신경망을 이용한 멜로디 패턴 인식

Melody Pattern Recognition using
Artificial Neural Networks

서울대학교 공과대학
컴퓨터 공학부
97419-012
김석환
apex@apex.pe.kr

요약

지금까지 컴퓨터를 이용한 문서 검색에는 많은 연구가 있어왔다. 우리가 이용하는 인터넷 검색 엔진은 이러한 문서 정보 검색을 기반으로 하는 하나의 시스템이라고 할 수 있다. 그러나 소리나 영상에 관하여 사용자 친화적(user-friendly)인 정보 저장 및 검색에 관해서는 아직 실용적인 연구가 미흡한 실정이다.

본 연구는 음악 곡에서의 멜로디 패턴을 이용한 저장 및 검색 방법을 통해 정보 처리의 한 방법을 모색한다. 이를 구현하기 위해 멜로디 정보를 표현하도록 사인 파를 생성하였으며, 이를 ADC(Analog to Digital Convert) 및 FFT(Fast Fourier Transform) 를 통해 컴퓨터 데이터로 변환한 후 평균율에 따라 음 높이를 계산하여 멜로디 패턴 데이터를 만들었으며, 패턴 학습 및 분류, 검색에 신경망 알고리즘을 사용하였다.

실험 결과 제한된 규모의 데이터 집단에 대해 신경망을 이용한 패턴의 기억 학습 및 검색은 어느 정도 좋은 결과를 보여주었으나 주관적 분류에서는 다소 좋은 결과를 보이지 못하였다. 추후 더욱 효율을 높이기 위한 신경망 구조의 연구 및 음성 처리와 다른 검색 방식과의 협력이 필요할 것이다.

Ⅰ. 서론

1.1 연구 목적

인류가 지금까지 쌓아온 정보의 양은 방대하며 그 분야와 종류 또한 매우 다양하다. 때문에 이를 개인이 모두 익히는 것은 불가능하며, 학습한 정보라도 시간이 지나면서 일부분 잊어버리게 된다. 따라서 핵심 정보만을 기억하였다가 필요에 따라 목적과 용도에 맞는 정보를 찾는 방법을 모색하지 않을 수 없다. 그러기 위해서는 이들 자료들을 분류 및 정리, 그리고 처리할 필요가 있다.

지금까지 정보의 저장 및 표현 수단으로 사용되던 필기구, 종이, 물감, 악기 등의 도구가 컴퓨터라는 도구 안에서 구현이 가능할 정도로 전자 기술이 발전되었다. 따라서 기존의 정보가 컴퓨터의 처리 범주로 들어오게 됨에 따라 컴퓨터가 다루어야 할 정보의 양이 커지게 되었고, 이들을 정리하고 분류해야 할 필요성이 생기게 되었다.

현재 정보의 주 저장 및 표현 수단으로서 문서에 대한 검색은 상당히 많은 연구가 이루어져 있다. 이는 문서의 특성상 문자라는 이산적(discrete) 언어에 의해 순차적(sequential)으로 표현되므로 컴퓨터가 쉽게 처리할 수 있기 때문이다. 그러나 다른 분야, 예를 들면 영상과 음향에 대한 검색은 비교적 다루기가 어려운데, 이들 정보는 병렬적(parallel)이며 연속적(continuous)인 특징을 가지고 있기 때문이다. 따라서 이를 인식하고 검색하는 데에는 컴퓨터가 처리할 수 있는 새로운 방법이 필요하다.

이 중 음향은 청각과 관련되어 인간이 친숙하게 의사 소통할 수 있는 분야라고 할 수 있다. 우리의 주위에서 듣게 되는 모든 소리가 음향이며 이러한 음향 정보 중에서 원하는 것을 저장하고 검색할 수 있는 방법이 필요할 것이다. 이 논문에서는 컴퓨터 내에서 음향의 한 분야인 음악을 어떤 방식으로 저장하고, 검색할 것인지에 대하여 살펴보기로 한다.

1.2 연구 내용

음악이란 소리를 이용하여 인간의 감정이나 생각을 표현하는 수단이다. 음악에서 소리는 음이라는 단위로 표현할 수 있으며, 이 음들이 시간적 순서에 따라 배열되어 하나의 곡이 완성된다고 생각할 수 있다. 이들 음은 악보의 형식에서 음표로 기호화 될 수 있으며, 지금까지 작곡된 수많은 음악들이 이렇게 이루어진 것이다. 따라서 어떤 음악을 컴퓨터가 처리할 수 있는 데이터로 변환 하려면 음악 소리로부터 그 음을 추출하는 것이 하나의 방법이 될 수 있다. 다시 말해, 특정 소리 음으로부터 그 음을 나타내는 기호로 변

환시켜 음표로 표현할 필요성이 있다.

음표를 이용해 높이, 길이, 강약 등을 표시할 수 있는데, 여기서는 음표들을 순서대로 배열한 표현 형태인 멜로디(melody: 선율)를 다룬다. 이를 위해 음악 소리로부터 음표 값을 추출하려면 다음과 같은 기술이 필요하다. 우선 연속적인 소리 신호를 이산적인 데이터 값으로 변환 시키는 ADC(Analog to Digital Convert) 과정이 필요하다. 이렇게 변환된 신호는 시간을 축으로 진행되는 파동의 형태를 띄게 되는데, 이를 다시 주파수를 축으로 표현되는 스펙트럼(spectrum)의 형태로 변환시켜 입력 신호의 주파수를 확인해야 한다. 여기서는 FFT(Fast Fourier Transform) 방식을 사용한다. 마지막으로 이 주파수로부터 우리가 인식하는 음계의 영역으로 표현하기 위해 평균율 지식(Equal Temperament)을 이용하게 된다.

이 과정을 거쳐 변환된 멜로디 데이터는 컴퓨터가 처리할 수 있는 이산적이고 순차적인 특징을 띄게 된다. 따라서 음악 소리를 기호화 하여 저장할 수 있고 이를 검색할 수 있는 선행 조건을 만족한다.

인간은 음악의 모든 소리를 듣지 않고도 멜로디 만으로 곡을 구분할 수 있는 능력을 지니고 있기 때문에 이러한 멜로디의 일정 부분인 멜로디 패턴을 이용하여 곡을 검색하는 방법을 생각할 수 있다. 멜로디 패턴을 검색하는 방법에는 크게 두 가지가 있을 수 있는데, 그 첫 번째는 문서검색 방법과 마찬가지로 찾고자 하는 멜로디 패턴과 저장되어 있는 멜로디 패턴이 일치하는지를 평가하는 것이다. 또 다른 방법으로 패턴의 값들을 입력값으로 하는 신경망을 구성하여 미리 학습 시킨 후 찾고자 하는 입력 패턴을 넣어주면 학습된 패턴의 번호를 출력하도록 하는 것이다. 전자의 경우 사전 학습 과정이 필요치 않으며 방대한 데이터를 저장하고 검색할 수 있는 장점을 가지는 반면, 일부 에러 값을 가지는 패턴에 대하여 원하는 결과값을 출력하는데 어려움이 있다. 후자의 방법은 사전 학습 시간이 필요하며, 비교적 입력 데이터의 수에 제한을 받는 문제가 있지만, 병렬적 연산으로 인해 검색 속도가 빠르고 에러 값이 포함된 경우에 대해서도 원하는 결과값을 낼 수 있는 능력이 있다는 장점을 가진다.

이번 논문에서는 신경망을 이용한 멜로디 패턴의 인식을 다루는 것으로, 멜로디 패턴의 학습과 표현에 관한 연구를 진행한다.

1.3 연구 범위

크게 두 가지 주제에 대해 진행한다. 하나는 여러 가지 멜로디 패턴을 신경망 학습을

통해 저장한 후 검색 값을 넣었을 때 원하는 패턴 번호를 출력시키는 방법으로, 은닉 층의 뉴런 수와 출력 층의 뉴런 수에 따른 기억 능력과, 에러를 포함하는 검색 값에 대해 원하는 결과값을 출력할 수 있는지를 알아보는 것이다.
다른 하나는 신경망이 실제 인간과 같이 감정에 대한 인식을 알아보는 것으로, 멜로디 패턴을 즐거운 것과 슬픈 것 두 가지로 분류 학습 시킨 후, 테스트 곡들에 대한 인식률을 평가하는 것이다.

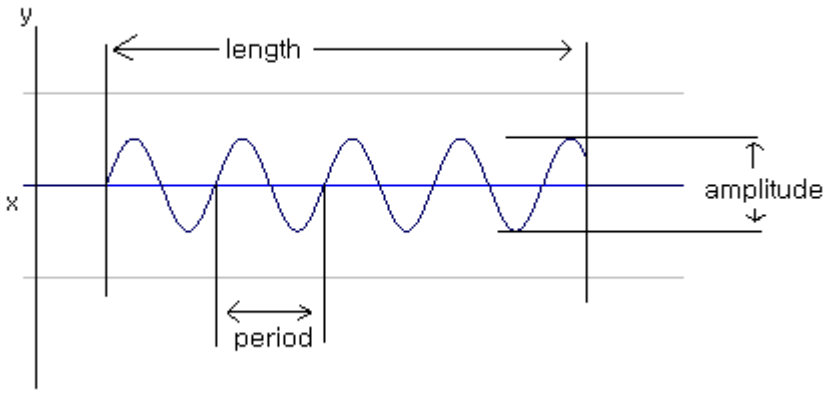
1.4 논문의 구성

본 논문은 멜로디 패턴의 생성과 학습 및 검색에 관련하여 기존의 여러 가지 기술의 적용과 신경망의 사용에 중점을 두어서 완성되었으며, 다음과 같이 구성되었다.
2절에서는 관련 연구로서 멜로디 패턴의 생성에 필요한 여러 가지 지식 및 기술에 대하여 살펴본다. 3절에서는 이번 연구에 사용할 신경망에 대하여 살펴본다. 4절에서는 생성 멜로디 패턴을 가지고 신경망에 실제 학습시켜 원하는 결과를 보이는가에 대한 실험을 행한다. 5절에서는 논문의 결론과 향후 과제에 대한 내용을 다루는 것으로 마무리한다.

