음악과 연결주의 : 컴퓨터 음악의 새로운 개념

전 지 호
〈서울대학교 음악대학 강사〉

차 례

들어가기

20세기 음악에서의 혁신 중 하나는 컴퓨터의 도입일 것이다. 어느덧 '컴퓨터 음악'이라는 말이 지극히 일상적 용어가 되어버렸다. 1950년대 중엽 이후 컴퓨터는 주로 작곡의 여러 절차들을 도와주는 역할부터 출발했는데, 당시만 해도 음악가들의 일반적 관심을 도출하는 데는 별로 성공적이지 못했다. 그러나 오늘날은 PC의 보급으로 컴퓨터에 대한 음악가들의 관심이 차츰 증대되어 우리나라에서도 기성 작곡가들 뿐 아니라 음악대학 학생들도 다양한 프로그램들을 사용하고 있다. 크게 나누어 악보를 사보하기 위한 이른바 Notation 프로그램들과 연주를 위한 Sequencer 프로그램들에 대한 활용도가 높다. 그러나 이런 프로그램들은 컴퓨터의 막강한 능력에 비추어 볼 때, 그 가능성을 충분히 활용하지 못하는 만족스럽지 못한 것들이라 할 수 있다. 작곡을 위한 다양한 필터로부터 컴퓨터를 사용한 자동화된 작곡에 이르기까지 더 큰 가능성들이 거의 무한으로 열려있다고 보아 무리가 없다.

그러나 음악가들이 컴퓨터에 대해 모두 호의적인 것은 아니다. 많은 음악가들이 컴퓨터 음악을 듣고 실망하는 까닭은, 물론 기대가 큰 탓도 있겠지만, 우선 그 음색의 부자연스러움이다. 전통적 악기들이 내는 '자연스런' 소리에 익숙해 있는 음악가들에게 컴퓨터 음악이 들려주는 소리는 거의 참을 수 없는 기계음에 지나지 않는다. 최근 들어 신디사이저의 기능이 향상되면서, 그리고 다양한 기법들이 발전되면서 상당히 자연음에 가까운 소리를 만들어 낸다고는 하지만, 전통 악기

음을 판단 근거로 하는 귀에게는 결코 만족스러운 것이 못된다. 그리하여 한편에서는 컴퓨터를 새로운 악기 개념으로 이해해야 한다는 주장도 있다. 즉, 실험적 음악을 만듦에 있어서 작곡가들이 전통적 소리 대신에 (아마도 20세기를 상징할 수도 있는) 새로운 음색을 창출하는 악기 그 자체로 이해되고 수용되어야 한다는 주장이다. 그러나 컴퓨터 음악이 그토록 음악가들을 매료하는 가장 중요한 이유 중의 하나는 "상상가능한 모든 소리"를 합성할 수 있는 능력이다. 그럼에도 사실은 그렇지 못하다는 데에 문제점이 있다. 즉, "상상가능한 모든 악기의 소리"를 합성해 내는 능력이 전혀 기대에 미치지 못하고 있다는 점이 컴퓨터에 대한 음악가들의 등돌림의 근본적 연원 중 하나이다.

이러한 약점에 대한 보완책이 오늘날 다양한 각도에서 추구되고 있지만, 그런 약점의 근본 원인 중 하나는 현존의 대부분의 컴퓨터 자체가 가지는 아키텍처 상의 문제이다. (이 문제는 비교적 전문적인 것이기는 하지만 본론에서 간략하게라도 살피기로 하자.) 그렇다면 대안은 컴퓨터 자체의 아키텍처를 바꾸는 일이다. 오늘날 이런 새로운 컴퓨터 개념으로 각광을 받고 있는 것이 소위 말하는 신경망 컴퓨터이다. 물론 신경망이 컴퓨터 음악에서의 그러한 요구 때문에 비롯된 것은 아니다. 이미 다른 여러 분야에서도 기존의 컴퓨터 개념의 한계로 인해, 보다 '인간의 사고에 가까운' 사고능력을 가지는 컴퓨터에 대한 요구가 있어왔고, 그로 말미암아 탄생을 본 것이라 할 수 있다. 외국에서는 이런 신경망 컴퓨터, 즉 연결주의를 음악에 응용하는 다양한 연구들이 행해지고 있지만, 필자의 좁은 정보 범위 내에서 볼 때 국내에는 아직 소개가 되지 않은 듯하다. 이 글에서는 연결주의의 기본 개념과 음악적 적용의 가능성들을 탐색해 보려 한다.

사실, 컴퓨터는 문자 그대로 계산기, 즉 어떤 수학적 공식을 빠른 속도로 수행해 내는 기계이다. 그러므로 컴퓨터를 음악에 적용한다는 것은, 거칠게 말한다면, 음악의 면모들을 수학적 공식으로 만들어 그 계산을 컴퓨터가 수행하도록 하는 것이다. 음악을 만드는 데 있어서 그러한 공식에 의한 체계(이를테면 수학에서의 순열과 같은)를 사용하고자 하는 시도가 본격적으로 이루어진 것은 비교적 최근의 일이지만 반드시 그런 것만은 아니다. 최소한 귀도(Guido d'Arezzo) 시대 이후 음악을 자동적으로 생성해 낸다는 개념 그리고 그러기 위해서 선택해야할 수많은 요소들을 구분짓기 위해 구조화 내지는 계층화 기법들을 사용한다는 생각, 이런 것들의 매력과 그에 따른 다양한 시도들이 꾸준히 지속되어왔다. 그러니까 대략 천년 전인 1026년에 벌써 귀도는 의전가사로부터 거의 자동화된 방식으로 직접 정선율을 만들어내는 방법을 서술하고 있다(Guido 1026. "Guidonis Aretini Micrologus." In J. Smits van Waesberghe, ed. Corpus Scriptorum de Musica IV. Rome: American Institute of Musicology. Printed in 1955. Loy 1991 p. 20 에서 재인용). 또한 이탈리아 오페라 작곡가들은 작곡을 보다 쉽게 하기 위해 음악적 난수표를 활용했던 것으로 전해지기도 한다(Loy 1991 p. 22).

18세기에 접어들어 확률이론이 발전하자 흥미롭게도 음악작품을 만드는 데 우연성 기법을 활용하는 예가 증가한다. 18세기 초 작곡가이자 이론가였던 키른베르거(Kirnberger)는 musikalische Würfelspiel이라 부르는 음악적 게임을 개발했다. 이것은 주사위 한 개(또는 두

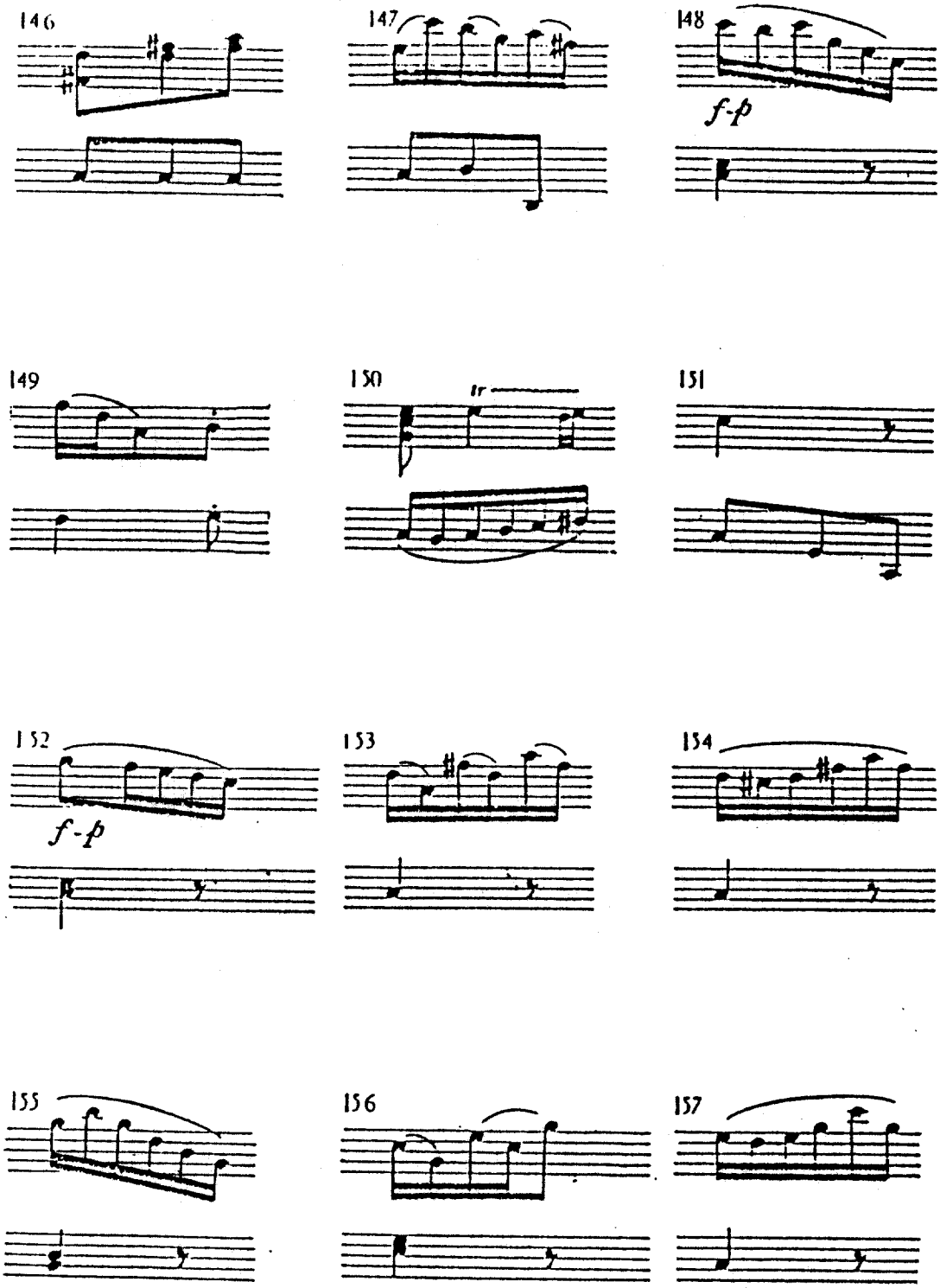
개)와 적절히 만들어진 단편적 악보들의 집합과 바둑판 모양의 판에 숫자가 들어 있는 표 하나를 가지면, 누구든지 진짜 미뉴엣과 폴로네이즈를 작곡할 수 있는 수단이었다. 이 게임은 대단한 유행을 보였으며, 여러 사람 중 최소한 모차르트(W. Mozart), 하이든, C.P.E. 바하의 손으로 만들어진 예들이 전해온다. 이 흥미있는 '작곡수단'을 좀 더 자세히 이해하기 위해 모차르트의 Köchel 294 D장조 "The Dice Composer"를 살펴보자.

