산업기술연구(강원대학교 산업기술연구소 논문집), 제26권 B호, 2006.
Journal of Industrial Technology, Kangwon Natl. Univ., Korea, No. 26 B, 2006.

다중 Stream 구조를 가지는 VQ를 이용하여 연산량을 개선한 CHMM에 관한 연구

A Study of CHMM Reducing Computational Load Using VQ with Multiple Streams

방 영 규* 정 익 주**
Bang, Young Gue Chung, IK Joo

Abstract

Continuous, discrete and semi-continuous HMM systems are used for the speech recognition. Discrete systems have the advantage of low run-time computation. However, vector quantization reduces accuracy and this can lead to poor performance. Continuous systems let us get good correctness but they need much calculation so that occasionally they are unable to be used for practice. Although there are semi-continuous systems which apply advantage of continuous and discrete systems, they also require much computation. In this paper, we proposed the way which reduces calculation for continuous systems. The proposed method has the same computational load as discrete systems but can give better recognition accuracy than discrete systems.

키워드 : *음성인식, CHMM, 연산량 감소*
keywords: *Speech recognition, reduction of computation*

1. 서 론

현재의 음성인식 기법은 통계적 특징을 이용한 방법이 그 대부분을 차지하고 있다. 특히 HMM(hidden markov model)은 확률적인 접근 방법의 일종으로 음성과 같은 시간적 변이의 특성에 대해 좋은 성능을 보이고 있어 고립단어 인식에서 연속음성 인식에 이르기까지 가장 널리 이용되고 있는 방법 중에 하나이다[1][2][3][7][8]. HMM은 확률분포를 추정하는 방법에 따라 continuous HMM, semi-continuous HMM, discrete HMM으로 나누어진다. 가장 높은 인식률을 얻을 수 있는 방법은 continuous HMM으로 설계하는 것이다. 그러나 continuous HMM은 높은 인식률에도 불구하고 방대한 연산량으로 인하여 그 사용에 있어 많은 제한이 있어 왔다. 본 논문에서는 continuous HMM의 연산량을 줄일 수 있는 방법으로 continuous HMM의 입력으로 VQ(vector quantization)된 data를 사용하는 방안을 제안하였다. 입력 data를 VQ하면 유한개의 예측 가능한 data가 되므로 예상되어지는 모든 data에 대하여 미리 output probability를 계산하여 메모리에 저장한 후 활용할 수 있다. 따라서 차후 인식 과정에서 요하는 output probability의 연산을 줄일 수 있어서 전체 인식기의 연산량을 감소시키는데 효과를 얻을 수 있다. VQ된 입력을 사용하는 continuous HMM은 인식과정에서 discrete HMM과 동일한 구조를 가지므로 인식과정에서 요구되는 연산량과 메모리는 discrete HMM과 동일하게 된다. 그러나 입력 data를 VQ함으로 발생하는 model mismatch로 인해 기존의 continuous HMM에 비하여 인식률이 저하되고, 인식에 사용되는 입력뿐만 아니라,

* 강원대학교 전자공학과 석사과정
** 강원대학교 전자공학과 교수

VQ된 data를 사용하여 훈련을 하는 discrete HMM에 비해서도 인식률이 감소된다. 이와 같은 VQ로 인한 인식률 감소의 문제를 보완하기 위해서 stream수 증가, full covariance matrix 사용, noise가 추가된 data를 이용한 훈련의 방안을 제안하였다. 본 논문에서 제안하는 방법을 통하여 컴퓨터 환경뿐만 아니라 embedded 환경에서도 음성인식 기술을 적용하는데 도움이 될 것이라 기대한다.

본 논문의 구성은 다음과 같다. 2장에서 음성인식에 사용되어지는 HMM에 따라 필요로 하는 연산과 연산량에 대하여 기술하였다. 3장에서는 본 논문이 제시하는 연산량을 줄일 수 있는 방안에 대하여 논함과 동시에 논문이 제시한 방법과 기존의 3가지 HMM의 연산량을 비교함으로써 본 논문의 유익성을 설명하였다. 4장에서는 본 논문이 제안한 방법으로 실제 인식 test를 하였을 경우 얻어지는 결과를 분석하였으며 5장에서는 본 논문의 결론과 향후 연구방향에 대하여 기술하였다.

