10/16 * 0.27 +6/16* 0.33 = **0.29**

[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

- Toy problem



The best split: the smallest entropy of the nodes after the split.
We choose this split to be a root node of the decision tree

[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

- Toy problem

Splitting node: color=red

color=red

	skin	size	flesh	class
3	smooth	large	soft	dangerous
5	hairy	small	hard	safe
6	smooth	small	hard	safe
10	hairy	large	hard	safe
13	hairy	small	soft	safe
14	smooth	large	hard	dangerous
15	smooth	small	hard	safe

Entropy of the node [skin=smooth]=-2/4 log 2/4 – 2/4 log 2/4=0.30
Entropy of the node [skin=hairy]=-3/3 log 3/3 =0
Averaged entropy of the split on skin=4/7 *0.30 = **0.17**

Entropy of the node [size=large]=-1/3 log 1/3 – 2/3 log 2/3=0.28
Entropy of the node [size=small]=-4/4 log 4/4 =0
Averaged entropy of the split on size=3/7 * 0.28 = **0.12**

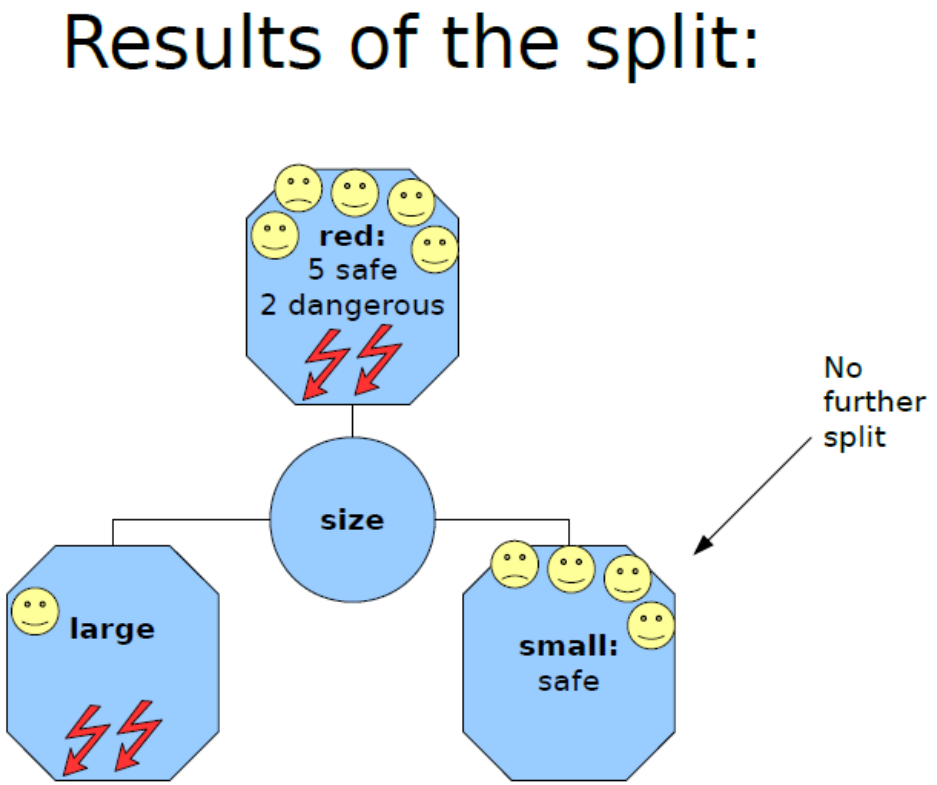
The best

Entropy of the node [flesh=soft]=-1/2 log 1/2 – 1/2 log 1/2=0.30
Entropy of the node [flesh=hard]=-4/5 log 4/5 – 1/5 log 1/5=0.22
Averaged entropy of the split on flesh=2/7 *0.30 +5/7*0.22 = **0.24**