이 작품은 대략 1770년 경에 만들어진 것으로 우연성을 응용하여 매번 새로운 왈츠와 트리오를 만들어낼 수 있도록 해주는 장치이다. 왈츠와 트리오 모두 각기 반복되는 8마디짜리 악절 둘씩으로 이루어지는 이른바 두도막 형식들로 되어있다. 물론 악보는 음자리표와 박자표만 있을 뿐 모든 마디는 다 비어있다. 각 마디에 음들을 채우기 위해 누구든지 2개의 주사위를 동시에 (또는 순차적으로) 던져 그 나오는 눈의 합을 구한다. 이때 나오는 눈의 합은 2에서부터 12 사이의 어느 한 숫자가 된다. 아래의 표와 같은 일종의 난수표(표 1)에서 맨 위의 행이 눈의 합을 나타내는 숫자들이다. 주사위를 던져 얻은 눈의 합에 해당하는 숫자 아래의 해당 마디에 있는 숫자를 고른다. 그리고는 이 숫자에 해당되는 번호에 그려져 있는 악보의 단편들(그림 1)을 비어있는 악보의 해당 마디에 '베껴 넣기만' 하면 된다. 이런 방식을 계속 반복하면 하나의 완벽한 왈츠와 트리오가 만들어진다. 한 곡 더 만들고 싶으면 다만 주사위를 다시 던지기만 하면 된다. 당연히 매번 새로운 왈츠와 트리오가 만들어질 것이다.

〈표 1〉 모차르트의 "The Dice Composer" 중 첫 악절을 위한 난수표

	2	3	4	5	6	7	8	9	10	11	12
마 디 1	96	32	69	40	148	104	152	119	98	3	54
마 디 2	22	6	95	17	74	157	60	84	142	87	130
마 디 3	141	128	158	118	168	27	171	114	42	165	10
마 디 4	41	63	13	85	45	167	53	50	156	61	103
마 디 5	105	146	153	161	80	154	99	140	75	135	28
마 디 6	122	46	55	2	97	68	133	86	129	47	37
마 디 7	11	134	110	159	36	118	21	169	62	147	106
마 디 8	30	81	24	100	100	107	91	127	94	123	5

〈그림 1〉 모차르트의 "The Dice Composer"의 단편 악보들 중 일부



이 기법은 작곡을 위한 어떠한 지식도 요구하지 않으면서 작품을 만들어낼 수 있도록 하는 순수하게 공식에 의한 방법의 효시에 해당한다. 귀도가 몇몇 가능한 요소들 중에서 선택하는 일을 아직도 작곡가에게 맡기고 있음에 비해, 몇 걸음 더 나아간 것이라 할 수 있다. 컴퓨터를 음악에 적용하려는 마음이 보여주는 자세는 이러한 사고와 시도들의 연장선 상에서 이해될 수 있을 것이다.

컴퓨터와 음악

1950년대 중엽에 작곡가와 음악학자들이 컴퓨터를 사용할 수 있게 된 이후 음악적 과정의 공식화 문제가 음악에서 전면에 부상했다. 컴퓨팅은 음악 작곡의 공식화를 용이하게 했다.

컴퓨터 작곡에서의 초기 작업은 대부분 인공적인 작곡 절차 체계를 개발하는 것이었는데 그것들을 인간의 지각 과정에 대한 심리학적 또는 생리학적인 어떠한 모델과도 연관시키려는 시도를 하지 않았다. 이런 작업은 컴퓨터를 매개로 한 작곡이라고 부를 수 있을 것이다. 작곡상의 다양한 설계들이 만들어낼 결과들을 빠르게 탐색하여 알려주는 컴퓨터의 속도와 능력에 매료되어 음악가들에게서 그런 도구를 지향하는 욕구가 점점 커졌고 필요한 접근방법들이 여러 가지로 개발되었다. 예를 들어 작곡가 크세나키스(Iannis Xenakis)는 추계학 기법을 사용하여 음악작품들을 생성해내는 프로그램을 썼다(Xenakis 1971). 그는

> 전자적 컴퓨터의 도움으로 작곡가는 일종의 비행사가 된다. 버튼을 누르고, 좌표를 도입하며 이전에는 멀리 떨어진 꿈으로만 얼핏 볼 수 있었을 뿐인 음향의 성좌와 은하수를 가로질러 음의 공간을 항해하는 우주선의 제어장치들을 통제한다. (Loy 1991, p. 23에서 재인용)

고 열을 올렸다. 크세나키스는 소리를 분석하는 푸리에(Fourier) 스펙트럼과 양자 물리학에서의 양자 표현방법의 근본원리가 서로 동일함을 입증하는 가버(Gabor 1947)의 매우 흥미있는 이론적 작업을 취해와서는 그것을 추계학 기법 및 기호 논리학의 요소들과 혼합했다. 쉔베르크와 마찬가지로 크세나키스도 자신의 체계를 작곡이론인 것으로 생각했다. 그러나 그 정도의 복잡성이라면 그의 아이디어를 실제로 실현시킴에 있어서 컴퓨터의 도움이 필수적이라는 사실이 놀라운 일은 아니다.

이외에도 많은 인공적 작곡과정들이 제안되었고 수행되었다. 질(Gill 1963)은 역탐색(backtracking)이라는 인공지능기법을 사용하여 음악을 생성하는 컴퓨터 프로그램을 짠 바 있고, 머튜즈와 로스러(Mathews & Rosler 1968)는 수학적 함수의 구성원리에 근거한 작곡 모델을 서술했다. 그들은 음악의 시간 함수들을 생성 및 조합하기 위해 보통은 음악과 연관되지 않는 결정론적 알고리듬들에 촛점을 모았다. 그들은 선율에 규칙 및 불규칙의 양화(量化) 함수들을 적용함

으로써 흥미있는 효과를 얻어냈다. 이를테면 서울의 스카이라인을 내다보는 어느 건물의 창문에 오선을 그려 왼쪽에서 오른쪽으로 건물들의 높이를 음으로 환산하여 선율을 만들어낸다고 상상해 보라. (빌라 로보스는 1940년대에 실제로 그렇게 했다.) 이 경우, 스카이라인은 선율을 나타내는 함수이고, 오선은 건물의 높이를 나타내는 가장 가까운 근사음을 발견하는 데 사용되는 양화함수이다. 머튜즈와 로스러는 이런 방식으로 음을 표현하고 조작하는 방법에 관한 이론을 개발하여 그것들을 다루는 컴퓨터 프로그램을 짠 바 있다.

작곡과정을 표현하고 수행하기 위한 많은 컴퓨터 언어들, 프로그램들 및 시스템들이 만들어졌다(Loy 1989 참조). 일반적으로 그것들은 모두 음악을 텍스트 또는 그래픽으로 서술하기 위한 메카니즘들, 공통관습 기보법, 음악적 대상들을 아이콘 또는 기호로 표현하거나 음악적 자료들을 한 단계 한 단계씩 절차적으로거나 또는 이 경우에는 이렇게 해야만 한다는 식의 선언적인 조작 방법들을 담고 있다. 특별히 음악을 위해 발명된 것도 많지만 이외에도 실제로 모든 표준적인 프로그래밍 언어들이 사용되어 왔다. 음악을 위해 발명된 언어들은 일반적으로 박자 개념, 음고 및 다이나믹스 등과 같은 특정한 음악적 자료 구조들을 조직적으로 채용하려는 시도 또한 보인다. 위에서 예를 든 것처럼 작품을 작곡해내는 체계 이외에도, 벅스톤과 몇몇 사람들(Buxton et al. 1978, 1979)이 함께 개발했던 것과 같은 이른바 작곡적 편집기들도 있다. 그것은 작곡가가 손으로 악보를 만들고 수정하는 데 도움을 주려는 것들이다.

고트프리트 쾨니히(Gottfried M. Koenig 1970a, 1970b)는 프로젝트 I과 II를 개발하여 컴퓨터 매개의 작곡과 컴퓨터가 직접 수행하는 작곡이라는 개념을 결합시켰다. 그것들은 어떤 점에서는, 작곡과정을 인간과 기계에 나누려는 방법상의 실험들이었다. 프로젝트 I은 작곡가로부터 최소한의 입력을 받아 수열 및 추계학적 절차들을 사용하여 어떤 음악작품에 대한 문자적 텍스트로 된 출력물을 만들어내는데 길이, 오케스트레이션 그리고 악기의 어택 포인트 부여 등의 파라미터들은 작곡가의 몫으로 남겨둔다. 그 후 작곡가가 그것을 오케스트레이션하는 과정에서 그 구조를 완성해야 한다. 반면에 프로젝트 II는 그 프로그램이 수행하고자 하는 작동들을 위한 세부적 구체사항들을 요구한다. 이 프로그램은 오케스트라로 편곡된 악보로 직접 바뀔 수 있는 텍스트 출력물을 생산한다.