2. 음성인식에서 수행되는 Processing

음성인식에서 사용되어지는 HMM은 output probability를 어떻게 추정하는가에 따라 discrete, semi-continuous, continuous HMM으로 나누어 볼 수 있다[4][5][6]. discrete HMM은 상태에서의 관측 출력 확률이 이산적인 특징 값으로 추정되는 것을 말한다. discrete HMM은 연속적인 음성 data를 이산 data로 만들기 위하여 VQ과정을 거치게 되는데 이 과정에서 VQ 오차(quantization error)가 발생하여 인식률 저하의 원인이 된다[1]. 그러나 discrete HMM으로 인식기를 설계할 경우 이산 data를 사용하기 때문에 연산량 감소의 효과를 얻을 수 있다. data를 통계학적인 분포를 가지는 연속적인 값이라고 간주하여 연속확률 밀도 함수(probability density function)를 data 값으로 사용하는 HMM을 continuous HMM이라고 한다. continuous HMM의 경우 discrete HMM과는 달리 음성신호에 대한 VQ 과정 없이 LPC-Cepstrum이나 Mel-Cepstrum을 통하여 얻은 feature vector를 그대로 사용할 수 있다. 이러한 이유로 인하여 continuous HMM을 이용한 음성인식의 경우 VQ로 인한 손실이 없기 때문에 discrete HMM에 비하여 높은 인식률을 얻을 수 있다[1]. 반면 연속적인 값을 사용하기 때문에 연산량이 많은 단점이 있다. semi-continuous HMM은 discrete HMM의 이산 data와 continuous HMM의 연속밀도함수를 혼합한 경우로서 두 HMM의 장점을 취한 형태로 볼 수 있다.

위의 3가지의 HMM 모두 인식 단계에서 대표적인 알고리즘인 viterbi연산을 수행한다[2][3]. viterbi 연산 과정 중에 각 state에서 발생하는 output probability를 계산하여 transition probability와 더해주는 부분이 반복적으로 수행된다.

$$\tilde{\delta}_t(j) = \log(\delta_t(j)) = \max_{1 \le i \le N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}] + \tilde{b}_j(o_t)$$

$$\psi_t(j) = \arg \max_{1 \le i \le N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}], \quad (1)$$

$$2 \le t \le T, 1 \le j \le N$$

이 반복적인 연산 과정에서 각 HMM이 서로 다른 연산을 하는 부분이 $b_j(o_t)$, 즉 state의 output probability를 계산하는 부분이다. 이 output probability를 계산하기 위하여 어떤 종류의 data를 쓰느냐에 따라 위에서 언급한 3가지의 HMM으로 나뉜다. 그리고 각 HMM에서 요구하는 연산량의 차이가 output probability를 연산하는 방법에 따라 발생한다.

다음은 각 HMM의 output probability를 연산하는 부분이다.

먼저 continuous HMM은

$$b_j(O_t) = \prod_{s=1}^{S} \left[\sum_{m=1}^{M_s} c_{jsm} \frac{1}{\sqrt{(2\pi)^n \sum_{jsm}}} e^{-\frac{1}{2}(o_{st}-\mu_{jsm})' \Sigma_{jsm}^{-1} (o_{st}-\mu_{jsm})} \right] \quad (2)$$

와 같은 연산을 하게 된다. 여기서 j는 state, s는 stream, m은 각 stream에 편성된 mixture를 나타낸다. μ은 평균, Σ은 분산을 의미한다. 그러나 이와 같은 연산은 1보다 작은 숫자를 반복적으로 곱할 경우 numerical problem을 발생시킬 수 있으므로 log 연산으로 대체하여 연산을 수행한다[3]. 보통 $(2\pi)^n |\Sigma_{jsm}|$ 은 training 과정에서 미리 계산하여 저장한 후 인식과정에서 읽어 들여 사용한다.

discrete HMM은 다음과 같은 연산을 수행한다.

$$b_j(o_t) = \prod_{s=1}^{S} P_{js}[v_s(o_{st})] \quad (3)$$

$v_s(o_{st})$는 input vector o_{st}에 대한 vector quantizer의 출력을 의미한다.
discrete HMM 역시 log 연산을 수행하여 numerical problem을 해결한다.

semi-continuous HMM의 output probability를 얻기 위해 다음과 같은 연산이 이루어진다.

$$b_j(o_t) = \prod_{s=1}^{S}\left[\sum_{m=1}^{M_s} c_{jsm} N(o_{st}; \mu_{sm}, \Sigma_{sm})\right] \quad (4)$$

$$b_j(o_t) = \prod_{s=1}^{S}\left[\sum_{m=1}^{M_s} c_{jsm} \frac{1}{\sqrt{(2\pi)^n \Sigma_{sm}}} e^{-\frac{1}{2}(o_{st}-\mu_{sm})' \Sigma_{sm}^{-1}(o_{st}-\mu_{sm})}\right]$$

m은 각 stream에 할당된 codeword의 개수, 즉 codebook의 크기이다. μ_{sm}과 Σ_{sm}은 각각 s 번째 stream에 해당하는 m 번째 codeword의 평균 벡터와 분산을 의미한다. continuous HMM과 다른 점은 각 state의 Gaussian component parameter들과 mixture component들이 state에 독립적이라는 것이다[1]. 그렇기 때문에 continuous HMM보다 연산량이 적다. semi-continuous HMM 역시 log 연산으로 대신한다.

이렇게 output probability를 연산하는 방법은 각 HMM의 연산량 차이를 가져오게 된다.
continuous HMM과 semi-continuous HMM의 경우 output probability의 입력으로 연속적인 data 값이 사용되어 지기 때문에 그에 따라 연산량이 증가한다. 그러나 discrete HMM의 경우 입력 data는 codebook에 의해 이산화된 값이 사용되기 때문에 유한입력에 대한 연산을 수행하므로 연산량이 적다. 이러한 특성에 기반을 두고 continuous HMM에 유한입력(vector quantized input data)을 사용함으로써 연산량을 줄일 수 있는 방법을 유도할 수 있다. 3장에서 구체적으로 그 방안에 대하여 설명하였다.