Ⅱ. 관련 연구

2.1 멜로디 (melody: 선율)

멜로디란 서로 다른 음높이(pitch)들의 시간적인 연속이며 연속 선으로 인식된다. 선율은 수평 구조로서. 한 음에서 다른 음으로, 그리고 또 다른 음으로의 진행 구조로 이루어진다[1]. 그러므로 선율을 데이터 구조화 시키려면 음높이와 시간을 묶어 하나의 원소화가 될 수 있는 음(note)에 대해 알아볼 필요가 있다.
음은 음파에서부터 추출된 특징 값을 가진 데이터 구조(structure)로 볼 수 있다. 따라서 우선 음파에 대한 이해가 필요하다. 음파(sound wave)라는 것은 물리적으로 고체, 액체, 기체 등을 매질로 하여 진행되는 종파(longitudinal wave)이다[2]. 이러한 음파는 특정 지점에서 시간에 관한 압력의 변화로 표현할 수 있는데, <그림1> 과 같이 시각적으로 표현된다.



< 그림1. 음파의 모양 >

음파는 그림에서 보듯 진폭(amplitude), 길이(length) 라는 주요 특징을 가진다. 이는 음의 높이(pitch)와 길이(duration)에 대응될 수 있도록 처리해 주어야 한다.

음의 높이는 다른 말로 주파수(frequency)라고 부를 수 있으며 이는 식 (2.1.1)을 사용하여 얻을 수 있다.

$$F = \frac{1}{T} \qquad \text{F: frequency} \qquad \text{T: period} \qquad (2.1.1)$$

우리가 약속한 라(A4) 음은 440Hz 의 주파수 값으로 약속되어 있다.

음의 길이에 대해서는 음파의 길이 값을 그대로 채용하는데 무리가 있으며, 실제 악보에서 표기하는 음의 길이 역시 곡 전체의 빠르기에 좌우 되기 때문에 상대적인 값을 가지는 것이 좋을 것으로 판단한다. 멜로디 패턴 내에서 각 음들 사이의 상대적인 음파의 길이(length) 비율로 값을 표현하는 것이 하나의 방법이 될 수 있을 것이다.

2.2 음계 (Musical Scale)

음계라는 것은 음(note)들이 연속적으로 높아지거나 또는 낮아지는 순서로 정렬한 것을 뜻한다. 대부분의 작곡은 음계를 기반으로 이루어 지는데 음의 구성 개수에 따라 5음(pentatonic), 12음(chromatic), 7음(major & minor diatonic, Dorian & Lydian modes 등) 정도가 가장 일반적으로 사용된다. 서양 음악에서는 한 옥타브를 반음계(semitone)라고 부르는 12개의 단계로 나눈다.

음계에서 음들 사이의 높이 값을 어떻게 책정하느냐에 따라 순정율(Just scale), 피타고리

안 음계(Pythagorean scale), 평균율(equal temperament) 세 가지가 있다[3].

순정 율은 각각의 음들 사이에 일정한 값의 주파수 비율을 적용시키고 있다.

< 표1. 두 음 사이의 주파수 비례에 따른 간격 >

2:1	옥타브
3:2	완전 5도
4:3	완전 4도
5:3	장 6도
5:4	장 3도
8:5	단 6도
6:5	단 3도

이를 적용하여 도~시까지의 음들의 주파수 비율은 다음과 같다.

< 표2. 순정 율에서 음들 간의 주파수 비율 >

도	레	미	파	솔	라	시	도
1	9/8	5/4	4/3	3/2	5/3	15/8	2

피타고리안 음계는 도 음을 기준으로 하여 위쪽으로 완전 5도씩 또는 아래쪽으로 완전 4도씩을 증감하면서 두 음 사이의 비율을 적용시켜 전체적인 음의 비율을 얻고 있다.

< 표3. 피타고리안 음계에서 음들 간의 주파수 비율 >

도	레	미	파	솔	라	시	도
1	9/8	81/64	4/3	3/2	27/16	243/128	2

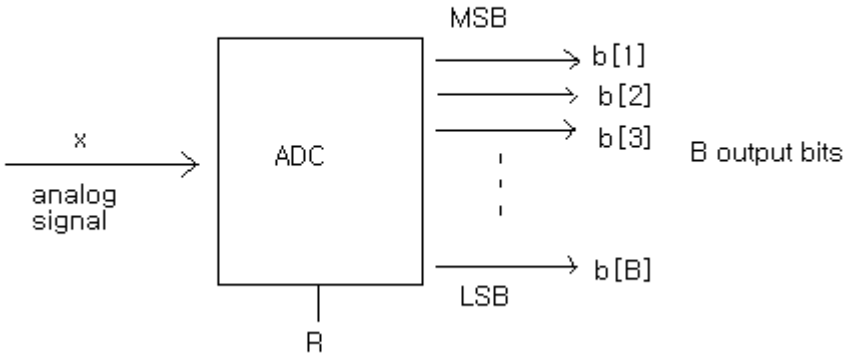
평균율은 옥타브 내의 12 반음계에 대하여 인접한 두 음 사이가 모두 동일한 비율을 가지는 음계이다. 한 옥타브 사이가 2배만큼의 주파수 차이가 나며, 그 사이에는 동일한 비율이 적용되도록 하기 위해 반음계 사이에 $\sqrt[12]{2}$ 배 만큼 차이가 나도록 하고 있다. 따라서 반음계 사이는 1:1.05946 의 비율 값을 가진다.

평균율은 두 음 사이의 소리가 가장 듣기 좋은 비율로 이루어 진 것은 아니지만, 음들

간의 불균형을 줄이며 실제 여러 악기들을 사용하는 협주곡에서 서로간의 음계를 조율할 수 있는 장점이 있어 가장 널리 사용된다고 할 수 있다.

2.3 ADC (Analog to Digital Convert)

A/D converter 는 연속 신호(continuous signal)를 이산 신호(discrete signal)로 바꾸는 장치이다. 이는 아날로그 입력 값 x를 B bit 값으로 양자화(quantize) 시키는 것이다[4].



< 그림2. A/D converter 구조도 >

이를 더욱 세분화 하면 C/D(Continuous-to-Discrete) 부분, Quantize 부분, Coder 부분으로 나뉜다. C/D 부분은 sample and hold 부분으로도 불리는데 주기적으로 입력 값을 검사하여 그때 포착한 값을 유지하도록 하는 부분이다. 이렇게 얻은 값을 B bit 의 이진 값으로 표현할 수 있는 양자화 과정을 거치게 된다. 마지막으로 이 값을 offset binary 또는 2's complement 방식으로 수치화 한다[5]. 식 (2.3.1)은 샘플링을 통한 C/D 변환 과정을 나타낸 것이다.

$$x_0(t) = \sum_{n=-\infty}^{\infty} x_a(nT)\delta(t-nT) \tag{2.3.1}$$

2.4 DFT(Discrete Fourier Transform)/FFT (Fast Fourier Transform)

DFT 는 입력 신호의 주파수 분포(spectrum) 분석을 행할 수 있는 기법으로, 특정 주파수 구간 내에서 균일한 간격으로 설정된 주파수에서의 값을 구할 수 있다. 이는 복소수의 형태로 표현되며 이를 절대값으로 변환하면 해당 주파수에서의 크기를 알 수 있다.

N-point DFT 라는 것은 N 개의 주파수에 대해 각각 DTFT(Discrete Time Fourier Transform) 처리를 해주는 것으로 이해할 수 있으며, 식(2.4.1)과 같다[6].

$$X(\omega_k) = \sum_{n=0}^{L-1} x(n) e^{-j\omega kn} \ , \omega_\kappa = \frac{2\pi\kappa}{N}, k = 0,1,..., N-1 \qquad (2.4.1)$$

FFT 는 DFT를 응용한 것으로 샘플들이 규모(dimension)가 커짐에 따라 증가하는 처리 속도를 개선한 방법이다. 전형적인 FFT 알고리즘은 다음 세 가지의 과정을 따른다[7].

1) N-차원 입력을 N 개의 1-차원 신호로 나눈다.
2) N 개의 신호에 대하여 1-point DFT를 수행한다.
3) N 개의 1-point DFT 값을 하나의 N-point DFT 값으로 합친(merging)다.