위에서 논의한 접근들은 컴퓨터를 작곡가의 조력자로 사용한다. 다른 예로, 인공 두뇌학과 정보이론으로부터 강한 영향을 받아 힐러(Lejaren Hiller)와 아이작슨(Isaacson)은 작곡 과정의 계산가능성에 관해 몇가지를 실험했다(Hiller & Isaacson 1959). 그들은 작곡이라는 것이 음악적 연쇄들의 모든 가능한 조합들의 공간으로부터 수용가능한 음악적 연쇄를 선택하기 위해 음악적 규칙들을 사용하는 문제로 보았다. 이런 전제를 기초로 그들은 추계학 및 마르코프 연쇄 기법을 사용하여 공통관습 음악을 작곡하는 과정에 대한 컴퓨터 모델을 개발했다.

힐러와 아이작슨의 과정을 간단히 소개하면 다음과 같다. 코랄 화성화 규칙과 같은 활용 가능한 음악적 규칙들의 집합을 선택하여 그 음악적 규칙들을 컴퓨터의 하위 프로그램들의 집합으로 표현하는 방법을 찾았다. 그리고는 새로운 음악작품을 작곡하기 위해 이 서브루틴들을 사용했다.

새로운 작품의 첫음으로 어떤 무작위 음을 가지고 시작한 후 그것을 뒤따를 다른 후보음을 무작위로 선택한다. 그리고 나서 (지금까지는 오직 첫음 하나뿐인) 현재의 작품과 그 후보음이 작곡 규칙들 중의 선택된 집합을 수행하는 모든 서브루틴들에 의해 평가된다. 그 서브루틴들이 규칙위반을 어느것도 발견하지 못하면 그 음은 현재의 작품에 부가되지만 규칙 위반이 있으면 새로운 후보음이 선택되고 평가과정이 반복되어 모든 규칙을 만족시키는 후보음이 찾아질 때까지 계속한다. 그러면 그 음이 현재의 작품에 부가된다. 이 과정을 지속적으로 반복하여 결국 적절한 길이의 작품을 얻어낸다.

이것은 말하자면 매우 간단한 1차의 생성과 검증과정이다. 하나의 후보음에 대한 수용가능성을 결정하기 위해 현존 작품에 들어있는 두 개의 음들을 사용한다면 그것은 2차질서 과정이다. 어떤 과정의 차수가 높아질수록 수용가능한 후보음을 발견하는 것이 더욱 어려워지며, 그 작품을 납득할 수 있는 길이만큼 확장하기 위해서는 일반적으로 훨신 더 많은 후보음들을 평가해야 한다.

핑커튼(Pinckerton)이 1956년에 컴퓨터를 사용하여 동요를 작곡한 것이 최초이기는 하지만, 일반적으로 힐러는 음악을 작곡하기 위해 컴퓨터를 사용한 첫번째 인물로 꼽힌다. 그는 다양한 양식들로 수많은 실험들을 행했으며, 그 결과들을 Illiac Suite (Hiller & Isaacson 1959)라는 현악 사중주 형태로 출판했다. Illiac Suite은 여러가지 양식 부분들로 이루어지는데, 코랄로 시작하여 자유로운 무조성으로 끝난다.

힐러와 아이작슨의 체계는 특정 작곡 규칙들을 따른다는 점을 제외하고는 작곡가들이 실제로 작곡할 때 사용하는 일반적인 방법을 보여주지는 않는다. 작곡가들은 '생성과 검증'이라는 방법론을 사용하지 않는 것이 보통이다. 그러나 힐러와 아이작슨이 채용한 이 기법에 특정 양식에 속하는 음악 작품들의 어느 집합에 대한 확률분석을 병행한다면 그 결과로는 어떠한 음악 양식이라도 모방할 수 있는 체계가 만들어질 것이다. 예를 들어, 베토벤의 초기 피아노 소나타들에서 특정 문맥에 있는 특정 음형에서 특정한 음이 나타날 확률을 분석하여 힐러와 아이작슨의 체계를 그렇게 얻어진 확률을 토대로 적용하면 베토벤의 초기 피아노 소나타와 유사한 작품을 생성해 낼 수 있다는 뜻이다.

좀 더 기술적으로 설명하자면, 이러한 유형의 방법에서는 그 음악작품들의 집합 내의 모든 음악적 문맥들에서의 모든 음의 변환 확률들을 서술하는 다양한 차수들(즉, 하나의 문맥을 이루는 음들의 숫자들)에서 확률분산함수들이 만들어지게 된다. 그 후 힐러와 아이작슨의 원래 방법에서의 규칙 서브루틴들을 이 확률분산함수들로 대체한다. 이 함수들은 모델에 해당하는 음악작품들의 특정 집합이 안내로 주어지면, 특정 음악적 문맥 내에서 어느 음이 수용가능한지 여부의 그럴싸함을 처방전적 규칙을 기초로 해서가 아니라 통계학적 기초 위에서 구체화하기 위해 사용된다.

이렇게 되면 이 방법은 모델들의 어떤 집합과 동일한 확률구조를 가지는 연쇄를 만들기 위해 흔히 사용되는 마르코프 연쇄과정의 한 유형이 된다. (표준적인 마르코프 연쇄 용법에서는, 계산된 확률분산이 원래의 무작위 후보를 검사하는데 사용되는 것이 아니라 그 자체가 어떤 연쇄의 다음 요소를 선택하기 위해 사용된다.) 저차의 마르코프 연쇄 분석은 그 집합의 국부구조를 밝혀

준다. 고차의 분석은 점점 더 보편적인 구조를 밝힌다. 확률분산 함수들의 집합을 기초로 하여 작곡을 하려면 새로운 음들의 선택을 위한 판단 근거로 특정 차수를 선택한다. 차수가 높을수록 만들어지는 음악은 목표 양식에 더욱 가까워질 것이다. 그러나 높은 차수들에서는 모델이 되는 문맥을 제공하는 재료의 매우 큰 집합이 주어지지 않으면 과정이 중단되어버린다. 왜냐하면 문맥들을 만족시키는 점점 더 희박해지는 후보음들을 찾아내기가 점진적으로 더 어려워지기 때문이다. 또한 보다 높은 차수에서는 원래 재료의 중요한 부분들을 다시 듣게 되기 시작한다는 심각한 난점이 있다.

더욱이, 루이스가 지적했듯이(Lewis 1991), 확률적 방법은 비경제적이다. 그 집합 내의 모든 바람직한 구조들 뿐 아니라 모든 바람직하지 않은 구조들의 개연성들까지도 하나 하나 구체화해야 할 필요가 있기 때문이다. 그에 반해 처방전적, 규칙에 근거한 체계는 바람직한 구조들만을 표현하면 된다. 이 문제에 접근해 가는 한 가지 방법을 코넨과 그 동료들이 서술한 바 있다(Kohonen et al., 1991). 그들의 방법을 따르면 분석 절차를 사용함에 있어서 음악 예들의 집합이 요구하는 바에 따라 분석의 차수를 변화시키는 것이 가능하다. 따라서 작품을 만들어 갈 때 따라가기에 효과적인 문맥만을 저장하면 된다. 그 외에도 복잡하고 어렵기는 하지만 다른 많은 가능성들 역시 존재한다.

그러나 이런 접근들에서 보여지는 복잡성 및 난관들은 앞서 지적했던 음색의 부자연스러움과 마찬가지로 사실은 기존 컴퓨터의 개념 자체에 상당 부분 기인한다. 대부분의 컴퓨터 사용자들은 컴퓨팅에 대한 표준적인 von Neumann 접근법에 너무도 익숙해서 그 내재적 아키텍처에 포함된 근본가정들에 대해 거의 의문을 제기하지 않는다. 그런 가정들 중 첫번째 것은 컴퓨터가 기본적으로 많은 주변 메모리를 가진 강력하고 세련된 중앙처리장치(CPU)로 구성된다는 생각이다. 이 모델—오늘날의 많은 컴퓨터에 공통된—은 최소한 두 가지 점에서 중요한 문제점을 드러낸다.

첫째, 그것은 CPU 세대들과 메모리가 점진적으로 속도가 향상됨에 따라 그것들 사이의 소통이 주요 병목이 됨을 뜻한다. 둘째, 컴퓨터를 사용하여 어떤 작업을 한다는 것이 실제로는 CPU로 하여금 메모리의 다양한 부분들과 함께 특정한 일련의 작동들을 수행하도록 교육시키는 것이 됨을 뜻한다.

특히 후자의 면모가 컴퓨터 과학자들의 많은 주의를 끌어왔다. 왜냐하면 그로 말미암아 컴퓨터 프로그래머들이 불필요한 지루함을 짐으로 짊어져야 하기 때문이다. 프로그래머는 컴퓨터에게 단순히 어떤 문제에 대해 무엇이 받아들일만한 해답을 구성하는가에 관한 정보를 제공하는 대신에 고통스럽게도 그 해답을 어떻게 얻어내는가를 가르쳐야 한다(Balaban and Murray 1986). 즉, 얻어내고자 하는 목표를 가르치는 것이 아니라 목표에 도달하는 방법을 가르친다는 점이다. 고급의 컴퓨터 언어들(이를테면 Lisp, Smalltalk, Prolog)이 꾸준한 발전을 보여 온 것은 이런 바람직하지 않은 복잡성을 프로그래머의 영역으로부터 컴퓨터 자체의 영역으로 돌리려는 지속적인 시도라고 볼 수 있다.