[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

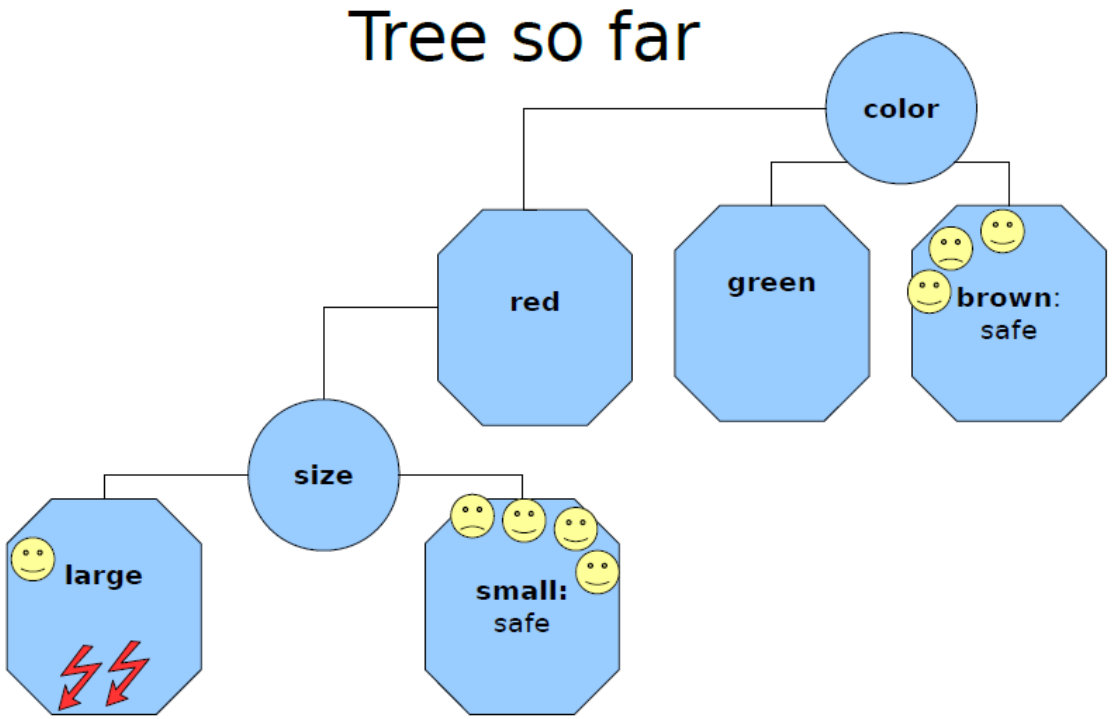
- Toy problem



[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

- Toy problem



[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

- Toy problem

Splitting node: color=green

color=green

skin	size	flesh	class
2 hairy	large	hard	safe
4 hairy	large	soft	safe
8 hairy	small	soft	dangerous
9 smooth	small	hard	dangerous
12 smooth	small	soft	dangerous
16 hairy	small	hard	dangerous

Entropy of the node [skin=smooth]=-2/2 log 2/2 =0
Entropy of the node [skin=hairy]=-2/4 log 2/4-2/4 log 2/4 =0.30
Averaged entropy of the split on skin=4/6 *0.30 = **0.20**

Entropy of the node [size=large]=-2/2 log 2/2 =0
Entropy of the node [size=small]=-4/4 log 4/4 =0
Averaged entropy of the split on size=**0.00**