3. VQ를 이용한 CHMM의 연산량 개선의 방법과 연산량 비교

3.1 VQ를 이용한 CHMM의 연산량 개선의 방법

output probability의 반복적인 연산을 피할 수 있는 방법은 discrete HMM과 같이 output probability를 매번 계산하지 않고 미리 계산되어진 값을 메모리로 부터 읽어서 사용하는 것이다. discrete HMM의 입력 data는 output probability를 계산하기 이전에 VQ를 통하여 codebook size 만큼의 대표되는 값들로 바뀌기 때문에 codebook size에 해당하는 유한개의 값을 갖는다. 이러한 특성으로 인해 discrete HMM은 output probability를 미리 계산하는 것이 가능하다. 그러나 continuous HMM의 경우 입력 값으로 무한개의 연속적인 값을 사용하기 때문에 입력으로 예측되어지는 모든 값들에 대하여 미리 계산하여 메모리에 저장하기란 사실상 불가능하다. semi-continuous HMM 역시 입력 값이 continuous HMM과 같이 무한의 연속적인 data를 사용하므로 어떤 값의 data가 입력될지 예측 할 수 없기 때문에 모든 입력에 대한 output probability를 미리 계산할 수 없다. continuous HMM과 semi-continuous HMM의 무한의 입력을 discrete HMM과 같이 미리 정해진 유한개의 입력으로 바꿀 수 있다면 입력되어지는 값을 미리 정해진 data 값들 중에 하나로 가정할 수 있고 그에 따른 output probability를 미리 계산할 수 있다. 무한의 연속적인 입력 data를 유한개의 data로 바꾸는 것은 VQ를 통해 가능하다. 입력 data가 VQ 되면 VQ된 data와 훈련으로 얻은 HMM을 이용하여 인식 단계 이전에 미리 output probability를 계산하여 메모리에 table로 저장할 수 있고, 저장된 output probability를 인식 단계에서 viterbi algorithm이 수행될 때 메모리로부터 계산 과정 없이 읽어서 사용할 수 있다. 따라서 discrete HMM과 동일한 연산량으로 viterbi algorithm을 수행할 수 있다. VQ된 입력을 가지는 continuous HMM이 동일한 입력을 사용하는 discrete HMM과 다른 점은 VQ 오차로 인한 인식률 저하를 다수의 mixture를 통하여 얻어진 HMM을 사용함으로써 보완 할 수 있다는 점이다. 또한 VQ된 입력을 가지는 continuous HMM은 VQ 오차로 인한 인식률 저하를 줄이기 위하여 full covariance matrix를 활용할 수 있어서 discrete HMM에 비하여 더 높은 인식률을 얻을 수 있다. VQ된 입력을 가지는 continuous HMM의 또 다른 장점은 continuous HMM으로 얻은 HMM을 그대로 사용하면서 입력 data에 대해서만 VQ를 하여 음성 인식기를 구현할 수 있기 때문에 VQ된 입력을 가지는 continuous HMM을 사용하기 위해서 새롭게 training을 실시할 필요가 없다는 것이다. 따라서 computing 환경에 따라 continuous HMM과 논문에서 제안한 방법을 선택하여 사용하는 것이 가능하다. 본 논문에서는 continuous HMM이 semi-continuous HMM보다 더 높은 인식률을 얻을 수 있다는 것이 일반적으로 증명되었기 때문에 continuous HMM을 이용하여 실험 하였으며, continuous HMM에 맞게 구성하였다.

이상을 정리하면 다음과 같다.

(1) 훈련 단계(training)
1) HMM 모델을 얻기 위하여 continuous HMM 의 방법으로 data를 training한다.
2) output probability를 연산하기 위한 codebook을 제작한다.
3) 1) 과 2)에서 얻은 HMM 모델과

codebook을 이용하여 output probability를 계산하여 메모리에 table로 저장한다.
이때, output probability의 입력으로 codebook의 각 codeword의 중심 벡터를 사용한다.
계산된 결과는 각 codeword별로 table을 만들어 메모리에 저장한다.

(2) 인식 단계(recognition)

1) 입력 data를 훈련단계 2)에서 사용한 codebook을 이용하여 VQ를 한다.
2) 1)에서 VQ된 입력 data와 동일한 값을 사용하여 계산한 output probability를 찾아 메모리에서 읽어 들인다.
3) 읽어온 output probability를 이용하여 viterbi decoding을 수행한다.

이와 같은 방법으로 output probability를 직접 계산하지 않고 메모리에서 access 함으로써 output probability 계산에 필요한 연산을 줄일 수 있다. 본 논문에서는 실험을 위하여 stream당 차수를 줄여서 stream의 수를 4개까지 증가시켰는데, 이 때 각 state에서 필요한 메모리의 크기는 512 codebook을 사용하였을 경우, 512*4이다. 이것은 discrete HMM에서 필요로 하는 메모리의 양과 같다.