연결주의

신경망은 컴퓨팅에 대해 매우 다른 접근법을 제공한다. 〈처리장치/메모리〉라는 이분법 및 고급 언어들에서 흔히 사용하는 일종의 기호에 의한 교육 세트 개념, 이 모두를 소거해버린다. 대신에 이상적인 "뉴런"에 의한 본질적으로는 아날로그한 망을 제공한다. 여기에서는 활성(excitation)의 서로 다른 패턴 또는 정도들을 뉴런들 사이의 상호연결 기능으로 생각한다. 고도로 병렬적인 하드웨어를 가지고 매우 빠르게 문제를 처리하기에 이상적으로 적합하도록 처리기능과 메모리가 폰 노이만 방식의 집중식과는 대조적으로 망 전체를 통해 균일하게 분산되어 있다. 마찬가지로 중요한 것은 신경망이라는 것은 종종 학습 메카니즘을 스스로 가질 수 있기 때문에 단순히 그 망을 원하는 행동의 예들에 반복적으로 노출시킴으로써 원하는 컴퓨테이션이 자동적으로 프로그램될 수 있다는 점이다. 그 망은 단순히 자신의 상호연결 상태를, 다양한 활성들의 결과가 그 망이 다룰 수 있는 한에서 원하는 행동에 최대한 근접할 때까지 고쳐나갈 뿐이다. 그리하여 신경망은 프로그래머들이 명료한 규칙이나 교육을 통해서는 생성해낼 엄두도 내지 못하는 복잡한 (그리고 아마도 인간이 수행하는) 행 동들을 모방할 수 있는 잠재력을 가지고 있는 것이다.

컴퓨터 음악과 마찬가지로, 신경망이라는 주제는 현재 학문들 사이의 상호협조가 매우 필요한 분야이다. (컴퓨터 음악에서의 학문들의 상호 협조에 관해서는 허영한의 "컴퓨터 음합성, 미디, 인터엑티브 시스템: 컴퓨터음악의 현재" 〔음악과 민족 제 6호, 1993, 215쪽〕에서 인용하고 있는 Moore의 멋진 도표를 참조하라.) 수학자, 물리학자, 전기 엔지니어, 생물학자, 그리고 인지과학자 등 모두가 그 발전에 기여해 오고 있으며, 각각이 그것을 각기 다른 용어로 각기 다른 학술지에 기고해 왔다. 따라서, 신경망, 인공 신경체계, 연결주의 모델, 병렬 분산처리 모델, 대량 병렬체계 등은 모두가 본질적으로 동일한 것을 가리킨다. 신경망은 일반적으로 아날로그 용어로 서술되기는 하지만 (그리고 아직은 막대한 비용의 문제가 있으나 언젠가는 아날로그 디바이스로서 널리 사용되겠지만), 오늘날의 거의 모든 작업은 디지탈 컴퓨터 시뮬레이션을 통해 이루어진다. 그러나 시뮬레이트된 형식으로도 신경망은 컴퓨터 프로그래밍에 대한 전통적 접근에 비해 상당한 장점을 제공한다.

신경망이 제공하는 여러 가지 장점 중 결정적인 것은 그야말로 탁월한 컴퓨터인 두뇌의 메카니즘을 모방하려고 시도한다는 점이다. 두뇌는 감각에서 들어오는 불분명한 정보를 놀라운 속도로 처리하는 뛰어난 능력을 보인다. 시끄러운 방에서도 휘파람 소리를 식별하며 어두컴컴한 곳에서도 사람의 얼굴을 알아보고 미묘한 정치적 발언에 담긴 숨은 뜻을 간파하기도 한다. 그 모든 능력 중에서도 가장 인상적인 것은—어떤 분명한 교육이 없어도—이런 재주를 부릴 수 있도록 하는 내적 표상을 만들어내는 법을 우리의 두뇌가 스스로 학습한다는 점이다.

두뇌가 정보를 처리하기 위해 스스로를 어떻게 훈련시키는가에 대해서는 아직 많은 것이 알려지지 않은 상태라서, 많은 이론들이 난무한다. 이런 가설들을 검토하기 위해 연구자들은 인공신경들로 이루어진 망들을 만들어 봄으로써 두뇌의 학습절차를 흉내내보려고 시도했다. 우선 신경

들의 본질적 면모들과 그 연결들을 연역함으로써 이런 신경망들을 구성한다. 그리고 나서 그 면모들을 시뮬레이트하도록 컴퓨터를 프로그램하는 것이 전형적 방법이다.

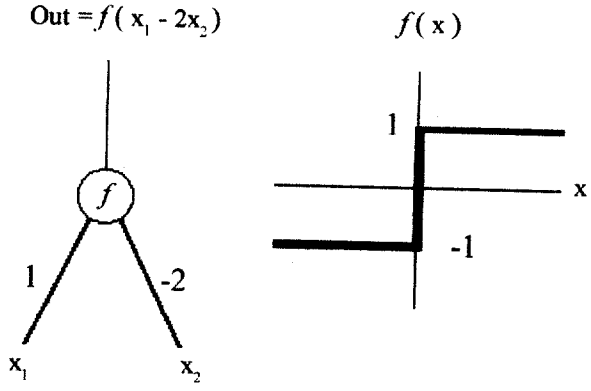
신경에 대한 지식이 빈약하고 컴퓨팅 능력 또한 한계가 있으므로 그렇게 만들어진 모델들은 필연적으로 실제의 신경망을 조악하게 모방한 것일 수 밖에 없다. 현재로서는 신경을 시뮬레이트함에 있어서 어떤 면모들이 가장 본질적인가라는 문제가 중요한 이슈가 되고 있는 실정이다. 그러나 본질적이라고 주장되는 면모들을 인공 신경망에서 검토해봄으로써, 두뇌가 정보를 어떻게 처리하는가에 관한 온갖 종류의 잡다한 이론들을 성공적으로 배제해버림으로써 혼란으로부터 해방될 수 있다. 또한 그 모델들은 두뇌가 그 탁월한 학습능력을 어떻게 이룩하는가를 드러내기 시작하고 있다.

인간의 두뇌가 가진 신경에 관해 지금까지 알려진 중요한 내용은 대략 다음과 같다. 인간의 두뇌에서 전형적인 신경은 수상돌기라고 하는 많은 멋진 구조를 통해 다른 신경들로부터 신호들을 모은다. 뿐만 아니라 축색이라고 알려진 길고 가느다란 가닥을 통해 전기적 활동의 뾰쪽파를 내보낸다. 축색은 수천 개의 가지로 갈라진다. 각각의 가지 끝에는 시냅스라고 부르는 구조가 있어서 축색으로부터의 뾰쪽파 활동을 전기적 효과로 바꾸어, 연결된 신경들에서의 활동을 억제 또는 활성화한다. 어떤 신경이 그 억제 입력에 비해 충분히 큰 활성 입력을 받으면, 다시 전기적 활동의 뾰쪽파를 축색을 따라 내보낸다. 이와 같이 신경들은 상호 연결되어 있으면서 상호 영향을 주고 받는다. 학습이 일어나는 것은 하나의 신경이 다른 신경에 미치는 영향을 변화시키도록 시냅스의 효율을 변화시킴으로써 이루어지는 것이다.

인공 신경망은 이와 같은 인간의 신경에 대한 지식을 토대로 하여 만들어진 것인데 기본적으로 매우 단순한 계산 요소들을 상호 연결시켜 놓은 것이다. 그 요소들이 신경 역할을 하며, 각각 그 유형에 있어서나 기능에 있어서 서로 동일하다. 시냅스의 기능은 변화시킬 수 있는 가중치로 모방한다. 각각의 요소(또는 유닛이라고도 부른다)들은 다른 유닛들로부터 입력을 받아 그 수입된 활동들의 패턴을 또 다른 유닛에게로 내보내는 단일한 수출 활동으로 바꾼다. 이처럼 바꾸는 일은 두 단계에 걸쳐 수행된다.

먼저 각각의 수입활동을 그 연결 가중치로 곱하고 이렇게 가중치가 부여된 입력들을 모두 합하여 전체입력이라 부르는 어떤 양(量)을 얻어낸다. 두번째로, 전체입력을 수출활동으로 변형하는 입력-출력 함수를 사용한다(그림 2). 이 구조가 생물학적 뉴런과 닮았으므로 어떤 저자들은 "유닛"이라는 말 대신 "뉴런"이라는 용어를 선호한다. 또 다른 사람들은 "매듭"이라는 용어를 사용하기도 한다. 이 세개의 용어는 본질적으로 같은 것이다. 대부분의 인공망은 수상돌기와 축색의 배열상태를 있는 그대로 세부적으로까지 반영하지는 않는다. 그리고 그 망들은 신경의 전기적 출력을 하나의 숫자로 나타내며, 이 숫자는 격발비율, 즉 그 활동비율을 나타낸다.