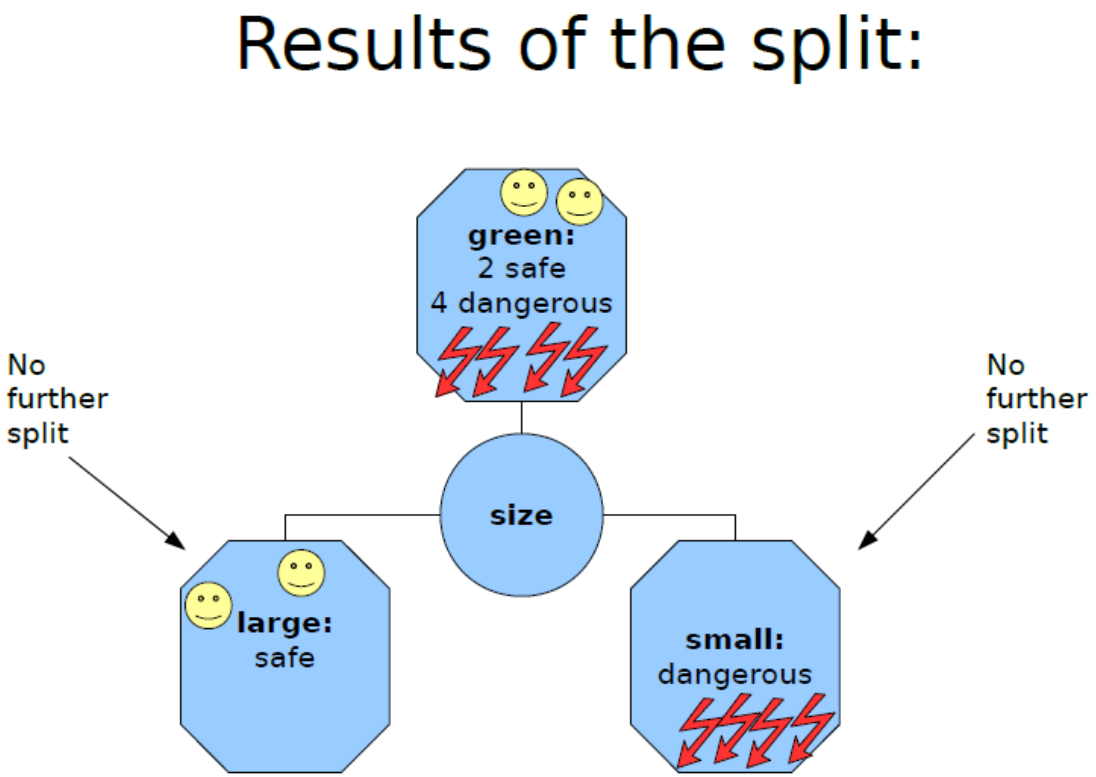
The best

Entropy of the node [flesh=soft]=-1/3 log 1/3 – 2/3 log 2/3=0.28
Entropy of the node [flesh=hard]=-1/3 log 1/3 – 2/3 log 2/3=0.28
Averaged entropy of the split on flesh=3/6 *0.28 +3/6*0.28 = **0.28**