지금까지 언급한 것을 그림으로 나타내면 다음과 같다.

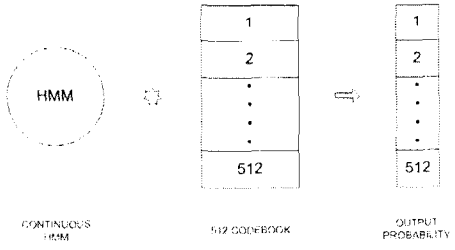


그림 1 output probability table 생성 과정

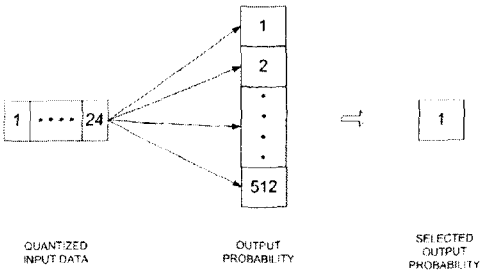


그림 2 output probability table을 불러오는 과정

3.2 연산량 비교

continuous HMM의 output probability를 구하는 식(2)에서 O_{st}를 codebook의 모든 codeword의 중심 벡터로 대신하고 training으로 얻은 HMM의 mean 과 variance, C_{jsm}를 이용하여 미리 계산 가능한 연산을 수행하여 A_s로 대신하면 다음과 같은 식이 된다.

$$b_j(O_t) = \prod_{s=1}^{S} A_s \tag{5}$$

numerical problem을 위해 log를 취하면 다음과 같다.

$$\log b_j(o_t) = \sum_{s=1}^{S} \log A_s \tag{6}$$

음성을 인식하는 절차에서 필요한 output probability를 계산하기 위한 연산은 메모리에 저장되어있는 $\log A_s$을 각 stream에서 선택되어지는 codeword에 따라서 읽어 들여 stream 개수만큼의 log 덧셈만 수행하면 된다.

다음은 각 HMM에서 필요한 출력확률을 구하기 위한 대략적인 연산량을 비교한 것이다.

표 1 각 system의 연산량 비교

HMM	연산량
Continuous	multiplication : (2N+1) • M • S • ST addition : {(2N • M)+M} • S • ST 추가의 log addition
Discrete	VQ - multiplication : N • S • L - addition : (N-1) • S • L addition : (S-1) • ST
제안한 방법	VQ - multiplication : N • S • L - addition : (N-1) • S • L addition : (S-1) • ST

(ST: strate 수, S: stream 수, N: stream당 vector size, L: codeword 수, M: mixture 수)

4. Test 결과 및 분석

Test에 사용된 program은 Cambridge University Engineering Department에서 제공하는 HTK(version 3.2)를 사용하였다. training을 위하여 사용한 DB는 ETRI에서 제작한 445PBW(40화자가 한국어의 음소가 골고루 포함되도록 선정된 445단어를 2회 발성), 국어공학센터에서 제작한 445PBW(452어절, 남38, 여32명 2회 발성), ETRI에서 제작한 POW3848(트라이폰의 분포가 실제와 유사하도록 선정된 3848단어를 48명의 화자가 나누어 발성)을 사용하였다. Test의 음성 data는 국어공학센터에서 제작한 고빈도 2000DB(고빈도 어절 2000개를 남녀 아나운서 각 1회 발성)를 사용하였다. 인식 실험은 diphone subword unit으로 시행하였으며 총 1375개의 음소 모델을 이용하였다. 다음 표2에 실험 파라메터를 정리하였다.

표 2 특징 벡터 추출 제원

Feature Vector	delta Mel-Cepstrum
Frame Period	25 msec
Pre-emphasis Coefficient	0.97
Number of Filterbank Channel	24
Number of Cepstral Parameters	12
Cepstral Liftering Coefficient	22
Window Kind	Hamming Window