N 개의 샘플 데이터를 N/2 개로 나누는 과정은 다음의 공식을 따른다.

$$X(k) = \sum_{n=0}^{N-1} W_N^{kn} x(n) \qquad (2.4.2)$$

$$X(k) = G(k) + W_N^k H(k) \qquad (2.4.3)$$

$$G(k) = \sum_{n=0}^{N/2-1} W_{N/2}^{kn} g(n), g(n) = x(2n) \qquad (2.4.4)$$

$$H(k) = \sum_{n=0}^{N/2-1} W_{N/2}^{kn} h(n), h(n) = x(2n+1) \qquad (2.4.5)$$

$$W_N^k = e^{-2k\pi j/N}, \quad k = 1,2,...N-1, \quad n = 1,2,..., N-1$$

기본적으로 식(2.4.2)와 같이 표시되는 N-point DFT 방식을 식(2.4.3)의 형태로 이분하는 것으로, 이렇게 나뉘어진 각각에 대하여 다시 이분하는 식으로 여러 번 반복하게 되면 최종적으로 N 개의 1-point DFT 식으로 표현할 수 있게 된다. 이를 DFT 처리한 후 이들을 다시 인접한 두 단위씩으로 묶어서(merging) 최종적으로 N-point DFT 와 같은 결과를 얻게 된다.

Ⅲ. 핵심 연구 내용

3.1 문제점 및 해결 전략

이번 연구에서는 국내외 동요 및 대중가요에서 100개를 선택하여 멜로디 패턴을 만들었

다. 멜로디 패턴의 길이는 두 소절 정도인 12개의 음표 값으로 구성하였는데, 패턴의 길이가 너무 길면 검색의 실용성이 떨어지고 길이가 너무 짧으면 신경망 학습이 어렵기 때문이다. 멜로디의 경우 각 음들이 높이와 길이를 가지는데, 이번 연구에서는 음의 높이만을 데이터화 하여 표현하였다. 음의 길이에 대한 수치화 방법은 앞에서 설명한 바와 같이 음들 간의 길이 비율 값으로 책정할 수도 있으나 다른 패턴들 간의 빠르기(tempo) 등을 고려해야 하고 대부분의 곡들이 음들 사이에 공백이 존재하는 문제가 있다고 판단하여 이번 연구에서는 포함시키지 않았다.

실제 학습 시 멜로디 패턴을 첫 번째 음에 대한 차이를 계산하여 첫 번째 값을 뺀 나머지 11개의 차이 값을 데이터로 사용하였다. 이는 실제 절대음감을 가진 사람이 매우 드물고 따라서 대부분의 사람은 처음 음에 대해 상대적인 높이 값을 가진 패턴을 기억하고 있기 때문에, 절대 값이 아닌 상대 값을 저장하는 것이 검색에 있어 신뢰성 있는 결과를 얻을 수 있는 방법이 되기 때문이다.

검색 방법으로는 문서 검색과 같은 값 비교 방식도 고려해 볼 수 있으나, 오류 값을 가지는 패턴에 대하여 처리 방법을 생각해 보아야 하는 문제가 생긴다. 이번 연구에서는 에러에 어느 정도 신뢰성을 가지고 있다고 평가 받는 신경망을 사용하여 학습과 검색 및 인식에 관해 알아보기로 한다.

3.2 멜로디 패턴 데이터

본 논문 연구를 위해 직접 만든 데이터로 그 생성 과정은 다음과 같다.

1) 생성시킬 멜로디 패턴을 선택한다. 곡 중 연속된 12개의 음을 선정한다.
2) 선정된 멜로디 패턴에 해당하는 음에 맞는 sine wave 를 생성한다.
3) Sine wave 를 8kHz로 A/D Converting 한다.
4) 1024-point FFT 변환을 통해 주파수 스펙트럼을 구한다.
5) 스펙트럼을 보고 가장 값이 큰 주파수의 index를 구한다.
6) 그 index가 가리키는 주파수에 해당하는 음 높이를 평균율을 적용하여 찾는다.
7) 1)~6)의 과정을 12개의 음에 대하여 반복하여 하나의 패턴 데이터를 완성한다. <표 4> 참고
8) 첫 번째 음을 기준으로 해서 다음 음들이 가지는 차이 값을 계산한 11개의 데이터를 구성한다. <표5> 참고

< 표4. 멜로디 패턴 1 >

1] 48 55 56 53 55 56 55 51 51 48 48 44
2] 51 54 54 54 56 53 49 53 53 51 51 54
3] 51 55 51 53 53 55 53 51 51 51 51 51
4] 49 51 53 54 46 46 53 53 51 49 46 54
5] 48 46 44 44 44 46 48 49 51 53 55 56
…

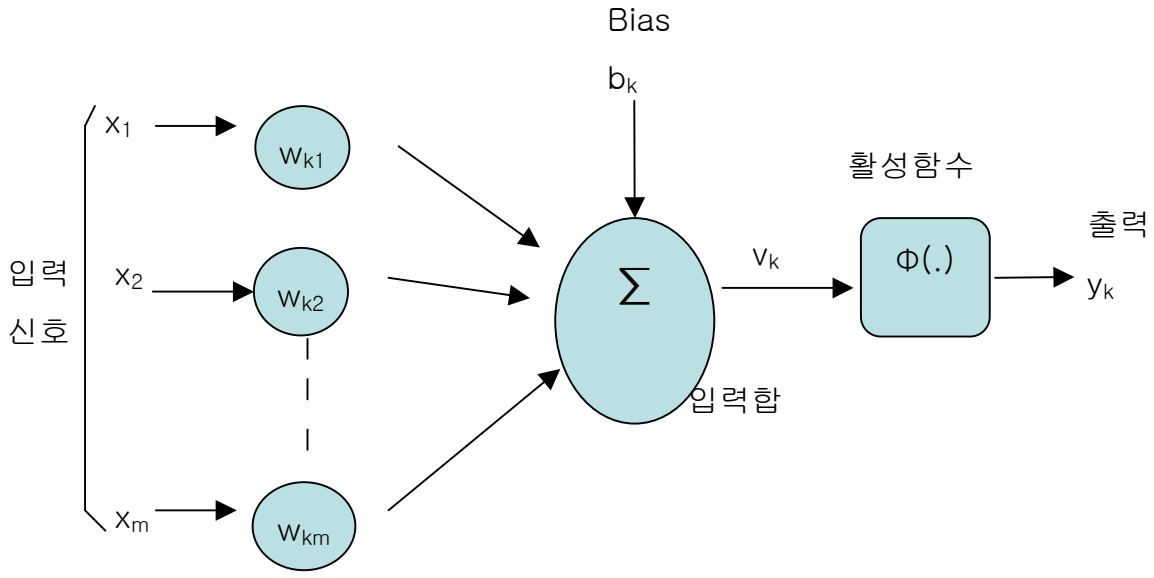
< 표5. 멜로디 패턴 2 >

1] 7 8 5 7 8 7 3 3 0 0 -4
2] 3 3 3 5 2 -2 2 2 0 0 3
3] 4 0 2 2 4 2 0 0 0 0 0
4] 2 4 5 -3 -3 4 4 2 0 -3 5
5] -2 -4 -4 -4 -2 0 1 3 5 7 8
…

3.3 인공 신경망 (Artificial Neural Networks)

인공 신경망이란 인간이나 동물의 뇌의 구조를 모방한 계산 모델이다[8]. 실제 생명체의 뇌는 단순한 일을 하는 개개의 뉴런들로 구성되는데, 이들은 천문학적인 수를 이루어 망을 형성하고 상호 작용하는 분산 병렬 방식의 처리과정을 통해 컴퓨터가 할 수 없는 고차원 적인 문제 해결을 가능케 한다.

신경망 모델은 각 뉴런의 기능, 망의 구조, 데이터에 따른 적응 알고리즘에 의해 구분된다. 모든 신경망은 기본적으로 <그림3>과 같이 하나의 뉴런에서 시작하며 이를 변형하거나 응용하여 만들어진다.



< 그림3. 인공 뉴런의 구조 >

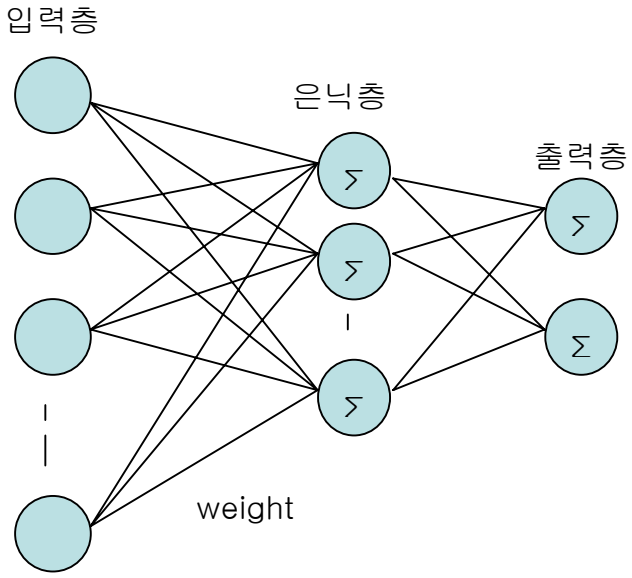
그림에서 x 값은 입력 신호들이며, 식(3.2.1)에서와 같이 Σ 기호 부분에서 x 값과 각각의 weight값인 w를 곱한 전체 합을 더하게 된다. 그리고 선택적으로 bias 값을 받아들이며, 식(3.2.2)에서처럼 활성함수(Activation Function)에서 처리를 통과하여 비로소 출력 값 y 로 나가게 된다. 활성 함수로는 threshold, piecewise-linear, sigmoid 등이 있다[9].

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \qquad (3.3.1)$$

$$y_k = \varphi(u_k + b_k) \qquad (3.3.2)$$

$$v_k = u_k + b_k \qquad (3.3.3)$$

일반적으로 널리 사용되는 것으로 다층 퍼셉트론 (MLP: Multi Layer Perceptron)이 있다.