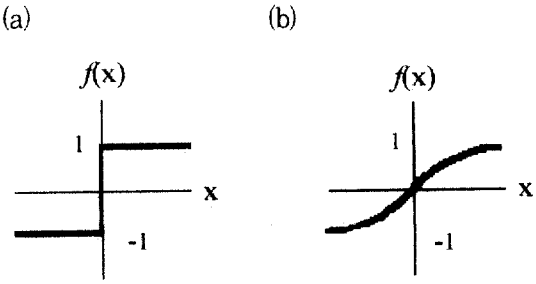
〈그림 2〉 신경망에서의 유닛의 작동 원리



〈그림 2〉는 인공 신경망에서의 유닛의 작동원리를 보여준다. 왼쪽의 그림에서 작은 원으로 표시한 것이 유닛인데, 그 자체로는 일종의 함수이다. 즉, 입력을 받아 그것을 한 가지의 출력값으로 변환시켜 내보내는 것이다. 그림에서 x_1 과 x_2 는 다른 유닛들로부터의 출력이다. 이것들에 각기 1 과 -2 가 곱해지는데, 이것은 다른 유닛들로부터 이 유닛에게로의 연결 가중치이다. 이렇게 가중치가 곱해진 값들의 합이 이 유닛의 입력 값이다. 이 유닛은 이렇게 얻어진 입력 값에 주어진 함수를 적용하여 출력 값을 생성해 낸다. 오른쪽의 그림은 이 유닛의 함수적 특성을 그래프로 나타낸 것이다. 즉, 입력 값이 0 보다 작으면 이 유닛은 출력으로 -1 을 내보내고, 입력이 0 보다 크면 1 을 출력으로 내보낸다.

신경망이 흥미있는 계산을 할 수 있는 이유는 이와 같이 각 유닛으로의 각각의 입력이 독특한 가중치 및 그 유닛들에 대해 구체화되는 입력-출력 함수에 연관되어 있기 때문이다. 가중치는 한 유닛으로부터 다른 유닛으로의 연결힘으로 생각될 수 있으며, 첫 유닛이 두번째 유닛에 끼치는 영향의 정도를 가리킨다. 가중치는 양수 또는 음수일 수 있으며, 따라서 영향도 활성적 또는 억제적이라 할 수 있다. 또 그 함수는 전형적으로 다음 세 가지 범주 중 하나에 해당한다. 선형, 문턱 그리고 시그마가 그것이다. 선형 유닛들에서는 출력 활동이 가중치가 부여된 전체입력에 정비례한다. 이른바 일차함수에 해당한다. 문턱 유닛들에서는 전체입력이 어떤 문턱값에 비해 더 큰지 혹은 작은지에 따라 출력이 두 레벨 중 하나로 된다(그림 3a). 시그마 유닛들에서는 입력이 변함에 따라 출력이 지속적으로 변하지만 선적이지는 않다(그림 3b). 시그마 유닛이 선형 또는 문턱 유닛에 비해 실제 신경과의 유사성이 더 크지만, 세 가지 모두 거친 근사꼴로 보아야 할 것이다.

〈그림 3〉 엄격한 제한자 함수(a)와 로그 함수(b)



특정 임무를 수행하는 신경망을 만들기 위해서는 유닛들이 서로 어떻게 연결될 것인가를 선택해야 하고, 그 연결들에 적절한 가중치를 부여해야 한다. 연결들은 한 유닛이 다른 유닛에 영향을 미치는 것이 가능한지 여부를 결정짓는다. 가중치는 영향의 강도를 구체화한다. 가중치를 곱한 입력 모두를 합하여 출력이 그 결과 합의 어떤 비선형 함수가 되도록 하는 것이 일반적이다. 그런 규칙을 수학적으로 표현하면, 그 망의 N 개의 유닛들 각각을 1과 N 사이의 단일한 숫자로 식별되도록 했을 때, i 번째 유닛의 출력 x_i는 다음과 같이 주어진다.

$$x_i = f(\sum_{j=1}^{N} w_{ij}x_j),$$

여기에서 w_{ij}는 j 번째 유닛으로부터 i 번째 유닛으로의 가중치이고, 함수 f는 어떤 비선형 함수이다.

위의 〈그림 2〉의 유닛 하나 짜리 망을 다시 생각해 보자. 거기에서는 입력 x_1이 입력 x_2의 두 배 이상일 경우에만 출력이 양(즉, 1)이다. 이 망이 진정한 컴퓨테이션 (다시 말해 결정을 내리는) 능력을 가지게 되는 데는 함수 f의 비선형성이 결정적이다. 그 비선형성으로 인해 입력에서의 양적 변화가 출력에서의 질적 변화를 산출할 수 있게 된다(즉, 출력이 단순히 입력에 정비례하여 변화하는 대신에 off에서 on으로 왔다갔다 할 수 있다). 실제로 f는 얼마든지 선택할 수 있다. 그러나 앞서도 언급했듯이 가장 일반적인 것 중 두 가지는 〈그림 3a〉의 엄격한 제한자(문턱)와 〈그림 3b〉의 부드럽게 한정하는 로그 함수(시그마)이다.

〈그림 2〉에서와 같은 간단한 한 유닛짜리 망은 실제로는 몇십년 전에 연구되었다 (McCulloch and Pitts 1943; Hebb 1949; Rosenblatt 1959). 그러나 그것들을 성공적으로 적용할 수 있는 문제들의 종류에는 근본적 한계가 있었다. 예를 들어 가중치를 어떤 방식으로 결합하더라도 한 유닛 망(역사적으로 단일층 perceptron이라 알려져 있다)이 XOR 문제(두 입력 x_1과 x_2가

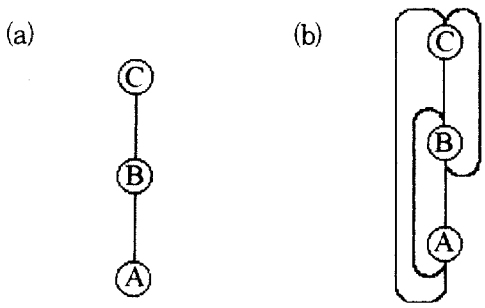
모두 -1.0 또는 1.0일 때 출력 -1.0을 산출하고, 입력 하나가 1.0이고 나머지 하나가 -1.0 일 때 출력 1.0을 산출하는 문제)를 해결할 수는 없다. 이러한 한계 및 여타의 잠재적 난점들이 분명해지자(Minsky and Papert 1969) 신경망에 대한 관심이 현저하게 감소했고 그 후 10년 동안에는 대신에 인공지능 쪽에 탐구활동이 보다 집중되었다.

그러나 1980년대 초에 종전의 한계에 구애받지 않는 보다 세련된 신경망들(가령 다층 perceptron)이 고안되었고, 인공 신경 체계에 대한 관심이 다시 활기를 띠게 되었다(Hopfield 1982; 1984; Hopfield and Tank 1986; Rumelhart et al. 1986; Sejnowski and Rosenberg 1986).

인공 신경망의 가장 보편적인 유형은 유닛의 세 그룹 또는 층으로 이루어진다. 입력 유닛의 층, "숨겨진" 유닛의 층, 그리고 출력 유닛의 층이 그것이다. 입력 유닛의 층은 숨겨진 유닛의 층으로 연결되어 있으며, 이것은 다시 출력 유닛의 층으로 연결된다. 입력 유닛의 활동은 그 망에 제공되는 가공되지 않은 정보를 나타낸다. 입력 유닛은 실제로는 어떠한 컴퓨테이션도 수행하지 않고 특정 입력 값을 받아들여 그대로 내보낼 뿐이므로 학자에 따라서는 이런 망이 사실은 두 개의 층으로 이루어진다고 주장하기도 한다. 숨겨진 유닛 각각의 활동은 입력 유닛들의 출력 값 및 입력과 숨겨진 유닛 사이의 연결 가중치에 의해 결정된다. 마찬가지로 출력 유닛들의 행위는 숨겨진 유닛들의 활동 및 숨겨진 유닛과 출력 유닛 사이의 가중치에 의존한다.

오늘날에는 매우 다양한 신경망들이 있지만 몇몇 주요 범주들로 나누어 볼 수 있다. 가장 근본적인 이슈 중 하나는 각 유닛의 출력이 모든 다른 유닛의 입력이 됨으로써 그 망의 유닛들이 완전하게 연결되느냐 아니면 그 상호연결 패턴이 보다 제한적이냐 하는 문제이다. 특히 만일 그 상호연결들이 〈그림 4a〉에서처럼 배타적으로 피드 포워드로 제한된다면, 주어진 입력 값으로부터 마지막 출력 값의 계산은 매우 곧바르다. 그와는 대조적으로, 만일 〈그림 4b〉에서처럼 그 망에 피드백 루핑이 있다면 유닛 B의 출력은 유닛 A의 출력에 의존적이고, 다시 유닛 A의 출력은 유닛 B의 출력에 의존적이 되는 식으로 계속된다. 이 출력들이 불안정하게 떨릴 수도 있지만, 망을 적절하게 설계하면 출력이 항상 안정된 값에 수렴할 것임이 입증되었다(Hopfield 1982). 따라서 실제로는 두 종류의 연결 도식들이 모두 실행가능한 것들이다.

〈그림 4〉 순수한 피드 포워드 망(a), 완벽하게 상호연결된 망(b)



또 다른 기본적 이슈는 가중치 자체에 관한 것이다. 어떤 경우들에서는 설계자가 원하는대로 그 망이 즉각적으로 기능하도록 가중치들을 고안할 수 있다. 그러나 당면 문제를 해결할 가중치가 어떤 것일지 전혀 알 수 없는 경우들이 대부분이다. 이런 상황에서는, 일련의 훈련 예들을 근거로 하여 가중치를 반복적으로 조정하기 위한 자동화된 과정인 감독된 학습 알고리듬을 사용할 수 있다. 그 망이 생산해내는 출력들이 요구되는 출력들에 필요한 만큼 근접하면 훈련 국면이 완료된다. 즉, 이제 가중치가 고정되고 그 망은 사용될 준비가 된 셈이다. 명료한 규칙들을 공식화해내기가 어려운 문제들에 대해 신경망이 그토록 매력적인 것은 바로 이런 학습 알고리듬이 존재한다는 점이다.