VQ된 입력을 가지는 continuous HMM의 인식 성능을 검증하기 위하여 continuous HMM으로 설계한 음성 인식기와 discrete HMM으로 설계한 음성 인식기에 대한 인식률을 test하여 VQ된 입력을 가지는 continuous HMM의 인식률과 비교하였다. continuous HMM의 경우 사용한 mixture에 대하여 뚜렷한 인식률의 증가를 보이는 1 mixture, 2 mixture, 4 mixture에 대하여 test하였고, discrete HMM의 경우 가장 높은 인식률을 얻을 수 있으면서 실제 구현 가능한 512 codebook에 대한 인식률을 test하였다. VQ된 입력을 가지는 continuous HMM의 인식률 test는 continuous HMM의 2 stream 의 diagonal matrix로 training한 HMM을 사용하였을 경우에 대하여 실시하였다. 그리고 VQ로 인한 인식률 손실을 보완하기 위하여 stream의 수를 증가시켜 훈련한 HMM을 사용하였을 경우, full covariance matrix를 사용하였을 경우, noise가 추가된 data를 가지고 훈련하여 얻은 model의 HMM을 사용하였을 경우의 각각에 대하여 인식률을 test하여 인식률 변화를 관찰하였다. continuous HMM의 인식률 실험에서 full covariance matrix를 사용하였을 경우에 대하여 test를 하지 않은 이유는 full covariance matrix의 사용은 인식률을 높이는데 효과가 있지만 연산량이 너무 많은 이유로 실제로 사용하기에 부적합하기 때문이다. 그러나 VQ된 입력을 가지는 continuous HMM의 경우 미리 output probability를 계산하여 메모리에 저장한 후 이를 활용하기 때문에 full covariance matrix에 따른 연산량 증가가 없다. 따라서 continuous HMM은 diagonal matrix를 사용하여 구현한 음성 인식기만을 test 하였고, VQ된 입력을 가지는 continuous HMM의 음성 인식기는 diagonal covariance matrix와 full covariance matrix를 이용하여 구현한 음성 인식기 모두에 대하여 test하였다.

표 3 continuous system의 test 결과

Stream	1 Mixture	2 Mixture	4 Mixture
2 Stream	89.82	92.30	93.13
4 Stream	89.82	91.07	92.05

continuous HMM으로 구현한 음성 인식기는 mixture 개수가 증가함에 따라 인식률이 증가하여 4 mixture 일 때 가장 높은 인식률을 얻을 수 있었다. 2 stream(12차 + 12차delta)을 사용한 경우가 4 stream(12차 + 12차 + 12차delta + 12차delta)일 때보다 더 높은 인식률을 얻을 수 있었는데 그 이유는 stream을 나눌 경우 나누어지는 data가 서로 independent하다는 가정 하에서 가능한데 실험에서 4개의 stream으로 나누었을 때 독립성이 유지되지 않았기 때문에 인식률 저하의 원인으로 작용하였

다. continuous HMM의 경우 4 mixture, 2 stream 일 때 93.13%의 가장 높은 인식률을 얻었다.

다음은 discrete HMM으로 설계한 인식기의 test 결과이다.

표 4 discrete system의 test 결과

코드북 크기	Stream	Result
512	1	87.22
512	2	89.10
512	4	88.65

codebook의 크기가 커질수록 인식률이 증가하였다. 그러나 2개 이상의 stream으로 나누었을 경우 표4의 결과와 같이 인식률이 감소되었다. continuous HMM의 경우와 마찬가지로 4개의 stream으로 나눌 경우 독립성이 유지되지 않아 오히려 인식률 저하의 원인으로 작용하였다. 이 같은 특성으로 같은 512 codebook의 경우 2개의 stream을 사용하였을 때 최고의 인식률을 얻을 수 있었다. 다음은 논문에서 제안한 방법인 VQ된 입력을 가지는 continuous HMM의 음성 인식기를 설계하여 test한 결과이다.

표 5 본 논문에서 제안한 방법의 test 결과

	코드북 크기	Stream	1 Mixture	2 Mixture	4 Mixture
Diagonal Matrix	256	4	86.10	87.53	88.43
	512	2	80.43	81.97	83.68
	512	4	86.55	88.03	88.95
Full matrix	256	4	88.50	88.90	89.80
	512	2	83.60	84.55	84.90
	512	4	89.75	90.28	90.72

앞서 설명한 바와 같이 VQ된 입력을 가지는 continuous HMM의 음성 인식기의 test는 continuous HMM의 2 stream, diagonal covariance matrix로 training한 HMM을 이용하여 기본적인 인식 결과를 확인 하였고, VQ 오차로 인한 인식률 감소를 보완하기 위해 선택한, stream 수를 증가시키고 full covariance matrix로 training한 HMM을 사용하였을 때의 인식률 변화를 test하여 비교하였다. 먼저 2 stream, diagonal covariance matrix로 훈련한 HMM을 사용하고 512 codebook으로 VQ한 data를 입력으로 하였을 경우의 인식률은 그림 3과 같다. 본 논문에서 제안한 VQ된 입력을 가지는 continuous HMM의 인식률은 continuous HMM과 discrete HMM에 비하여 낮은 결과를 얻었는데 VQ로 인한 오차가 전체 system에 크게 영향을 주었기 때문이다.

그림 4는 512 codebook으로 VQ한 입력 data에 대해 VQ 오차의 영향을 줄이기 위해 stream 수를 2배 증가하여 훈련한 HMM을 사용하여 test한 결과이다. stream의 수 증가에 따른 확실한 인식률 향상을 얻을 수 있었다. 이 결과는 continuous HMM과 discrete HMM의 결과와 다른데, 그 이유는 VQ 오차가 stream을 나누었을 경우 크게 줄어들었고 그로인한 영향이 독립성이 유지되지 않아 발생하는 오차보다 HMM에 더 큰 영향을 주었기 때문이다.