< 그림4. 다층 퍼셉트론 >

다층 퍼셉트론은 비선형 활성함수를 가지고 있고, 은닉 층으로 불리는 중간층을 가지며, 각 층의 뉴런들은 인접한 층의 다른 모든 뉴런들과 상호 연결되어 있다는 특징을 지닌다.

$$e_j(n) = d_j(n) - y_j(n) \qquad (3.3.4)$$

$$E(n) = \frac{1}{2}\sum_{j=L} e_j^2(n), \quad E_{av} = \frac{1}{N}\sum_{n=1}^{N} E(n) \qquad (3.3.5)$$

학습은 출력 층의 에러 값이 최소가 되는 방향으로 이루어진다[10]. 식(3.3.4)는 j번째 출력 뉴런에서의 에러 값 e를 구하는 것으로 출력 값 y 와 학습 값 d 사이의 오차 값을 사

용한다. 그리고, 식(3.3.5)는 출력 층의 모든 뉴런에 대한 에러 값을 더하여 2로 나눈 후, 전체 학습 데이터 패턴에 대한 평균 에러 값을 구하는 과정이다.

실제 학습은 다음의 과정을 거친다. 먼저 입력 층의 뉴런은 입력 신호 값에 가중치를 곱한 후 은닉 층으로 보낸다. 은닉 층의 뉴런은 입력 층의 뉴런들로부터 받은 값들을 모두 더한 후 비선형 활성함수 처리를 하고 다시 가중치를 곱하여 출력 층으로 보낸다. 출력 층의 뉴런에서는 은닉 층의 뉴런들로부터 받은 값들을 모두 더하여 다시 활성함수 처리를 하여 결과를 내보내게 된다. 이렇게 이전 층에서 다음 층으로 값을 전달하는 방식을 순차 방식(feed-forward)이라고 한다. 이들 출력 층에서 나오는 결과 값들과 실제 우리가 값들을 비교하여 그 오차 값을 계산한 후, 신경망의 weight를 갱신해 주어야 하는데, 출력 층에서 입력 층까지 반대 방향으로 갱신 작업이 이루어 진다고 하여 오류 역 전파 방식(error-back propagation)이라 한다.

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha\left[w_{ji}^{(l)}(n-1)\right] + \eta\delta_j^{(l)}(n)y_i^{(l-1)}(n) \tag{3.3.6}$$

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n)\varphi_j'(v_j^{(L)}(n)) \\ \varphi_j'(v_j^{(L)}(n))\sum_k \delta_k^{(l+1)}(n)w_{kj}^{(l+1)}(n) \end{cases} \tag{3.3.7}$$

$$\varphi_j(v_j(n)) = \frac{1}{1+\exp(-av_j(n))}, a > 0 \,\&\, -\infty < v_j(n) < \infty \tag{3.3.8}$$

$$\varphi_j'(v_j(n)) = ay_j(n)\left[1 - y_j(n)\right] \tag{3.3.9}$$

각 층의 오류 역 전파를 통한 가중치(weight) 갱신은 식(3.3.6) 같이 이전 층의 i번째 뉴런과 현재 층 l(중간 층) 또는 L(출력 층)의 j번째 뉴런에 대한 계산으로 이루어 지며, 이렇게 나온 값으로 식(3.3.7)를 이용하여 에러 값 δ를 계산하게 된다. δ값을 구하기 위해서는 활성함수가 미분 가능해야 하는데, Logistic function 또는 Hyperbolic tangent function 등이 사용된다. 식(3.3.8)과 식(3.3.9)는 전자에 해당하는 시그모이드 함수(sigmoid function)의 원형 식과 미분 식을 보여주고 있다.

실제 학습에 있어서 크게 순차적(sequential)인 방법과 일괄적(batch)인 방법이 있다. 순차적인 방법은 개개의 입력 값들에 대한 에러 값을 계산하여 바로 가중치를 갱신하는 방식으로 구현이 간편하다는 점과 방대하고 복잡한 패턴을 학습하는데 상대적으로 신속한 장점이 있다. 반면 일괄적인 방법은 모든 입력 패턴의 값을 처리한 후 마지막에 한번 에러 값을 통한 갱신을 해주는 방식으로 비교적 적은 양의 데이터를 빠르게 처리할 수 있는 장점이 있다. 식(3.3.5)는 순차 학습 방법의 에러 계산 식이며, 식(3.310)은 일괄 학습 방법

의 에러 계산 식이다.

$$E_{av} = \frac{1}{2N}\sum_{n=1}^{N}\sum_{j=C} e_j^2(n) \tag{3.3.10}$$

그 외, 패턴을 효율적으로 학습하기 위해서 입력 데이터를 정규화(Normalize)시키는 방법도 사용되며, 학습 시 에러의 수렴 속도를 빠르게 하기 위하여 모멘텀 항의 추가, 학습계수 최적화, 선택적 재 학습 방법 등이 동원된다[11].

이번 연구에서 사용한 구체적인 신경망 알고리즘의 순서는 다음과 같다.

1) 네트워크의 상태를 결정하는 연결강도 w 는 random(-0.5,0.5), offset 은 random(-0.05,0.05)로 초기화.
2) 훈련 패턴 값을 정규화 하여 읽음.
3) 훈련 패턴 값을 입력뉴런에 넣어서 출력되는 값 o, 입력뉴런과 은닉뉴런 사이의 연결강도 w와 은닉뉴런의 offset 을 이용하여 은닉뉴런의 입력 net을 구한다. 다음으로 net과 시그모이드 함수 f를 이용하여 은닉뉴런의 출력 o 를 구한다.

$$net_{pj} = \sum_i w_{ji} o_{pi} + \theta_j \ , \ o_{pj} = f_j(net_{pj})$$

4) 은닉뉴런의 출력 o, 은닉뉴런과 출력뉴런 사이의 연결강도 w와 출력뉴런 offset을 이용하여 출력뉴런의 입력 net 을 구한다. 다음 net와 시그모이드 함수 f를 이용하여 출력뉴런의 출력 o를 구한다.

$$net_{pk} = \sum_j w_{kj} o_{pj} + \theta_k \ , \ o_{pk} = f_k(net_{pk})$$

5) 훈련 세트의 목표 출력 t와 실제 출력 o와의 차로부터 출력뉴런에 연결된 연결강도와 출력뉴런의 offset에 대한 오차 d를 구한다.

$$\delta_{pk} = (t_{pk} - o_{pk}) o_{pk} (1 - o_{pk})$$

6) 오차 d와 함께 은닉뉴런과 출력뉴런 간의 연결강도 w와 은닉뉴런의 출력 net으로부터 은닉뉴런에 연결된 연결강도와 은닉뉴런의 offset에 대한 오차 d를 구한다.

$$\delta_{pj} = \sum_k \delta_{pk} w_{kj} o_{pj} (1 - o_{pj})$$

7) 5단계에서 구한 출력뉴런에서의 오차 d, 은닉뉴런의 출력 o, 상수 알파와의 곱을 더하여 은닉뉴런과 출력뉴런에 연결된 연결강도 w를 수정한다. 또 오차 d와 정수 베타와의 곱을 더하여 출력뉴런의 offset을 수정한다.

$w_{kj} = w_{kj} + \alpha \cdot \delta_{pk} o_{pj}$, $\theta_k = \theta_k + \beta \cdot \delta_{pk}$

8) 은닉뉴런의 오차 d, 입력뉴런의 o, 상수 알파와의 곱을 더하여 입력뉴런과 은닉뉴런에 연결된 연결강도 w를 수정한다. 또 오차 d와 상수 베타의 곱을 더하여 은닉뉴런의 offset을 수정한다.

$w_{ji} = w_{ji} + \alpha \cdot \delta_{pj} o_{pi}$, $\theta_j = \theta_j + \beta \cdot \delta_{pj}$

9) 다음 훈련 패턴을 학습시킨다.

10) 모든 훈련 패턴에 대하여 전부 학습할 때까지 2단계로 되돌아 간다.

11) 학습의 반복 회수를 센다. 기본적인 학습 반복 회수는 10,000 번 이다.

12) 학습의 반복횟수가 제한 횟수보다 작으면 2단계로 돌아간다.

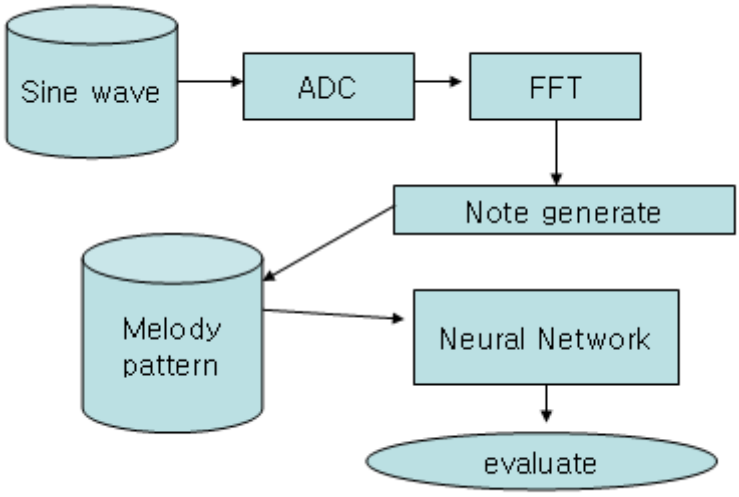
13) 이렇게 학습된 뉴런을 기존 훈련 패턴에 대해 테스트 해본다.