역추진 알고리듬

다음과 같은 과정을 밟음으로써 어떤 3층망이 특정한 임무를 수행하도록 가르칠 수 있다. 먼저 그 망에 훈련예들을 제시한다. 훈련예라는 것은 입력 유닛들의 활동 패턴(어떤 값들)과 출력 유닛 활동의 요구되는 패턴(값)으로 구성된다. 그러면 그 망의 실제 출력이 요구되는 출력에 얼마나 근접하는지 알 수 있다. 다음에 각 연결의 가중치를 변화시켜 그 망이 요구되는 출력에 더욱 가까운 근사값을 산출하도록 한다. 이 과정을 밟으려면 각 가중치를 그 가중치가 변함에 따라 오차가 변하는 비율을 계산하여 그에 비례하는 양만큼씩 변화시켜 가야 한다. 이 양—그 가중치의 '오차 도함수'로서 EW로 표시한다—은 효과적으로 계산하기가 쉽지 않다. EW를 계산하는 한 가지 방법은 가중치 하나를 약간 교란시켜 오차가 얼마나 변하는지를 보는 것이다. 이렇게 하자면 수많은 가중치들 각각에 대해 따로따로 그런 교란과 계산을 수행해야 하기 때문에 비효율적이다.

1974년경에 Paul J. Werbos는 하버드 대학 박사과정 중에 EW를 계산하는 훨씬 효과적인 방법을 발명했다. 그 방법은 오늘날 '역추진 알고리듬'이라고 알려진 것으로서 신경망을 훈련하는 보다 중요한 도구들 중 하나가 되었다. 역추진 알고리듬은 망의 모든 유닛들이 선형일 경우 이해하기가 가장 쉽다.

이 알고리듬은 처음에는 EA, 즉 어떤 유닛의 활동 수준이 변화함에 따라 오차가 변하는 비율을 계산함으로써 각 EW를 계산한다. 출력 유닛에서는 EA가 단순히 실제와 요구되는 출력 사이의 차이일 뿐이다. 출력층 직전의 층에 있는 어떤 숨겨진 유닛의 EA를 계산하려면 먼저 그 숨겨진 유닛과 그것이 연결된 출력 유닛들 사이의 모든 가중치를 확인해야 한다. 그후 그 출력 유닛들의 EA들을 그 가중치들과 곱하여 얻은 곱들을 합한다. 이 합이 그 선택된 숨겨진 유닛의 EA이다.

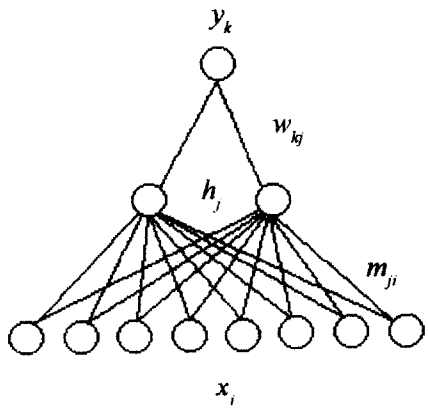
출력층 직전의 숨겨진 층의 모든 EA들을 계산한 후, 다른 층들의 EA는 활동들이 그 망을 통해 진행하는 것과 반대방향으로, 말하자면 출력층에서 숨겨진 층으로, 그리고 숨겨진 층에서 입력 층에게로 한 층 한 층 옮겨가며 마찬가지 방법으로 계산할 수 있다. 이로 말미암아 역추진이

라는 이름을 얻게 되었다. 어떤 유닛의 EA가 일단 계산되면, 그 유닛의 수입연결 각각을 위한 EW를 계산하는 것은 식은 죽 먹기이다. EW는 EA 곱하기 수입연결을 통한 활동이다. 그러나 비선형 유닛들에 대해서는 역추진 연산에 또 다른 단계가 포함되어야 한다. 역추진 전에 EA가 반드시 EI로 변환되어야 한다. 이것은 어떤 유닛이 받아들이는 전체입력이 변함에 따라 오차가 변화하는 비율이다.

역추진 알고리듬은 처음 발명된 후, 수년 동안 별 관심을 끌지 못했다. 아마도 그 유용성이 충분히 평가받지 못했던 때문인 것 같다. 1980년대 초에, 당시 San Diego의 University of California에 있던 David E. Rumelhart와 Stanford University에 있던 David B. Parker가 각기 따로따로 그 알고리듬을 재발견했다. 1986년에 Rumelhart와 또한 University of California에 있던 Ronald J. Williams, 그리고 University of Toronto의 Geoffrey E. Hinton 이렇게 세 사람은 이 알고리듬을 통해 숨겨진 유닛들에게 복잡한 입력 패턴들을 표상하는 법을 가르칠 수 있음을 보임으로써 그것을 유행시켰다.

역추진이 어떻게 작용하는가를 이해하기 위해 〈그림 5〉의 망을 살펴보자. 그리고 가중치가 무작위 값을 갖는 것으로 상정하자. 입력 활성의 어느 패턴이든간에 (즉 x에 어떤 값이 들어가더라도) 출력 활성(y_k)을 쉽게 계산해 낼 수 있는 패턴이 만들어진다. 그렇기는 하지만 아마도 x_i의 주어진 집합에 대해 y_k가 무엇이어야 하는가에 대한 몇 가지 예를 이미 가지고 있을 것이다. 따라서 실제 출력과 요구되는 출력 사이의 차이를 계산함으로써 각 출력의 편차를 쉽게 계산해낼 수 있다.

〈그림 5〉 피드 포워드 3층망



각각의 편차의 제곱을 합함으로써 전체 편차(표준 편차)를 계산할 수 있다. 제곱을 사용하는

것은 양수와 음수의 편차들이 상쇄하지 못하도록 하기 위함이며 또한 수학적 도함수를 보다 단순화하기 위함이다. 수학적으로 전체 편차(ε으로 표시)는 다음과 같이 정의된다.

$$\varepsilon = \sum_{k} (y_k - \tau_k)^2,$$

여기에서 τ_k 는 목표값(즉, 그 망이 스스로 산출하는 법을 배우기를 우리가 원하는 올바른 대답)을 나타내고 총합(Σ)은 출력 유닛에 해당하는 k의 모든 값에 대해 수행된다. 우리는 편차 ε이 가능한한 작아지기를 바란다. 그렇게 할 수 있는 유일한 길은 그 망이 주어진 입력들로부터 직접 목표출력을 산출하도록 하는 가중치들의 집합을 찾아내는 것이다. 그러나 이와 같은 가중치들의 최적의 집합을 찾아내기 위해 (전체를 다 뒤져보는 것 말고는) 어떤 절차를 사용하는 것이 가능하겠는가? 이런 최적의 가중치 집합에 도달하는 것은 처음 시작하는 무작위적 가중치들을 가지고 반복적으로 매우 조금씩 조정해 주기만 함으로써 가능하다는 것이 밝혀졌다.

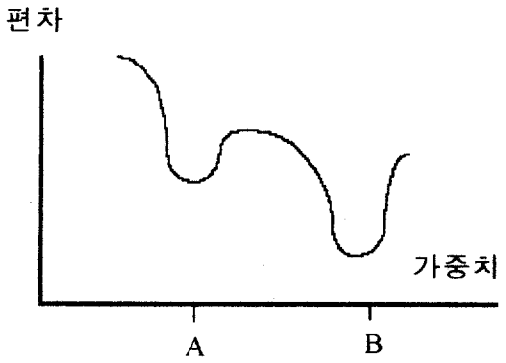
그 망에 하나의 입력 패턴을 제시하고 편차 ε을 측정할 때마다 가중치 하나 하나를 검토하고 각 가중치를 ε이 축소될 방향으로 아주 조금씩 증가 또는 감소시킨다. 이렇게 가중치를 변화시킴으로써 지속적으로 ε이 감소되는 한, 궁극적으로는 거의 편차가 없는 결과를 가질 가중치 집합에 도달할 것이다. 그러나 각 가중치를 정확히 얼마만큼 증가 또는 감소시켜야 하는지를 어떻게 알 수 있는가? 이 문제의 정확한 대답을 위해서는 어느 정도의 계산이 필요하다. 그렇긴 하지만 기본 아이디어는 매우 직관적이다. 출력 유닛에서, 어떤 특정 출력 활성이 그 유닛의 목표값에 못 미치면, 기본적으로 그 유닛은 현재의 입력 패턴에 의해 양수로 활성화되는 숨겨진 유닛들과 더 좋은 연결을 (즉, 더 큰 가중치를) 가질 필요가 있다. 마찬가지로, 그 유닛은 현재 음수로 활성화되는 숨겨진 유닛과는 더 약한 연결을 (즉 더 적은 가중치를) 가져야 할 것이다.

입력에서 숨겨진 유닛으로의 가중치는 평가하기가 좀 더 어렵다. 왜냐하면 숨겨진 유닛에 대한 정확한 목표값을 결코 사용할 수 없기 때문이다(출력 유닛만 그 목표값을 알 수 있다). 그럼에도, 마찬가지의 일반적 접근을 사용할 수 있다. 각 숨겨진 유닛이 전체의 출력 편차에서 어느 정도 비율을 담당하는가를 묻기만 하면 된다. 그리고는 이 담당정도가 축소되도록 입력 유닛으로부터의 연결들을 강화 또는 약화시킨다. 이 절차를 통해 ε이 바람직한 만큼 낮은 값에 도달되지 않을 수도 있다. 왜냐하면 주어진 입력들에 대한 목표 출력을 산출할 수 있는 가중치들의 어떠한 집합도 없는 경우를 생각할 수 있기 때문이다. 이런 경우 아마 그 망은 숨겨진 유닛을 더 많이 필요로 하거나 또는 그 문제를 풀기 위해서는 상호 연결의 어떤 다른 설계가 요구될 수도 있다.