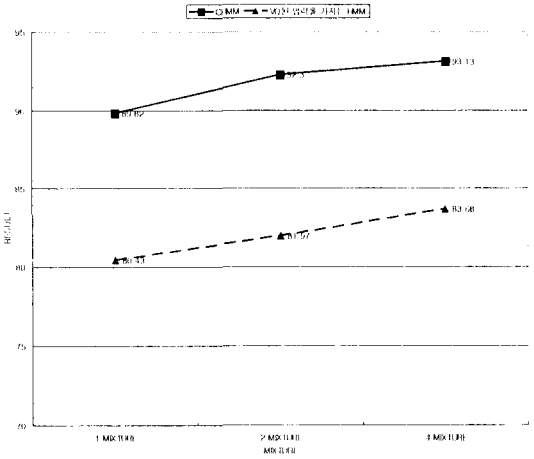
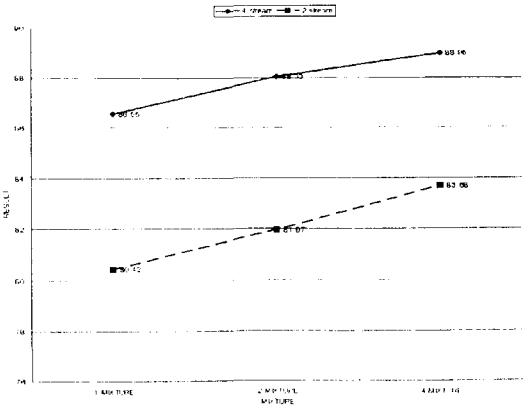


그림 3 stream의 VQ된 입력을 가지는 HMM과 CHMM의 인식률 비교

그림 4 stream의 HMM을 사용한 인식률

다음으로 matrix 변화에 따른 인식률의 차이를 그림 5에 나타내었다. 512 codebook으로 VQ한 입



력 data에 대하여 4 stream, diagonal covariance

matrix로 훈련한 HMM을 사용하여 test한 결과와 4 stream, full covariacne matrix를 사용하여 test한 결과를 비교한 것이다. diagonal covariance matrix를 사용하였을 때보다 full covariance matrix를 사용하였을 경우 더 높은 인식률을 얻을 수 있었는데 이것역시 full covariance matrix를 사용함으로써 VQ 오차로 인한 인식률 저하를 보완하였기 때문이다.

다음은 각 HMM에서 얻은 인식률을 비교한 것이다.

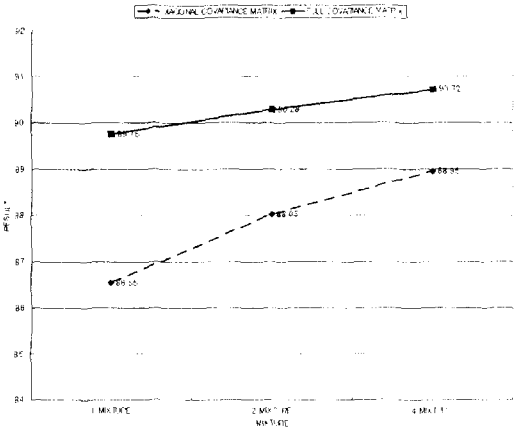


그림 5 matrix에 따른 인식률 변화

표 6 각 HMM의 인식률 비교

HMM	코드북 크기	Stream	Mixture	Result
continuous	•	2	4	93.13
discrete	512	2	•	89.10
제안한 방법	512	4	4	90.72

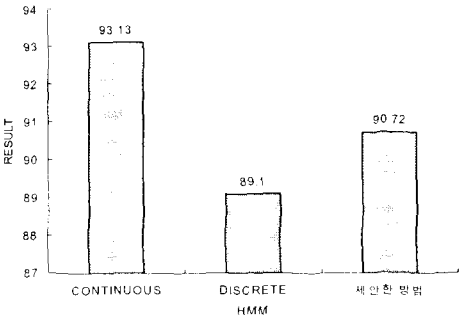


그림 6 각 HMM의 인식률 비교

표 6에서 4 mixture, 2 stream의 continuous HMM과 512 2 stream을 가지는 discrete HMM을 VQ 오차로 인한 인식률 저하를 보완하기 위해 stream을 증가시키고 full covariance matrix를 사용한 본 논문에서 제안하는 방법과 비교하였다. 연산량은 VQ된 입력을 가지는 continuous HMM이 discrete HMM에 비하여 2개 더 많은 stream을 사용하였으므로 약간의 덧셈을 추가 되지만 1.62% 더 높은 인식률을 얻을 수 있었다. continuous HMM과 비교해서는 인식률에 대하여 2.41% 더 낮은 인식률을 얻었지만 연산량은 아주 크게 줄어든 것을 알 수 있다.

본 논문에서 제안하는 방법의 인식률이 continuous HMM에 비하여 낮은 이유는 VQ에 의하여 오차가 발생하였기 때문이다. VQ 오차를 noise라고 가정하면 본 논문에서 제안하는 방법은 noise에 노출된 환경에서 test한 경우라고 생각할 수 있다. 따라서 training data역시 같은 noise에 노출된 환경을 만들어 주고 이에 대한 test를 실시하였다. training data에 noise를 추가하기 위해 training data를 VQ한 후 이에 대한 표준편차를 각 차수별로 구하고 표준편차범위에 해당하는 random noise를 생성하여 기존의 training data에 더하였다. 아래에 그 내용을 정리하였다.