[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

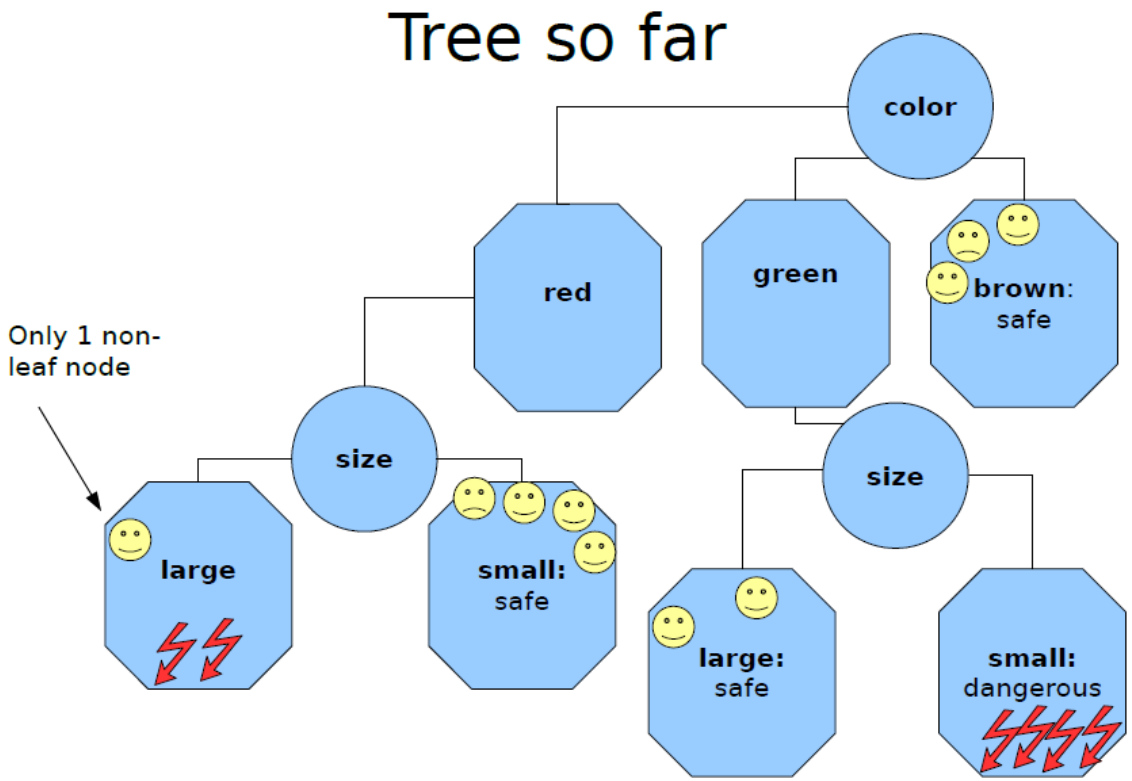
- Toy problem



[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