14) 훈련 패턴대신 다른 집합인 검사 패턴에 대해서 테스트 해본다.

15) 결과를 기록한다.

3.4 구현한 시스템 구조

이번 연구에서 사용한 데이터 및 처리기들의 전체 시스템을 <그림.5>에 나타내었다.



< 그림5. 전체 시스템 구조 >

전체적으로 다음의 과정으로 진행된다.

1) 멜로디 패턴에 해당하는 사인 파(sine wave) 생성

2) ADC를 통해 discrete wave data 값으로 표현

3) FFT를 통해 spectrum 분석

4) 평균율을 적용하여 음(note) 추출

5) 생성된 멜로디 패턴을 이용하여 신경망 학습

6) 학습된 신경망을 평가

Ⅳ. 실험 및 결과

4.1 실험 내용

실험은 크게 두 가지 주제에 대하여 행한다.

첫 번째는 멜로디 패턴의 학습 후 검색 멜로디 패턴에 대한 정확한 응답을 보이는지 평가하는 것이다. 멜로디 패턴 데이터를 분석하면 11개의 입력 값들이 가지는 패턴이 모두 다르기 때문에, 이들을 신경망에 기억시킨 후 어느 정도의 오차에서도 올바른 패턴을 인식할 수 있는지 알아본다.

두 번째는 신경망이 멜로디 패턴의 특징을 인식할 수 있는지를 알아보는 것이다. 슬플 멜로디와 즐거운 멜로디 두 가지로 분류 학습 시킨 후, 학습 시 사용하지 않았던 새로운 멜로디 패턴에 대한 평가를 행하는 것이다.

4.2 실험 방법

첫 번째 실험에서는 100개의 멜로디 패턴을 모두 기억 시킨 후, 기존의 패턴 값을 약간씩 변형 생성하여 인식 율을 평가하였다. 이는 실제 사람이 멜로디를 입력하는 과정에서 중간에 다소 틀린 음높이를 넣게 되더라도 원하는 결과를 얻을 수 있을지 알아보는 것이다. 따라서 틀린 음은 정상 음에 비해 한두 음 정도가 높거나 낮게 입력되는 결과를 낳기 때문에, 이를 고려하여 오차 패턴 데이터를 생성하였다. 우선, 정상 패턴이 가지는 11개의 값 중에서 하나 또는 둘 정도를 임의로 지정하여 1~3 정도의 오차 값으로 변경하여 전체적으로 새로운 패턴 데이터를 생성하였다. 그리고, 각 1~11번째 입력에 대해서만 오차 값을 생성하여 몇 번째 음이 에러 율이 높은지도 알아보았다.

첫 번째 실험의 신경망 학습은 입력 뉴런 수 11, 은닉 뉴런 수 가변, 출력 뉴런 수 7, 30, 60, 100, 그리고 30+n으로 설정하였다. 출력 뉴런 수를 7로 설정한 것은 100개의 데이터에 대한 은닉 뉴런의 결과가 이진수로 표현되도록 한 것이고, 출력 뉴런 수를 30, 60, 100으

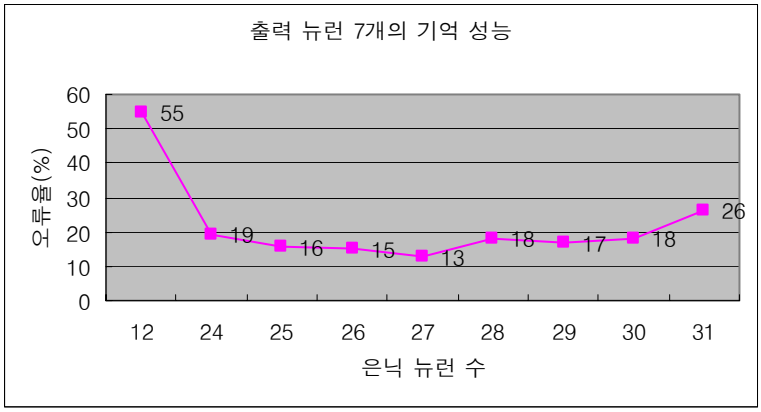
로 한 것은 입력 뉴런 패턴 각각이 하나의 출력 뉴런을 가지는 효과를 위한 것이다. 이는 경쟁 층을 이용하는 SOM(Self Organizing Map)과 비슷한 의도라고 할 수 있다. 30+n 방식은 60개의 패턴에 대하여 뉴런 수 7의 이진 방식과 뉴런 수 30, 60, 100의 1-1 방식을 혼합하여 어느 정도 효과가 있는지를 알아보기 위한 것이었다. 출력 뉴런 수 7의 경우에는 각각의 출력 뉴런의 값이 0.5를 기준으로 높으면 1, 낮으면 0으로 평가하였고, 출력 뉴런 수 100의 경우에는 100개의 출력 뉴런 값 중 가장 높은 값을 가지는 출력 뉴런이 1을 가지도록 하여 실제 값과 비교하였다. 이는 멜로디 패턴의 기억 학습이 패턴 경향에 따른 분류가 되지 않기 때문에 적은 수의 출력 뉴런으로는 한계가 있을 것으로 판단했기 때문이다.

두 번째 실험에서는 100개의 멜로디 패턴을 슬픈 것과 즐거운 것의 두 가지로 구분하여 75개의 훈련 패턴과 25개의 실험 패턴에 대한 인식률을 평가했는데, 훈련 및 실험 데이터의 임의 구분이 인식률에 영향을 줄 것을 고려하여, 100개의 패턴을 순서대로 25개씩 실험 패턴으로 설정하여 네 가지 훈련 및 실험 데이터를 생성하였다. 또한 멜로디 패턴의 요소 중 하나인 음 길이를 추가하여 주었을 때의 인식률도 평가하였다.

두 번째 실험의 신경망 학습은 입력 뉴런 수 11, 은닉 뉴런 수 가변, 출력 뉴런 수 2로 설정하였다. 패턴이 가지는 특징에 따라 01 또는 10으로 각각의 출력 뉴런이 발현(fire)되도록 의도한 것이다. 음 길이를 고려한 신경망 학습은 입력 뉴런 수 11+12, 은닉 뉴런 수 h1 + h2, 출력 뉴런 수 2로 설정하여 음높이 관련 은닉 뉴런 수 h1 과 음 길이 관련 은닉 뉴런 수 h2 의 값을 변경하면서 성능을 측정하였다.

4.3 첫 번째 실험 결과 및 분석

먼저 출력 뉴런이 7개인 경우 신경망 기억 학습 능률을 알아 보았다.



< 그림6. 출력 뉴런 수 7 의 학습 정도 >

<그림6>은 입력 11, 은닉 가변, 출력 7 에서 20,000 번 학습 시킨 후의 오류 율을 보인 것이다. 그림에서 보듯 7개의 출력 뉴런으로는 은닉 뉴런 수 27 에서 100개의 입력 패턴 중 최대 87개의 패턴만을 제대로 기억할 수 있다. 은닉 뉴런 수를 고정하고 학습 회수를 30,000번 50,000번 200,000번으로 증가시켜 보았으나 개선되지는 않았다. 이는 신경망이 특성상 이산적인 데이터 구분에 취약하기 때문인 것으로 생각된다. 출력 뉴런 수를 7개로 시작한 것은 입력 패턴의 번호에 따라 출력 뉴런이 이진수의 형식으로 표현되도록 하기 위한 것인데, 예를 들면 17번째 패턴의 출력 뉴런 값은 0010001 이 된다. 이는 뇌의 구조를 모델링 한 신경망에서는 비록 컴퓨터 기반이더라도 사람의 경우와 같이 번호 매김이 어렵다는 것을 알 수 있다. 또한 일반적으로 신경망 학습에서 은닉 층의 뉴런 수는 입력 층의 뉴런 수 보다 같거나 적은 경향을 보이는데, 여기서는 오히려 두 배 정도 많은 수의 은닉 뉴런이 사용되는 것을 알 수 있다. 때문에, 출력 뉴런의 수를 증가시키더라도 오류 율을 개선 시킬 필요성이 생겼고, 우선 30개의 패턴에 대하여 30개의 출력 뉴런을 할당하여 해당 패턴에 대해 단 하나의 출력 뉴런만 1의 값을 가지도록 학습시켰다.