그러나 그 망이 그 문제를 해결할 수 있을 때에도 더욱 큰 어려움이 발생할 수 있다. 그 망이 가중치들의 최상의 집합을 찾아내지 못하는 경우이다. 이러한 일은 〈그림 6〉에 설명된 상황이 발생하면 언제든지 일어난다. 가중치의 함수로서의 편차 그래프를 보면 편차가 낮은 결과를 얻을

수 있는 가중치들의 서로 다른 두 집합이 있다. 그러나 하나의 집합(점 B)만이 가능한 가장 낮은 편차를 제공한다. 일단 그 학습 알고리듬이 점 A로 표시된 가중치 집합을 발견해버리면, 점 B에 도달하는 것은 불가능하다. 그 학습 알고리듬은 가중치를 한 번에 극히 조금씩 조정할 뿐이고 그것도 편차가 점점 감소하는 방향으로이다. 점 A에서 점 B로 옮아간다는 것은 그 사이에 있는 가중치들의 어떤 집합에서는 편차를 증대시키는 셈이 될 터이고 이것은 결코 행해지지 않는다. 이것은 국부적 최소점(local minima: LM)이라 알려진 것으로서, 잠재적 심각성을 지닌 문제이기는 하지만 치명적인 경우는 드물다. 그 까닭 중 하나는 변화시킬 가중치들이 많을 경우 그런 LM으로부터 벗어나는 일이 보다 쉽다는 점이다. 또 다른 가능성은 비록 우리가 어떤 LM 에 빠짐으로써 끝났다고 하더라도 그 LM은 어쨌건 우리가 만족할 수 있을 충분히 낮은 ε을 가진다. 또는 무작위 가중치들의 다른 집합을 가지고 그 훈련과정을 다시 시작하면서 이번에는 그 LM을 피하기를 기대한다. (역추진에 의한 학습을 설명하는 다른 흥미있는 예로는 석봉래 옮김, P.M. Churchland, 물질과 의식, pp. 243-51을 보라. 거기에서는 잠수함의 음파탐지기에 의한 반사파를 입력으로 사용하여 적 항구의 기뢰와 바위를 구분하는 신경망을 훈련시키는 방법을 상당 정도로 자세히 서술하고 있다.)

〈그림 6〉 편차 그래프



역추진 알고리듬은 여러 개의 층들을 지닌 망이 다양한 임무들을 수행하도록 훈련시킴에 있어서 놀랍도록 탁월한 능력을 발휘한다는 사실이 입증되었다. 입력과 출력의 관계가 비선형이고 훈련자료가 풍부할 때 특히 유용하다. 그 알고리듬을 적용하여 탐구자들은 손으로 쓴 숫자를 인식하거나, 환율을 예상하고, 화학변화의 생산물을 극대화하는 신경망들을 개발해냈다. 심지어 망원경에서 공기에 의한 왜곡을 없애기 위해 거울을 조정하는 망을 훈련하는데 그 알고리듬을 사용하기도 했다.

신경과학 분야에서는 Massachusetts Institute of Technology에 있는 Richard Andersen

그리고 University of California의 David Zipser가 대뇌피질의 몇몇 신경들의 기능을 설명함에 있어서 그 알고리듬이 훌륭한 도구임을 밝혔다. 그들은 역추진을 사용하여 시각적 자극에 반응하는 신경망을 훈련시켰다. 그 후 그들은 숨겨진 유닛들의 반응이 망막으로부터의 시각 정보를 대뇌의 더 깊은 시각영역에 적합한 형태로 바꾸는 역할을 하는 실제신경들의 반응과 매우 유사하다는 점을 발견했다.

그러나 역추진은 생물학적 신경이 학습하는 방법에 관한 이론으로서 다소 혼합적 양식의 수용방법을 보이고 있다. 한편으로, 역추진 알고리듬은 추상적 수준에서는 탁월한 기여도를 보였다. 숨겨진 유닛에서 탁월한 표상들을 창출함에 있어 매우 능숙하다. 그 결과 탐구자들은 가중치들이 오차를 줄이도록 차츰 조정되는 과정을 학습하도록 함에 있어서 확신을 가지게 되었다. 이전에는 많은 사람들이 그런 방법으로는 가망이 없다고 생각했다. 왜냐하면 그렇게 하면 국부적으로 최적의 해결에는 도달하지만 전체적으로는 끔찍한 해결점에 이르게 되는 것이 필연적일 것으로 여겼기 때문이다. 예를 들어, 숫자 인식망은 1과 7을 식별하게 해줄 이상적인 가중치 집합이 존재하더라도 1과 7을 혼동하도록 하는 가중치 집합에게로 계속해서 이끌고 갈 수도 있다. 이러한 두려움으로 말미암아 어떤 학습절차건 궁극적으로 전체적 최적해결로 수렴한다는 것이 보장되는 경우에만 흥미를 줄 수 있다는 널리 퍼진 신념이 더욱 뒷받침을 얻었다.

다른 한편, 역추진은 생물학적으로는 그럴싸해 보이지 않는다. 가장 뚜렷한 난점은 정보가 동일한 연결들을 통해 역방향으로, 즉 한 층에서 그 전의 층으로 여행해야 한다는 점이다. 분명히 이것은 실제 신경에서는 일어나지 않는다. 그러나 이러한 반론은 실제로는 다소 피상적이다. 두뇌는 보다 나중의 층으로부터 이전의 층으로 가는 수많은 통로들을 지니고 있으며, 학습에 요구되는 정보를 전달하기 위해 이 통로들을 여러가지 방법으로 사용할 수도 있음이 알려졌다.

보다 중요한 문제는 역추진 알고리듬의 속도이다. 여기에서의 중심논제는 망이 커질수록 학습에 요구되는 시간이 얼마나 증대되는가이다. 주어진 훈련예의 가중치들의 오차 도함수를 계산하는데 소요되는 시간은 망의 크기에 비례한다. 계산 양이 가중치들의 수효에 비례하기 때문이다. 그러나 일반적으로 망이 클수록 더 많은 훈련예가 필요하고 가중치들을 더 자주 최근 것으로 바꾸어야 하기 때문에 학습시간은 망의 크기가 커지는 것보다 훨씬 더 빠른 속도로 커진다.

실제 학습에 대한 모델로서의 역추진에 대한 가장 심각한 반론은 각각의 훈련예에 대해 요구되는 출력을 공급해줄 교사가 필요하다는 점이다. 그와는 대조적으로 사람들은 대부분의 사물들을 교사의 도움없이 배운다. 외부세계에 대한 내적 표상을 우리는 감각 입력으로부터 추출해내는 법을 배워야 한다. 그러나 아무도 우리에게 그 방법을 상세히 서술해주지 않는다. 문장이나 시각적 장면들을 이해하는 법을 우리는 어떠한 직접적 교육도 없이 학습한다. 만일 어떤 망이 아무런 지식도 교사도 없이 출발한다면 어떻게 적절한 내적 표상법을 학습할 수 있을 것인가? 어떤 망에게 패턴들의 방대한 집합을 제시하고 그것들을 가지고 어떻게 해야 할지에 대해서는 아무런 정보도 주지 않는다면, 해결해야 할 문제가 무엇인지 조차 알지 못할 것임은 자명하다. 그럼에도 불구하고, 탐구자들은 그 망의 가중치들을 적절하게 조정할 수 있는 몇몇 범용의 이른바 "감독되지 않

는" 과정들을 개발했다.

간단한 음악 예 : 리듬을 평가하기

이번에는 인공 신경망의 이해를 위해 그리고 음악에 대한 연결주의의 적용 가능성을 이해하기 위해 신경망을 음악에 적용한 간단한 예를 인용해 보자(Dolson 1991, pp. 5-9). 가령, 간단한 리듬들에 대하여 우리의 판단(이를테면 좋은 리듬 또는 나쁜 리듬을 식별하는 판단)을 흉내낼 수 있는 소프트웨어를 원한다고 가정해보자. 이 소프트웨어를 어떤 자동화된 작곡 어플리케이션에서 일종의 필터로 사용하고 싶어할 수도 있다. 물론 우리가 좋아하거나 싫어하는 내용을 단순한 규칙들로 서술할 수 있다면 어느 리듬이건 좋거나 나쁜 것으로 분류하기 위해 일련의 만일-그러면(if-then) 문장으로 된 프로그램을 짤 수 있을 것이다. 그러나 우리는 왜 좋아하는지를 실제로 정확히 알지 못하면서도 무엇을 좋아하는지는 아는 경우가 더 많다. 우리는 다만 그 망에게 좋은 리듬과 나쁜 리듬의 예들을 보여주어(그것에게 우리의 판단에 따르자면 어느 것이 좋고 어느 것이 나쁜가를 말해 주어) 우리의 판단을 흉내내는 법을 배우도록 시킬 수 있다.

이 예를 구체적으로 하기 위해 8분음표를 가장 짧은 음으로 하는 4/4박자의 한 마디를 평가하는 것으로 한정하자. 이는 한 마디에 8개의 시간 퀀타(quanta)가 있어서, 각각이 어택 또는 쉼표를 가짐을 뜻한다. 이제 우리 예의 목적을 위해, 실제로는 우리의 선호를 지배하는 내재적 규칙이 있지만 우리는 청자로서 그 존재를 알지 못한다고 가정하자. 따라서 이 예에서는 다음과 같이 가정한다. 다만 첫 박에 어택이 있고 두번째와 네번째 박(즉 세번째와 일곱번째 시간 퀀타)가 모두 어택을 가지거나 또는 두번째와 네번째 박 모두에 쉼표가 오는 그런 리듬만을 좋아하지만 그렇다는 사실을 알지는 못하고 있는 것이다. 이 문제를 신경망으로 어떻게 공식화할 수 있을까?