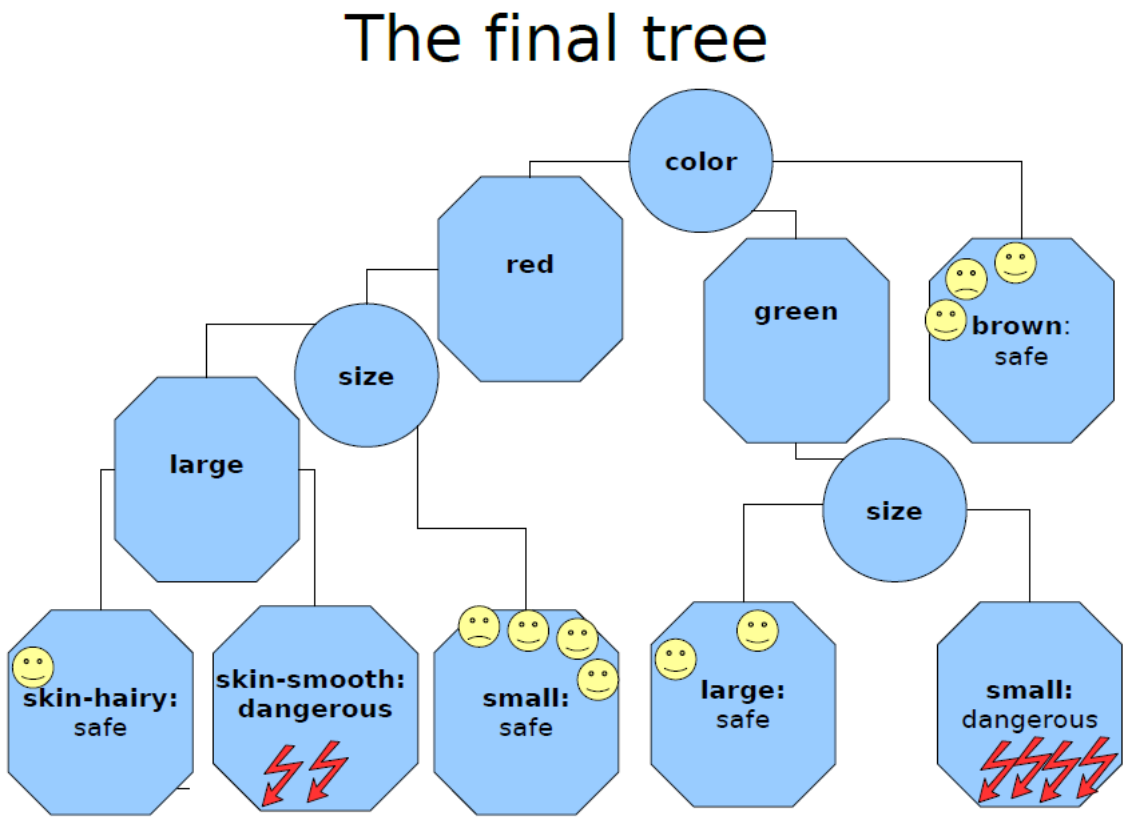
- Toy problem



[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

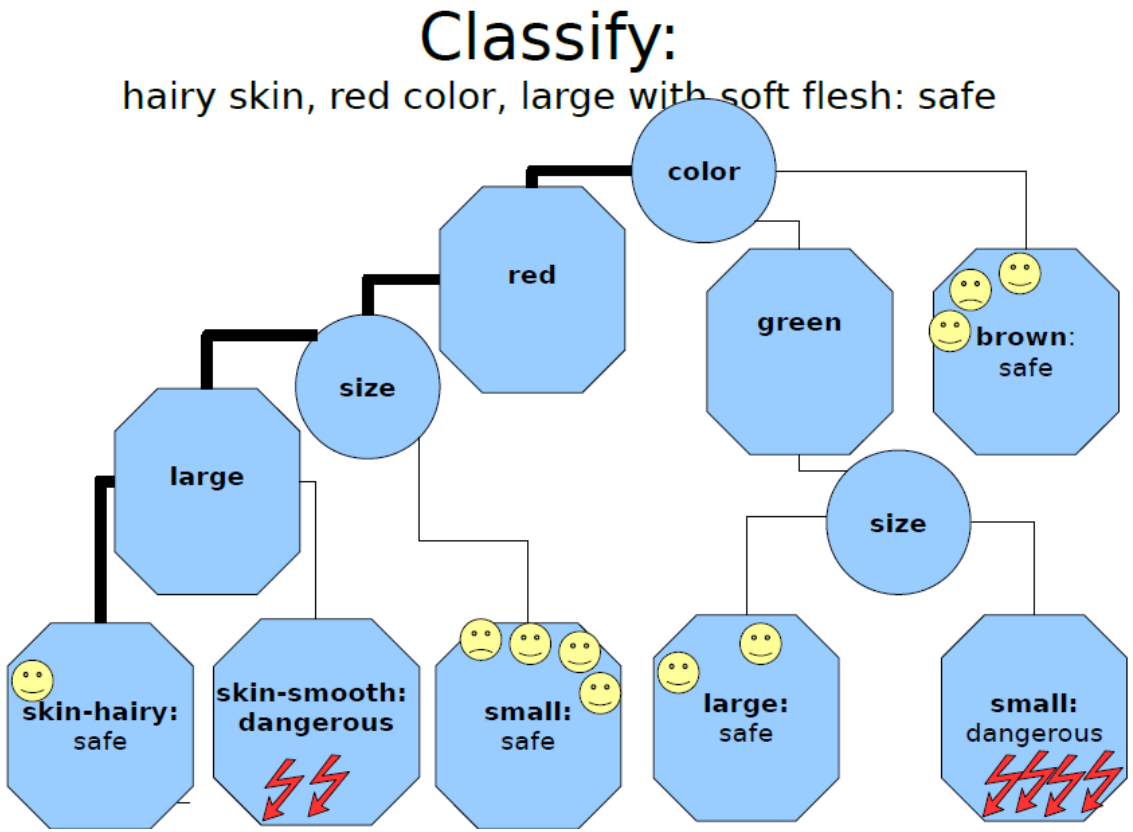
- Toy problem