< 표6. 출력 뉴런 수 30의 경우 오류 >

은닉뉴런 수	6	7
오류 수/총 수	1/30	0/30

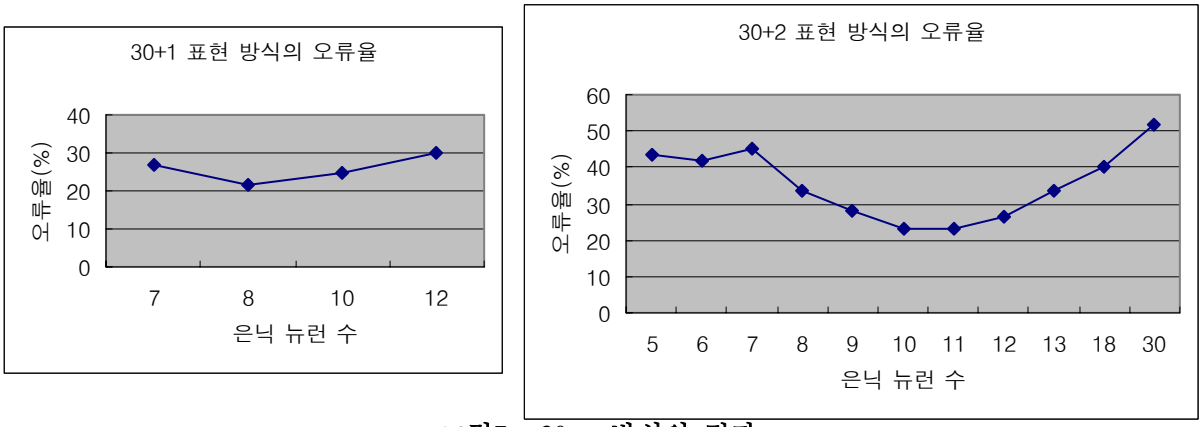
그 결과, 30개의 입력 패턴에 은닉 뉴런 수 7 에서 100%의 기억 율을 보였다. 이에 60개의 입력 패턴에 대하여 추가적인 실험을 행하였는데, 이번에는 앞의 이진 수 표현 방식과 지금의 1-1 표현 방식을 혼합하였다.

< 표7. 출력 뉴런의 표현 방식 >

패턴번호	30+1 표현 방식	30+2 표현 방식
1번째	0 0000...0001	01 0000...0001
2번째	0 0000…0010	01 0000…0010
…	…	…
29번째	0 0100…0000	01 0100…0000
30번째	0 1000…0000	01 1000…0000

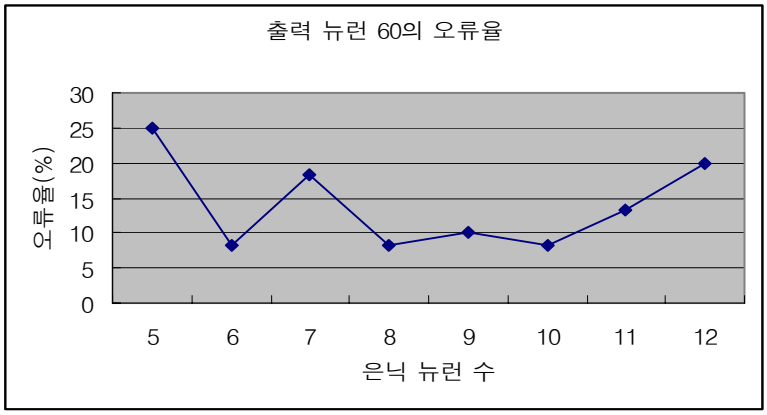
31번째	1 0000…0001	10 0000…0001
…	…	…
60번째	1 1000…0000	10 1000…0000

두 가지 표현 방법을 모색하였는데, 하나는 30+1 개의 출력 뉴런을 사용하여 하나의 출력 뉴런을 이진수 방식으로 표현하도록 하는데, 앞의 30개씩에 대하여 0을 출력 하고 뒤의 30개씩에 대하여 1을 출력 시켰다. 다른 하나는 30+2 의 표현 방식으로 앞의 30 개의 패턴에 대하여 두 개의 추가 뉴런을 0 1로 표시하고, 뒤의 30 개 패턴에 대하여 1 0으로 표시하였다.



< 그림7. 30+n 방식의 결과 >

실험 결과 기대와는 달리 30+n 방식은 처음의 이진수 표현 방식보다도 낮은 학습 정도를 나타내었다. <그림7>을 보면 60개의 패턴에 대하여 가장 좋은 경우 20% 정도의 오류율을 보여준다. 이에 60개의 입력 패턴에 대하여 60개의 출력 뉴런을 테스트 하였다.



< 그림8. 60개의 출력 뉴런을 사용한 결과 >

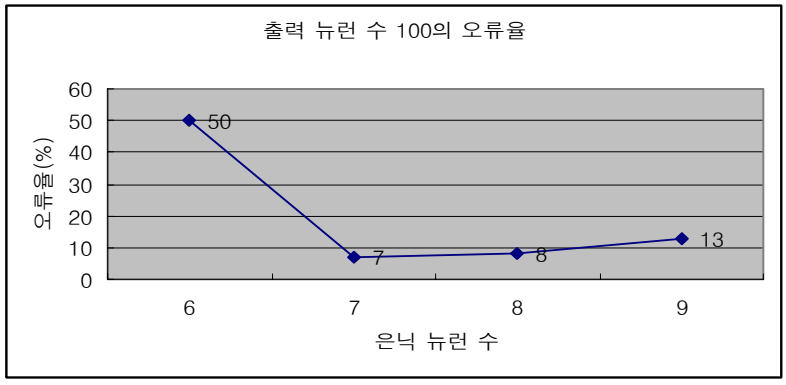
실험 결과 은닉 뉴런 수 6과 8에서 8% 정도의 오류 율을 보이는데, 여기서 학습 율을 좀 더 높이기 위하여 은닉 뉴런 수 8에 대하여 학습 회수를 높여 실험하였다.

< 표8. 은닉 뉴런 수 8에서 학습 회수에 따른 오류 율 >

20,000번	30,000번	50,000번	100,000번
5%	5%	5%	5%

<표8>에서 보듯 학습 회수가 증가하면서 초반에 어느 정도 오류 율에 영향을 주지만, 학습 회수가 아주 커지게 되면 그다지 큰 영향이 미치지 않는 것을 확인할 수 있다.

다음은 100개의 입력 패턴에 대해 100개의 출력 뉴런을 사용한 결과이다.



< 그림9. 100개의 출력 뉴런을 사용한 결과 >

<그림9> 는 학습 회수 10,000 번의 경우 오류 율을 보인 것이다. 은닉 뉴런이 7개인 경우 7%로 가장 높은 효율을 보이는데, 학습 회수를 100,000 번 이상으로 증가시켰더니 5% 까지 낮아졌다.

출력 뉴런의 표현 방식에 따른 몇 가지 경우에 대한 실험 결과를 정리하면, 이진 수 방식이 출력 뉴런 수는 줄일 수 있으나 많은 은닉 뉴런 수가 필요하며 학습 기억 율이 그리 높지 않은데 반해, 1-1 방식은 많은 출력 뉴런 수를 사용하는 문제가 있으나 높은 학습 기억 율을 보여주는 것을 알 수 있었다.

이를 고려하여 오차를 포함한 패턴에 대한 인식 율을 검사할 때 오류 율이 낮은 1-1 방식을 사용 하였다. 그리고, 학습 회수에 따른 인식 율을 알아보기 위해 10,000 번, 50,000

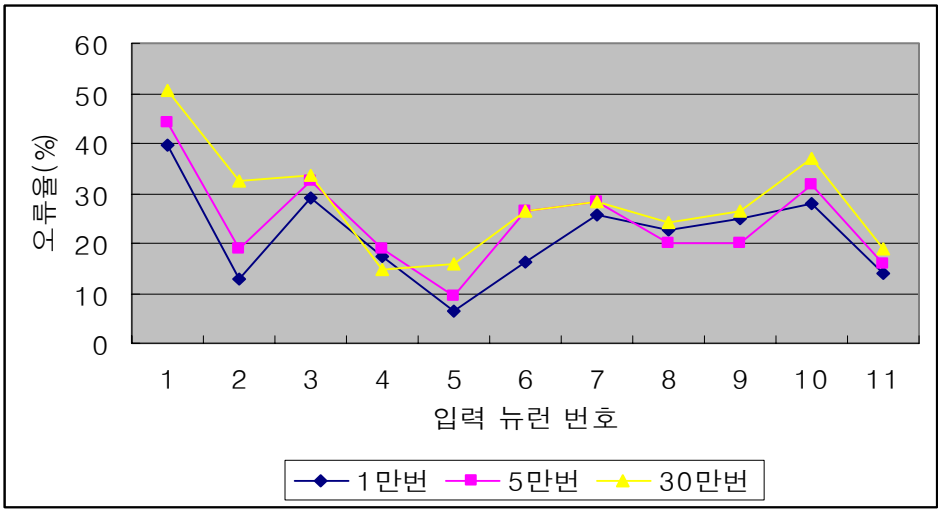
번, 300,000 번의 학습 회수에 대한 결과를 알아 보았다.

< 표9. 1-point 오차 패턴에 대한 결과 >

학습 회수	10,000	50,000	300,000
오류 개수/전체 개수(오류 율(%))	14/93(15%)	21/95(22.1%)	29/95(30.5%)

<표9> 는 패턴의 11개 입력 값 중 임의의 한 지점에 대해 +/- 1 만큼의 오차가 생기도록 하여 오류 율을 알아본 것이다. 그 결과 학습 회수 10,000 개의 경우 기본적인 훈련 패턴의 인식에서 93개를 인식하고, 해당 93 개의 오차 검사 패턴에 대해 14개를 인식 하지 못함으로써 85% 수준의 오차 패턴 인식률을 보인다. 300,000 번의 학습 회수의 경우 훈련 패턴 95개를 인식하고, 그 중 29 개의 오차 생성된 패턴에 대해 인식하지 못하여 69.5% 수준의 오차 패턴 인식률을 보인다.

다음은 11개의 입력의 n 번째 데이터만 +1 또는 -1 하여 오차 패턴을 생성하였을 때의 결과를 알아보았다.