1. training data에 VQ를 실행함
2. 특징벡터의 각 차수마다 VQ 오차의 표준편차를 구함.
3. 표준편차만큼의 random noise를 생성하여 원래의 특징벡터에 더함.

random noise 생성 범위는 다음과 같이 정하였다.

$$\sigma^2 = \frac{\triangle^2}{12} \qquad (7)$$

σ^2은 분산을 나타내고, $\triangle$은 원하는 random

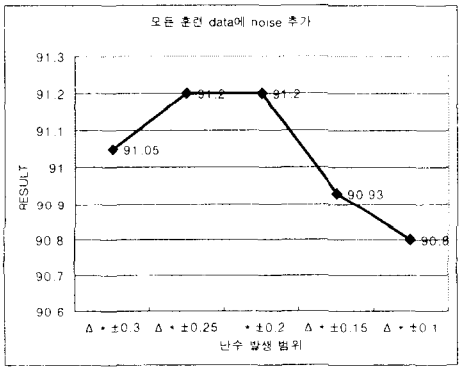
noise의 생성 범위를 나타낸다.

4. random noise 생성 범위를 변경하면서 3의 과정을 반복 실험.

다음은 train 특징 벡터에 noise를 추가하여 test한 결과이다.

표 7 특징벡터에 noise를 추가하여 test한 결과

Noise가 추가된 data	난수 발생 범위	4 Mixture
모든 훈련 data	△ * ± 0.3	91.05
	△ * ± 0.25	91.20
	△ * ± 0.2	91.20
	△ * ± 0.15	90.93
	△ * ± 0.1	90.80
음소단위로 녹음된 훈련 data (boot strap data)	△ * ± 1.49	91.20
	△ * ± 1.5	91.78
	△ * ± 1.52	91.40
	△ * ± 1.55	91.30



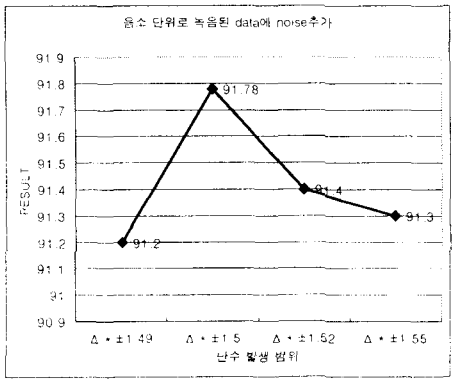


그림 7 훈련 data에 noise추가하여 test한 결과

training data로 음소단위로 녹음한 음성 data(boot strap data)와 단어 단위로 녹음한 음성 데이터를 사용하였는데, 전체 training data와 음소 단위로 녹음된 training data(boot strap data)에 각각 noise를 추가하여 실험을 하였다. random noise의 생성 범위를 변경해가면서 가장 높은 인식률을 얻은 범위를 표 7에 제시하였다. 그림 7에서 알 수 있듯이 전체 training data에 noise를 추가하여 test한 인식률 보다 음소 단위로 녹음된 training data에 noise를 추가하여 test한 인식률이 더 좋았다. 결론적으로 continuous HMM의 HMM을 그대로 사용한 것에 비하여 training data에 noise를 추가하여 training한 후 얻은 HMM을 사용할 경우 더 좋은 인식률을 얻을 수 있는 것을 알 수 있다.

다음 표 8과 그림8에 VQ된 입력을 가지는 continuous HMM에 대해, VQ 오차로 인하여 인식률이 감소하는 것을 보완하기 위해 stream 수를 증가, full covariance matrix 사용, noise를 추가하여 훈련한 HMM을 사용하였을 때의 누적된 인식률 변화를 나타내었다.

표 8 VQ로 인한 인식률 감소를 줄이기 위해 선택한 방법에 따른 인식률 변화

조건	stream	2	4	4	4
	matrix	diagonal	diagonal	full	full
	HMM	기존의 HMM	기존의 HMM	기존의 HMM	*noise* 추가된 HMM
인식률		83.68	88.95	90.72	91.78

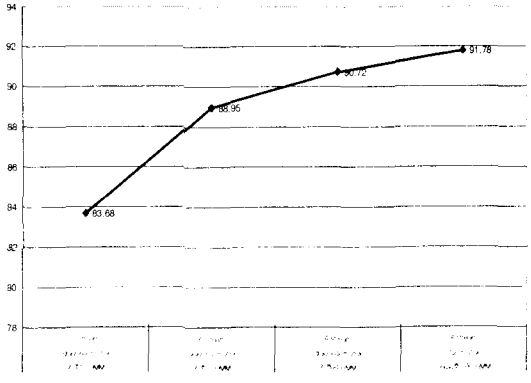


그림 8 VQ 오차로 인한 인식률 감소를 줄이기 위해 선택한 방법에 따른 인식률 변화

표 9과 그림 9에서 noise를 추가하여 얻은 인식률과 다른 3개의 HMM의 인식률을 비교하였다.