[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

- Toy problem



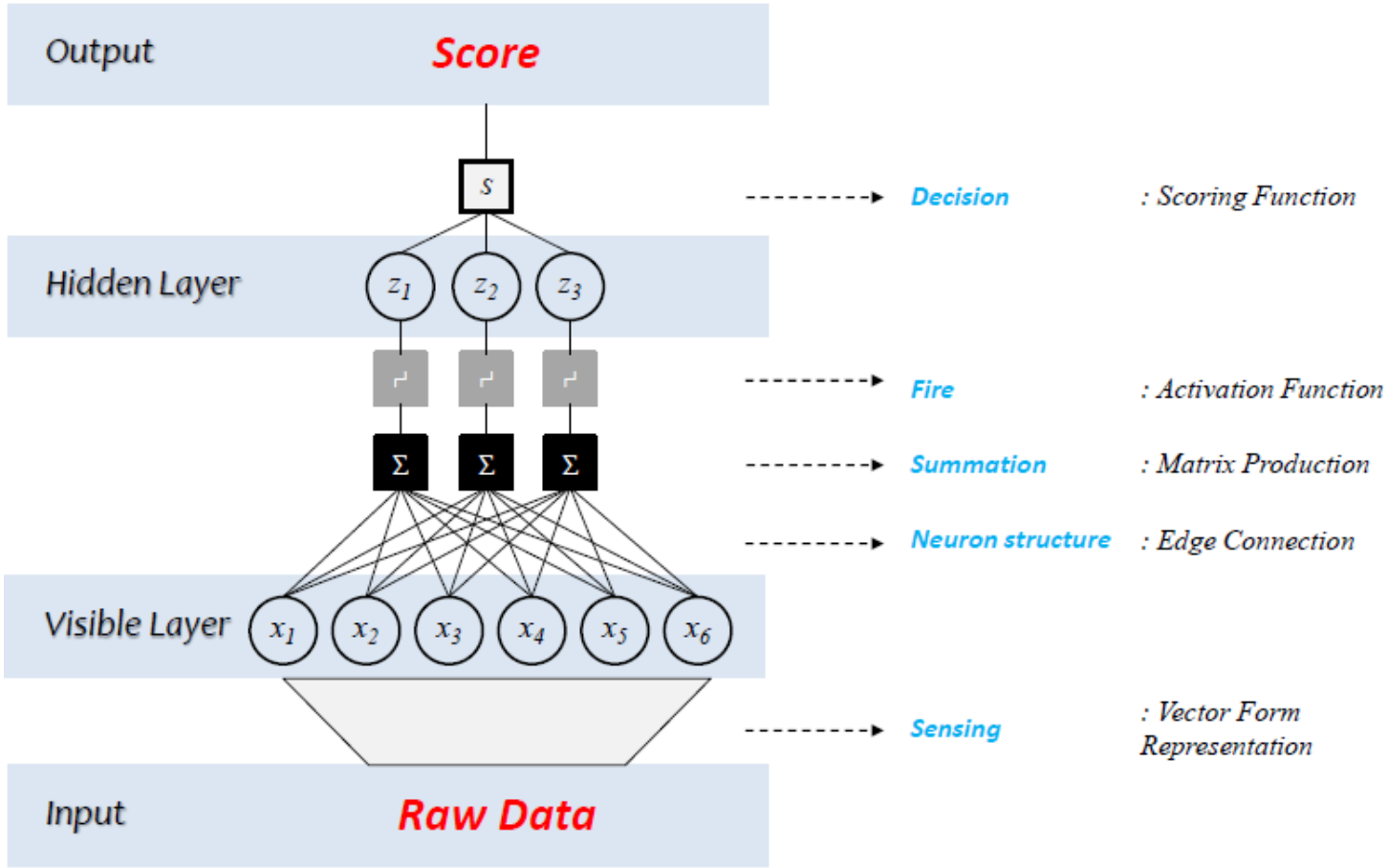
[http://csci.viu.ca/~barskym/teaching/DM_LABS/LAB_3/Lab3_decisiontreeexample.pdf]