< 그림10. n 번째 입력 값의 오차에 대한 결과 >

<그림10>을 보면 학습 회수가 클수록 오류 율도 대체적으로 높은 것을 알 수 있다. 그리고 특정 지점, 예를 들면 4, 5, 11번째의 입력 값에 대해 비교적 낮은 오류 율을 보이고 있는데, 이는 훈련 멜로디 패턴의 한 소절이 5~6 정도에서 끝나는 경우가 많다는 것과 관련이 있을 것으로 생각된다. 하지만 일반적인 경우에 적용될 수 있을 지는 판단할 수 없다.

다음으로 임의의 지점에 대한 오차를 +/-2 와 +/-3 에 대하여 실험한 결과를 알아보았다.

< 표10. +/-2, +/-3 에 대한 오차 패턴 검사 결과 >

회수	10,000번	50,000번	300,000번
+/- 2	48/93(51.6%)	46/95(48.4%)	43/95(45.3%)
+/- 3	55/93(59.1%)	53/95(55.8%)	56/95(58.9%)

<표10>에서 보면 +/-1 의 경우에 비해 전체적으로 오류 율이 50% 수준으로 높은 것을 알 수 있다. 각 학습 회수의 따른 오류 율의 차이는 크게 나지 않는다.

이번에는 11개의 입력 값 중 임의로 2~3 개를 추출하여 +/-1 의 오차를 주어 패턴을 생성시키고 결과를 알아보았다.

< 표11. 2-point, 3-point 에 대한 오차 패턴 검사 결과 >

회수	10,000번	50,000번	300,000번
2-point	21/93(22.6%)	34/95(35.8%)	32/95(33.7%)
3-point	28/93(30.1%)	32/95(33.7%)	36/95(37.9%)

여기서는 대체로 30% 정도의 오류 율을 보이며 오차 지점이 늘어날수록 오류 율도 증가하는 것을 알 수 있다. 학습 회수는 10,000 번의 경우가 다른 학습 회수에 비해 다소 낮은 값을 보여주고 있다.

마지막으로 임의적인 오차가 아닌, 실제 사용자가 음성을 통해 패턴을 검색 할 경우 생길 수 있는 오차를 고려하여 만든 실험 패턴의 결과를 알아보았다.

< 표12. 발생 가능한 오차에 대한 실험 결과 >

학습 회수	10,000	50,000	300,000
오류 개수/전체 개수(오류 율(%))	35/93(37.6%)	42/95(44.2%)	43/95(45.3%)

이 경우 40% 안팎의 오류 율을 나타내고 있으며, 역시 10,000번의 학습 회수의 경우가 오류 율이 상대적으로 낮게 나오는 것을 알 수 있다.

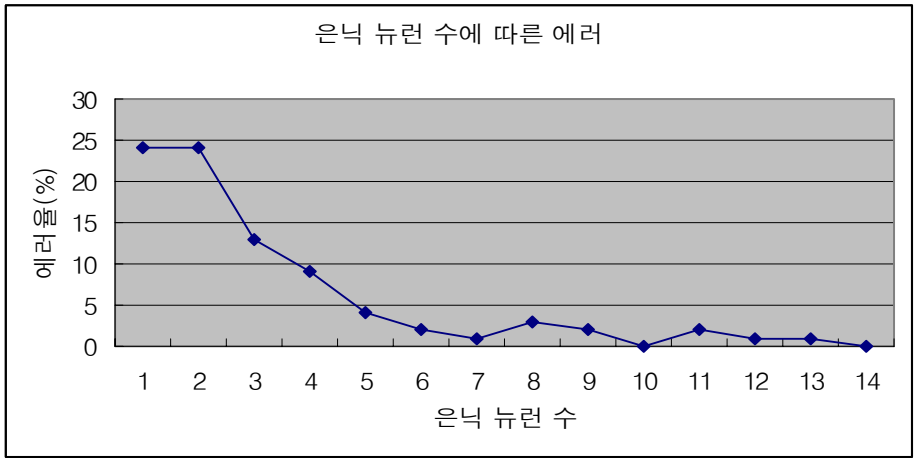
오차에 따른 인식 결과를 종합적으로 정리해 보면 오차 개수(point)가 많아지면 인식률이

떨어지게 되며, 오차의 값이 커질 경우에도 마찬가지로 인식률이 떨어지게 되는 것을 알 수 있다. 또한 학습 회수가 높으면 훈련 패턴에 대해서 최적화 됨에 따라 오차를 가진 검사 패턴에 대한 인식률이 오히려 떨어지는 결과를 초래한다는 점도 알 수 있다. 따라서 학습 패턴에 대한 충분한 인식과 오차 패턴에 대한 만족스러운 인식을 보이기 위한 최소한의 학습 회수가 필요할 것으로 보인다.

4.4 두 번째 실험 결과 및 분석

먼저 100개의 입력 패턴 모두를 인식 시켜보는 실험을 하였다.

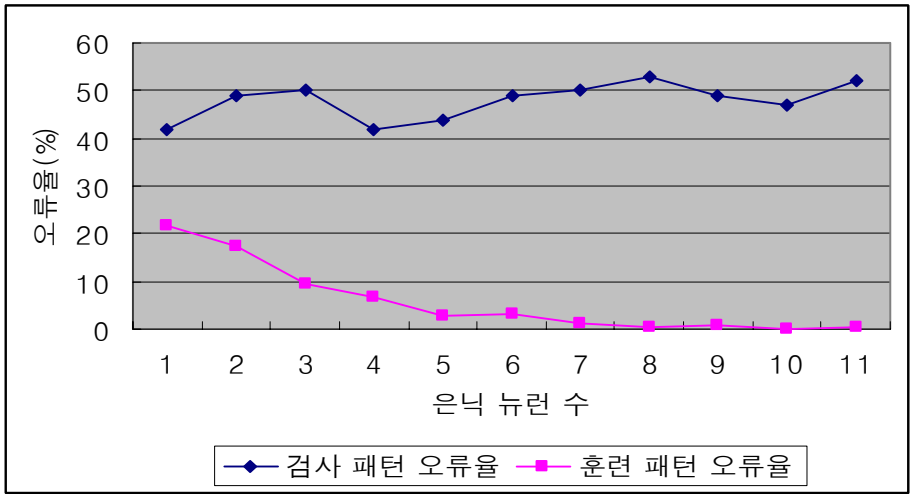
학습 회수는 20,000 번으로 설정하였다.



< 그림11. 은닉 뉴런 수에 따른 100개의 입력 패턴 인식 >

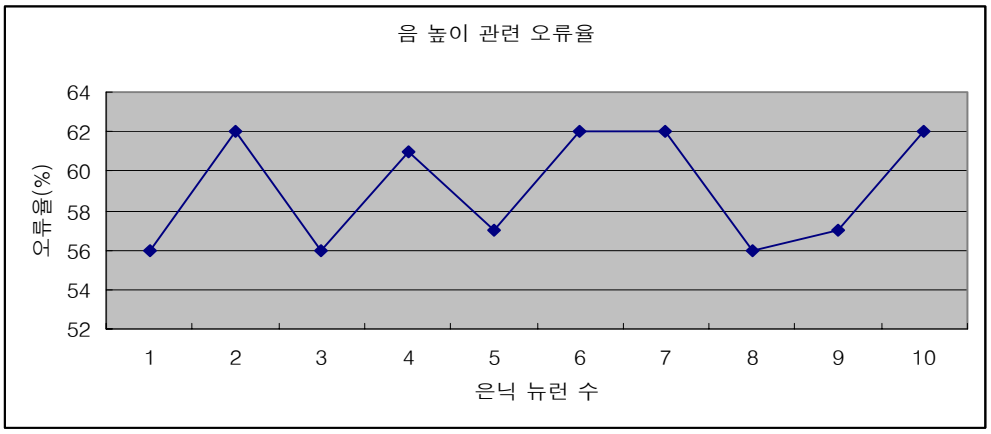
그 결과 은닉 뉴런 수 11, 15 에서 100개의 패턴 모두를 제대로 분류하였다.

다음으로 패턴 100개 중 두 집단으로 나누어 75개는 학습 용으로 사용하고, 25개는 검사용으로 사용하여 실험 하였다. 평균적인 성능 결과를 알아보기 위해, 100개를 25개씩 네 집단으로 분리하여 해당 집단 마다 25개에 속하지 않는 나머지 75개를 학습용으로 사용하고 그 결과를 합한 후 평균하였다. 첫 번째 실험에서 학습 회수가 적을수록 새로운 데이터에 대한 적응력이 높다는 점도 고려하여 학습회수를 처음의 20,000번 보다 적은 10,000 번으로 설정하였다.