표 9 noise를 추가하여 얻은 인식률과 다른 3개의 HMM의 인식률 비교

HMM	코드북 크기	Stream	Mixture	Result
continuous	•	2	4	93.13
discrete	512	2	•	89.10
제안한 방법 (noise를 추가하였을 경우)	512	4	4	91.79

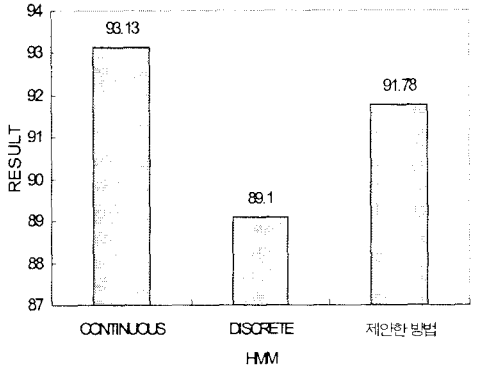


그림 9 noise를 추가하여 얻은 인식률과 다른 3개 HMM의 인식률 비교

5. 결론

본 논문에서는 VQ된 입력을 가지는 continuous HMM을 이용하여 continuous HMM에서 필요로 하는 방대한 연산량을 줄이고 discrete HMM보다 높은 인식률을 얻을 수 있는 방법에 대하여 제시하였다. 이 방법은 discrete HMM과 동일한 연산량으로 인식 알고리즘을 수행할 수 있으며 discrete HMM과 동일한 메모리를 가지는 음성 인식기의 구현이 가능하다. 또한 VQ된 입력을 가지는 continuous HMM으로 음성 인식기를 구현할 경우 continuous HMM으로 훈련하여 얻은 HMM을 그대로 사용할 수 있어서 여건에 따라 기존의 continuous HMM과 VQ된 입력을 가지는 continuous HMM을 선택적으로 사용할 수 있다. 한편 VQ된 입력을 가지는 continuous HMM은 기본적으로 discrete HMM보다 낮은 인식률을 얻었으나 VQ로 인한 인식률 감소를 보완하기 위해 다음의 3가지의 방법을 적용하여 discrete HMM보다 높은 인식률을 얻을 수 있었다. 첫 번째로, stream의 수를 증가하는 방법을 활용하였다. stream의 수가 증가함에 따라 VQ 오차가 감소되어 인식률의 향상을 얻을 수 있었다. 두 번째로, full covariance matrix를 사용하여 인식률 저하를 보완하였다. 본 논문에서 제안하는 방법은 full covariance matrix를 사용하더라도 인식단계 이전에 미리 full covariance matrix를 사용한 연산이 이루어지기 때문에 full covariance matrix 사용에 따른 연산량의 증가 없었다. 세 번째로, training data에 noise를 추가하여 훈련한 HMM을 사용함으로써 VQ 오차로 인한 인식률 감소를 보완하였다. 이와 같은 방법으로 VQ된 입력을 가지는 HMM의 인식률을 증가시켜 discrete HMM보다 높으며 continuous HMM에 근접한 인식률을 얻을 수 있었다. 본 논문에서 제안한 방법의 효과는 더 많은 mixture를 사용하여 인식기를 설계 할 경우 극대화 된다. mixture의 수가 증가할수록 그에 비례하여 연산량이 늘어나기 때문이다. 5장의 test 결과를 통하여 VQ된 입력을 가지는 continuous HMM으로 output probability를 위한 연산량을 줄이면서 continuous HMM에 근접한 인식률을 얻을 수 있음을 확인 할 수 있었다.

참고문헌

[1] *The HTK Book* (for HTK Version 3.2)
[2] Lawrence Rabiner & Biing-Hwang Juang, *Fundamentals of Speech Recognition*, Prentice Hall.
[3] John R. Delldr, Jr & Hohn G. Proakis & John H. L. Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan Publishing Company
[4] L. R. Rabiner and B. H. Juang, "*An introduction to hidden markov models*," IEEE ASSP Magazine, pp. 4-16, 1986.
[5] A. Viterbi, "*Error bounds for convolution codes and an asymptotically optimum decoding algorithm*," IEEE trans on Info. Theory, Vol. 13, pp. 260-269, Apr., 1967.

[6] B. H. Juang, "*Maximum-likelihood estimation for mixture multivariate stocastic observations of markov chain*," AT&T Technical Journal, Vol. 64, pp. 1235-1249, 1985.
[7] J. Picone, "*Continuous speech recognition using hidden markov models*," IEEE ASSP Magazine, Vol. 7, pp. 26-41, July, 1990.
[8] K. F. Lee and H. W. Hon, "*Speaker independent phone recognition using hidden markov models*," IEEE Trans. on ASSP, Vol. 37, pp. 1641 1648, 1989.