8.3.1.1 Tree-based state clustering

- DNN output layer node
  - Decision tree를 통해서 clustering이 된 states들을 DNN 학습 시 output layer node로 사용함
    - 192,000개의 3 state tri-phone
    - 약 10,000개의 seen tri-phone으로 줄임
      - 10000 * 3 = 30000 states
    - 약 6,000 ~ 10,000개의 clustered states

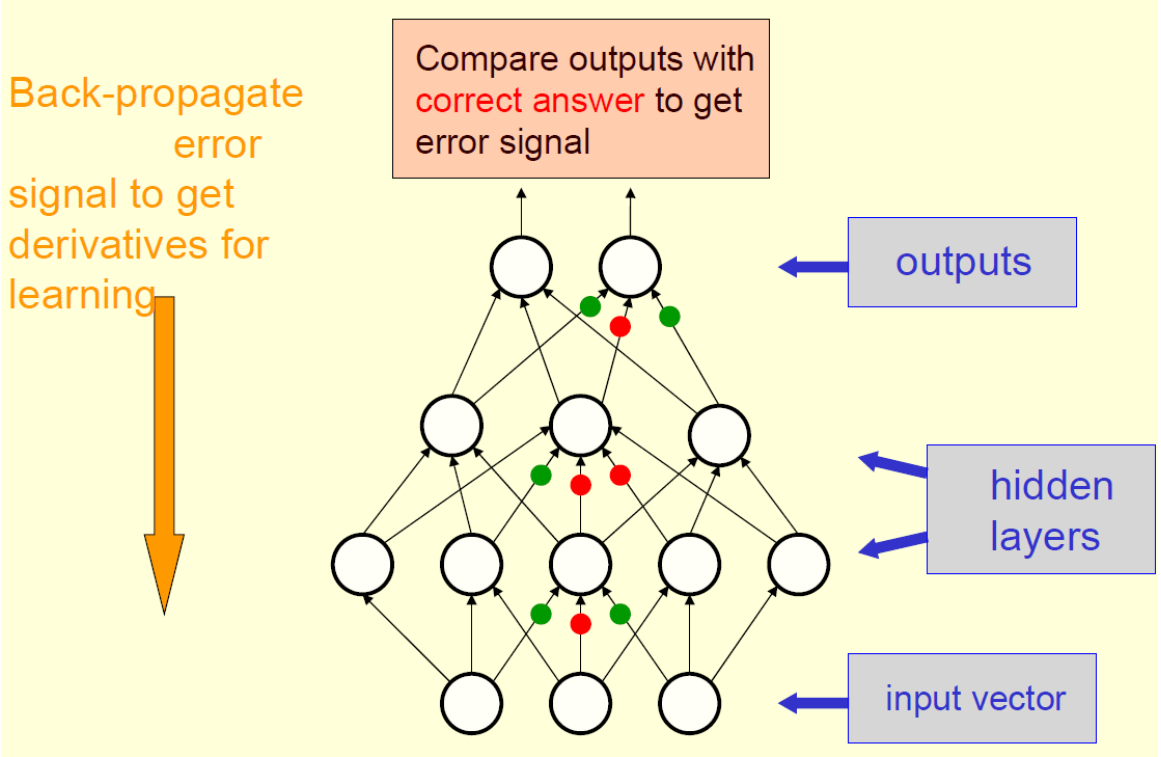
8.3.2 Deep Neural Network

- DNN은 input layer, hidden layer, output layer로 구성



8.3.2 Deep Neural Network

- Hidden layer의 개수가 2 이상인 모델을 DNN이라 함
  - If # of hidden layers <=1 → Shallow neural network
  - If # of hidden layers >=2 → Deep Neural Network (DNN)

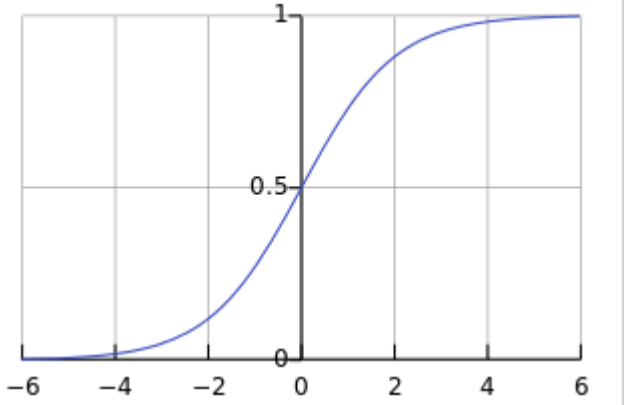


[Hinton, 2013]

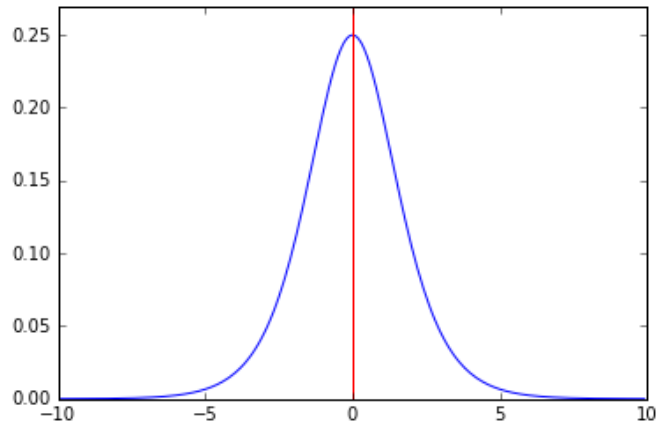
8.3.2.1 DNN 학습에서 생기는 문제점

- Vanishing gradient problem
  - Non-linear activation 함수 이용 시 문제가 발생
    - Gradient가 층을 하나씩 내려가면서 error가 희석되는(작아지는) 문제가 발생
      - 시그모이드 함수의 경우, 값이 급격히 변하는 구간을 제외하면, 미분 값 (경사도)는 0에 가까움

$$f(k) = \frac{1}{1 + e^{-k}}$$

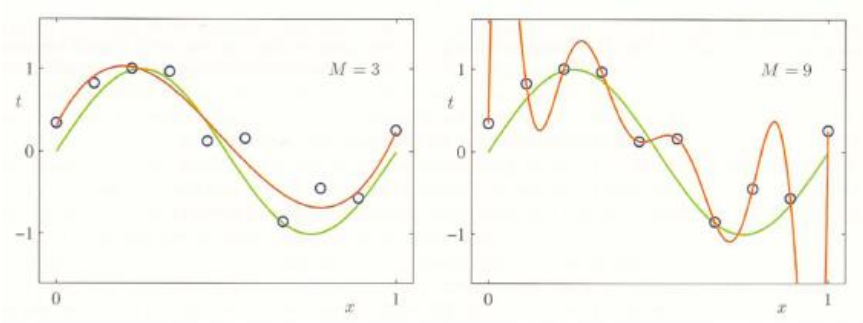


$f'(k)$



8.3.2.2 Error Back-propagation 문제점

- Typically requires lots of labeled data
  - 데이터 수집의 어려움 (비용, 시간)
- Overfitting problem
  - 학습자료가 부족한 경우, 부족한 학습자료에 대해서만 parameter 값이 overfitting 되어 테스트 데이터에 대해서 인식율이 타 모델 보다 낮게 나오는 문제가 발생함