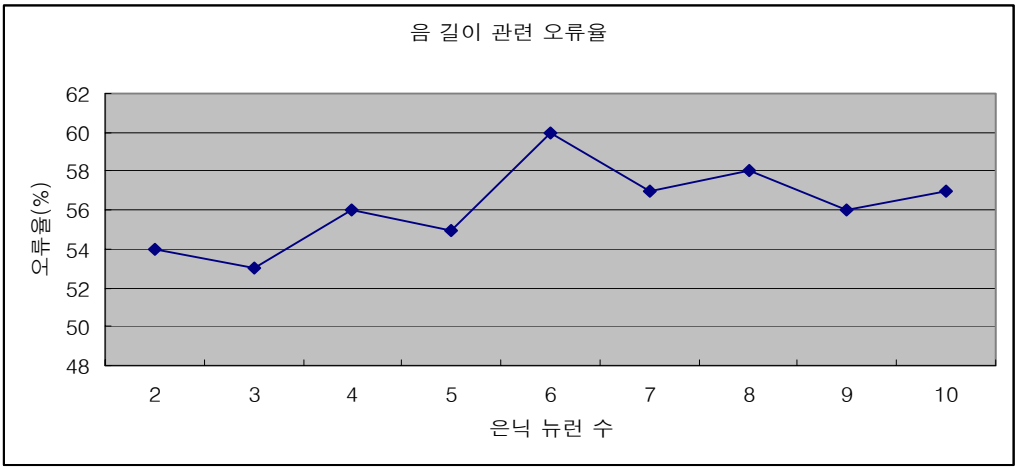


< 그림12. 은닉 뉴런 수에 따른 분류 성능 결과 >

<그림12>를 보면 은닉 뉴런수가 증가함에 따라 훈련 받은 75개 패턴에 대한 분류 능력이 증가하는데 비해 25개의 검사 패턴에 대해서는 꾸준히 50% 정도 분류 능력을 보여주고 있다. 실제 효과적인 사용을 위해서는 95% 이상의 분류 성능이 필요하다는 점을 감안하면 상당히 낮은 수치임을 알 수 있다. 때문에 이 실험에 대해서는 별도로 음 높이와 그에 따른 음 길이를 함께 추가하여 신경망의 입력 정보를 추가해 보았다. 음 길이는 패턴 내의 상대적인 음 길이와 패턴의 빠르기를 고려한 숫자 값을 사용하였다. 그리고, 이를 학습시키는 데에는 부분 연결 방식(partially connected)의 신경망을 사용하였는데, 여기서는 음 높이 입력 값과 음 길이 입력 값이 각각의 은닉 뉴런을 사용하고 이들 은닉 뉴런이 바로 출력 뉴런으로 연결되도록 하였다.



< 그림13. 음 높이 관련 은닉 뉴런 수 에 따른 오류 율 >



< 그림14. 음 길이 관련 은닉 뉴런 수 에 따른 오류 율 >

<그림13> 은 음 길이에 관련된 뉴런 수를 1로 고정한 상태에서 음 높이에 관련된 뉴런의 수를 증가 시킴에 따른 오류 율을 보이고 있으며, <그림14>는 <그림13>에서 얻은 가장 낮은 오류 율의 음 높이 관련 은닉 뉴런에서 음 길이 관련 은닉 뉴런을 증가 시킴에 따른 오류 율을 보인 것이다. 결과에서 보듯 음 길이를 추가한 경우에 분류 성능이 오히려 더욱 떨어진 것을 알 수 있다. 이는 음 높이와 깊이를 가진 멜로디 만으로는 신경망을 통해 곡이 슬픈지 즐거운지 분류하기가 상당히 어렵 다는 것으로, 추가적으로 곡의 빠르기 및 화성학적 지식 등에 대한 정보가 필요할 것으로 보인다.

Ⅴ. 결론

5.1 연구 결과 요약

이 논문에서는 멜로디 패턴 인식을 컴퓨터에서 어떻게 처리할 것인가에 대하여 살펴보고, 이를 분류, 검색하는데 신경망의 사용이 효과적인지를 실험해 보았다.

사인 파로 생성한 멜로디 패턴을 컴퓨터가 다룰 수 있는 데이터로 변환시키기 위해 ADC 기법이 사용되었으며, 주파수 분석을 행하기 위해서는 FFT 기법이 필요하였다. 주파수 값으로부터 음 높이를 구하는 데에는 평균율의 지식이 필요하였고 이를 반복하여 하나의 완전한 패턴을 얻을 수 있었다.

실험 결과를 보면, 첫 번째 실험의 경우 다층 퍼셉트론 구조의 신경망을 이용한 멜로디 패턴의 기억 학습이 생각과는 달리 100%의 인식률을 보여주지 못하는 것을 알 수 있었

다. 학습에 사용한 패턴이 첫 음에 대한 오차 값을 가지도록 함으로써 다른 패턴들 사이의 값들이 가지는 유사성이 높아졌고, 따라서 신경망 학습 시 이를 확연하게 인식시키는 것이 쉽지 않는 것으로 보인다. 이는 인간의 경우도 마찬 가지로 비슷한 멜로디 패턴에 대해 혼동을 일으킬 확률이 높다는 것을 고려하면 이해할 수 있겠다.

인식률을 높이기 위해서 출력 뉴런의 표현 방식을 몇 가지 시도하였으며, 이진수와 같은 컴퓨터적인 표현 보다는 인간의 사고 방식에 가까운 1-1 방식이 더 효율이 높다는 점을 알 수 있었다.

오차를 가진 패턴에 대한 실험 결과, 학습 회수에 따라 오차 패턴에 대한 오류 율이 달라지는 것을 볼 수 있었고, 가급적 학습 회수를 높이지 않는 것이 오차에 강한 결과를 나태 내는 것을 알 수 있었다.

두 번째 실험에서는 멜로디 패턴이 슬픈 곡과 즐거운 곡으로 분류될 수 있는지 알아보았으나, 실험 결과 인식률이 50% 수준에 머무는 불만족스러운 결과를 보여주었다. 이는 실험에 사용한 멜로디 패턴이 가진 음 높이와 음 길이의 순수 데이터 만으로는 신경망이 분류 작업을 효과적으로 하는데 한계가 있는 것으로 보인다. 따라서 멜로디의 화성적 지식과 신경망의 구조 변경이 필요할 것이다.

5.2 연구 결과 활용 방안

논문에서 제시된 멜로디 패턴 인식 시스템은 실용적으로 활용될 수 있는 여지가 있다. 굳이 곡의 제목이나 가사를 모르더라도 곡의 멜로디를 입력하는 것으로 원하는 곡을 찾을 수 있는데, 특히 입력 부분에서 sine wave 대신 사람의 목소리로 입력하도록 처리해 주면 가정용 컴퓨터에 마이크를 연결하는 것만으로 쉽게 검색이 가능하게 된다. 이것을 기존의 인터넷 검색엔진에 적용할 경우 멜로디를 통한 곡 검색이 가능하게 되고, 제목과 가사 검색을 동반할 경우 더욱 정확한 곡 검색이 가능할 것이다. 또한, 노래방(karaoke) 시스템에 응용될 경우 원하는 곡을 부르기 위해 곡 번호를 일일이 찾아야 할 필요 없이 마이크에 멜로디를 입력해 주는 것만으로 쉽게 곡을 찾을 수 있게 된다.

한편 멜로디의 분위기를 인식시키는 문제는 인공 신경망이 감정적 처리도 다룰 수 있는지를 알아볼 수 있는 연구 분야로, 인간의 뇌에 더욱 가까운 인공 신경망 구조를 모색할 수 있는 기반 자료로서 활용할 수 있다. 또한 앞의 검색과 마찬가지로 감정적 주제에 대한 분류 검색용도로도 충분히 사용가치가 있을 것이다.

5.3 향후 과제

이번 실험에서는 멜로디 패턴의 자료화 방법과 더불어 이를 검색하는 방법으로 신경망이 어느 정도 실용성이 있는지를 알아보았다. 멜로디 패턴은 ADC, FFT, 평균율의 지식을 사용하여 원하는 멜로디 음을 추출해 낼 수 있으며, 향후 입력 부분에서 음성 인식이 가능토록 하여 음 높이, 음 길이뿐만 아니라 가사까지 한번에 입력, 검색이 가능하도록 구현하면 다목적 검색 시스템으로도 사용이 가능할 것이다.

다음 연구에는 신경망을 이용한 멜로디 패턴을 기억 학습에 있어 수백~수천 개 이상의 곡에 대하여 100%에 가까운 학습 율을 구현할 수 있어야 하며, 동시에 검색 입력 과정에 생길 수 있는 오차 값에 대한 인식 능력을 높일 수 있는 신경망 구조에 관한 연구가 필요하다. 또한 일부분의 멜로디 패턴이 아닌 곡 전체의 멜로디를 자료구조화 하여 저장하고 이를 어떻게 효율적으로 검색할 지에 관한 연구도 고려해 봐야 할 것이다.

다른 한편으로 검색을 위해 입력한 패턴과 저장되어 있는 자료 패턴들의 값들을 일대일 검사 하여 일정 값 이하의 오차가 나는 곡들을 추천하는 방식도 고려해 봄 직하다. 이를 신경망 검색과 함께 사용하면 더욱 높은 검색 정확도를 가질 수 있을 것이다.

참고 문헌

[1] 박관우 & 안정모 옮김, 음악을 위한 음향학, 삼호출판사, 1990, pp. 138

(원저: Donald E. Hall, Musical Acoustics An Introduction)

[2, 3] Thomas D. Rossing, The Science of Sound 2nd edition, Addison Wesley, 1989, p.40, pp.171-178

[4] Sophocles J. Orfanidis, Introduction to Signal Processing, Prectice Hall, 1996, p.77

[5] Oppenheim & Schafer & Buck, Discrete-Time Signal Processing 2nd edition, Prentice Hall, 1999, pp.187-193

[6, 7] Sophocles J. Orfanidis, Introduction to Signal Processing, Prentice Hall, 1996, pp.483-490, pp.513-520

[8] 장병탁, 신경망, Natural Sciences, Vol. 8, 2000.

[9, 10] Simon Haykin, Neural Networks a comprehensive foundation 2nd edition, Prentice Hall, 1999, pp.10-13, pp.156-173

[11] 김대수, 신경망 이론과 응용(I), 하이테크 정보, 1992, pp.110-117