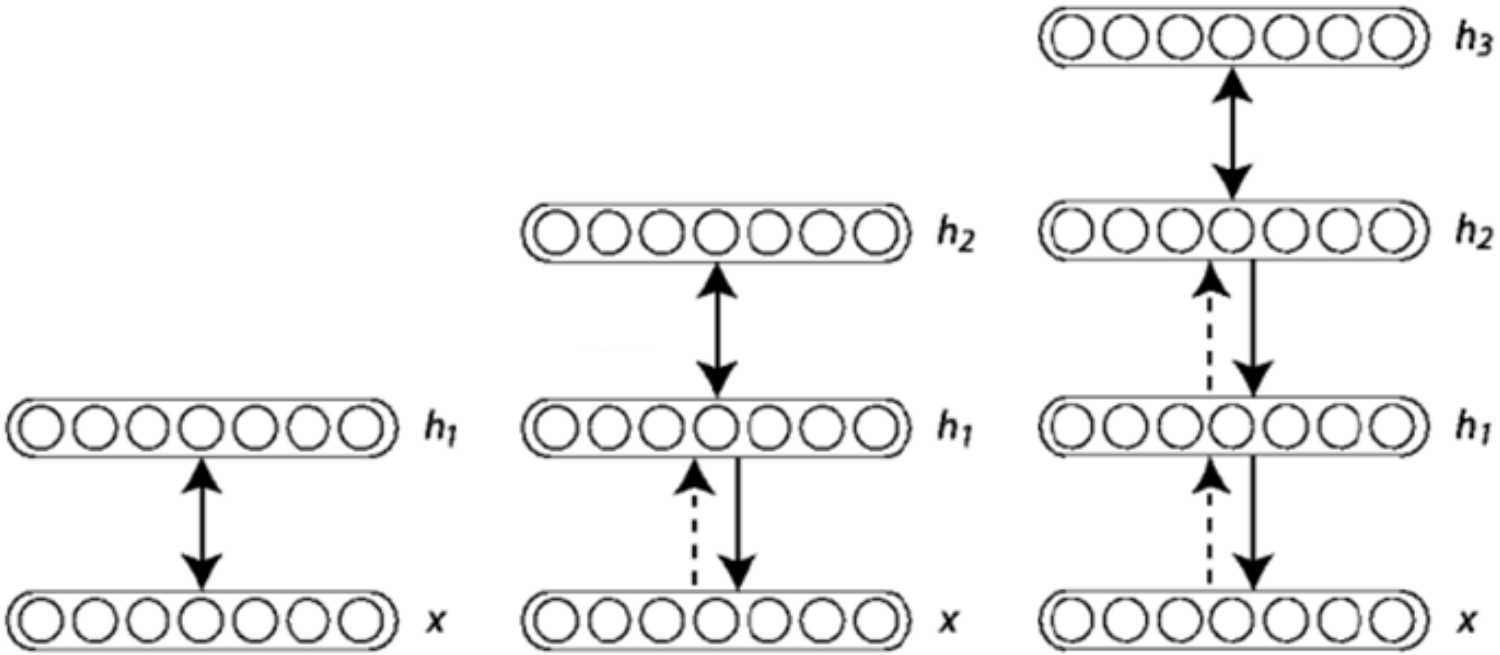
- Get stuck in local minima
  - Random 초기화와 같이 ‘good’ regions으로 부터 너무 멀리서 학습이 진행되는 경우 발생

8.3.2.3 DNN 학습에서 생기는 문제점 해결방안

- Vanishing gradient problem
  - ReLU(Rectified Linear Unit), Layer-wise training
- Typically requires lots of labeled data
- Overfitting problem
- Get stuck in local minima
  - 데이터 양이 증가하고 computation power가 증가함에 따라 해결가능.

8.3.3 DNN을 이용한 음향모델 학습

- Layer-wise training [Hinton, 2006]



8.3.3 DNN을 이용한 음향모델 학습

- 대용량 데이터 및 이를 사용할 수 있는 computing power가 제공되어 back-propagation만으로 충분히 모델 학습이 가능해짐 [Hinton, 2013]