우선, 입력(리듬)과 출력(판단)에 대한 적절한 코드화를 결정해야 한다. 여기에는 많은 가능성들이 있다. 그러나 가장 좋은 것들은 유닛의 사용과 연결에서 경제적이어야 하고 해결될 문제를 명료하게 반영해야 한다. 예를 들면, 그 리듬이 8개의 시간 퀀타로 구성되는 것으로 정의되며, 각각이 어택 또는 쉼표를 가지기 때문에, 입력 유닛 8개를 생각하여 각각이 출력으로 1.0 또는 -1.0을 가지도록 하는 것이 자연스럽다. 마찬가지로 우리의 판단은 단 하나의 출력 유닛으로 나타낼 수 있으며 좋은 리듬에 대해서는 1.0을 출력하고 나쁜 것에 대해서는 -1.0을 출력한다. 그러므로 입력층에 8개의 유닛을 가지며 출력층에 한 개의 유닛을 가질 것이다.

현재로서는 숨겨진 층에 얼마나 많은 유닛을 할당해야 하는지를 결정할 간단한 방법은 없다. 다만 다양한 숫자를 배당해보고, 그 망이 각각의 경우에 얼마나 잘 수행되는가를 볼 수 밖에 없다. 그러나 일반적으로 유닛을 적게 사용할수록 컴퓨테이션이 더욱 효과적이며 결과가 더욱 보편적으로 된다. 이 예를 위해서는 단 두 개의 숨겨진 유닛을 쓰기로 하자. 그 결과의 망이 앞서의 〈그림 5〉이다.

8개의 입력 유닛이 있어서 각각은 두 개의 숨겨진 유닛에 연결되고(전체 16개의 가중치), 이 2

개의 숨겨진 유닛 각각이 하나의 출력 유닛에 연결된다(2개의 가중치가 더해진다). 그림에는 나와 있지 않지만, 이에 덧붙여, 각각의 숨겨진 유닛과 출력 유닛에 항상 1.0을 출력하는 보조적인 편향 유닛으로부터 한개의 추가적 입력선이 있다. 이 편향 입력의 가중치를 조정하면 입력들이 양수가 되기 위해 넘어야 하는 문턱의 높이를 변화시키게 된다. 그 망은 모두 합해 21개의 가중치를 가진다(입력에서 숨겨진 층으로 16개 더하기 숨겨진 층에서 출력으로 2개 더하기 3개의 편향 가중치).

이제 그 망을 훈련시킬 필요가 있다. 이 값들 각각은 어택과 쉼표의 좋은 조합이 8개의 입력 유닛에 제시되었을 때만 전체의 망이 양수를 출력하도록 조정되어야 한다. 역추진 알고리듬을 컴퓨터가 적용하도록 함으로써 이것을 행할 수 있다.

일반화 문제

그 망은 8개의 이중적 입력들을 가졌다. 따라서 가능한 입력 집합들은 256 (2^8)개이다. 그러나 그 망을 훈련시킬 때 256가지 가능성 모두를 섭렵해야 한다면 불행한 일이다. 더 적은 수의 예들을 가지고 훈련과정을 끝내고 싶을 것이며 그렇게 함으로써 시간을 절약하게 된다. 신경망의 매력은 그것이 예들로부터 학습될 수 있다는 점 뿐 아니라 가능한 모든 예들의 전부가 아니라 한정된 부분집합으로부터도 학습될 수 있다는 점이기도 하다. 그것은 이전에 본 적이 없는 예들에 대해서도 올바른 해답을 찾아낼 수 있다.

훈련용 부분집합을 넘어서까지의 이러한 일반화가 어떻게 가능한가? 그리고 어느 정도 작은 부분집합이 적당한가? 이런 질문들은 신경망의 사용과 이해에 있어서 중심적인 것들이다. 앞에서는 어떤 망이 산출하는 출력값들이 목표값들의 집합에 가능한한 근접하게 되도록 그 망 내의 각 가중치들이 반복적으로 그리고 증식적으로 조정되는 그런 학습 알고리듬을 서술했다. 그리하여 그 망은 본질적으로는 훈련 예의 입력에서 출력으로의 매핑에서 어떤 패턴들을 "발견"한다. 그리고 이 패턴들이 그 망이 궁극적으로 획득하는 가중치의 특정 집합에 반영된다. 그 망의 입장에서 보자면, 이 패턴들은 순수하게 통계적 사고의 결과 얻어지는 것이다. 그러나 훈련과정이 끝난 후 그 망을 검토할 때 우리는 여지껏 모르고 있었던 그 내재적 문제를 표상하는 내재적 규칙들 또는 방법들의 집합을 발견하기 위해 그 패턴들을 사용할 수 있을 것이다.

그 망은 각각의 훈련 예에 개별적으로 반응하기 위한 충분한 유닛과 가중치를 가지지 않으면 언제든지 그런 패턴들을 발견하도록 강요된다. 예를 들어, 우리의 망을 훈련시키려 하면서 거기에 단 두 개의 훌륭한 리듬 예를 보여준다고 가정해 보자. 아마도 그 학습 알고리듬은 각 숨겨진 유닛이 그 리듬들 중 정확히 하나에 대해 양수로 반응하도록 가중치를 조정할 것이다. 그러나 이것은 우리가 원하는 바가 아니다. 왜냐하면 이런 식으로 문제를 해결하는 망은 왜 어떤 입력 패턴들은 좋은 것으로 판단되는 반면 다른 것들은 나쁜 것으로 판단되는가를 설명해 줄 내재적 관계들을 어느 것도 개발하지 못하기 때문이다. 따라서 그 망은 일반화 능력이 없다. 그 망이 훈련

기간 동안 보지 못했던 선호되는 입력 패턴이 이제 나쁜 것으로 판단될 수도 있다. 왜냐하면 그 망은 훈련기간 동안 본질적으로 쉬운 길을 택했기 때문이다. 즉, 단순히 정답을 외웠을 뿐이다.

훌륭한 일반화를 훈련기간 동안에 어떻게 확신할 수 있는가 하는 문제는 오늘날 신경망에서 중요한 탐구 논제이다. 그렇지만 단지 몇 개만의 숨겨진 유닛과 많은 훈련 예들을 가지는 것이 올바른 방향으로의 단계임은 분명하다. 따라서 이 경우에는 숨겨진 유닛 2개를 사용했고 그 망을 40가지 패턴에 대해 훈련시켰다. 〈표 2〉는 우리의 망을 12개의 훌륭한 입력 패턴과 28개의 나쁜 패턴에 대해 훈련시켰을 때 얻어진 가중치들을 보여준다. 처음에는 그 망에 무작위로 선택된 가중치들의 집합을 제시하고 그 후 역추진 학습 알고리듬을 반복적으로 사용하여 실제 그 망의 출력과 목표출력 사이의 편차의 제곱이 0.0002 미만이 될 때까지 수정했다. 여기에는 대략 20회의 조정이 필요했으며, 각각의 가중치 조정은 40개의 훈련 패턴 모두의 편차를 누적한 후에야 이루어졌다. 이 훈련은 보통과는 다르게 빨리 끝났는데 부분적으로는 첫 가중치 집합을 우연히도 잘 선택했던 때문이다.

〈표 2〉 가중치 표. 각 숫자는 해당 행의 유닛으로부터 해당 열의 유닛에게로의 가중치를 나타낸다. 이를테면 H1 행과 O1 열이 만나는 지점의 -5.61 이라는 숫자는 O1 유닛이 H1 유닛으로부터의 출력활성에 가중치 -5.61 을 곱한 만큼의 입력활성을 받게 됨을 뜻한다. BIAS는 편향 유닛을 뜻하며, I 는 입력 유닛, H는 숨겨진 유닛, O는 출력 유닛을 뜻한다.

	BIAS	***I1***	***I2***	***I3***	***I4***	***I5***	***I6***	***I7***	***I8***	***H1***	***H2***
H 1	-0.24	-5.35	0.01	5.65	1.55	1.30	-0.15	-5.57	-0.32	0.00	0.00
H 2	-0.54	-3.50	-0.77	-6.65	-0.23	0.20	0.63	0.00	-0.57	0.00	0.00
H 3	-5.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-5.61	-5.65

그 망은 40개의 훈련 예들로부터 어떻게 성공적으로 일반화할 수 있었는가? 256개의 패턴 전체를 가지고 시험해본 결과 모든 가능한 입력에 대해 정답을 산출했다(0.0 보다 큰 모든 출력이 좋음을 뜻하는 것으로 가정한다). 가중치 자체를 검토해보면 또다른 통찰력을 얻을 수 있다. 신경망에서 흔한 일이지만, 우리는 그 망이 우리 스스로 발명한 것과는 좀 다른 해결을 발견했음을 알게 된다.

출력 유닛은 편향 유닛과의 -5.13 연결에 의해 off 쪽으로 강하게 기울어진다. 숨겨진 유닛으로부터의 음수 가중치 때문에, 그것은 숨겨진 유닛 둘 모두가 그 자체 음수 출력을 가져

$$(-5.61 \times \text{unit1}) - (5.65 \times \text{unit2}) > 5.13$$

와 같이 될 경우에만 on이 될 수 있다. 숨겨진 유닛 둘 모두는 다만 약간 off 쪽으로 편향성을

가진다. 그러나 각각은 강박상의 어택(즉, 첫 입력 유닛으로부터의 1.0)에 의해 보다 강하게 off로 된다. 그러나 숨겨진 유닛 중 첫번째 것의 경우에서는, 제 2박과 제 4박에서의 어택-쉼표의 조합이 그 음수 입력을 극복하여 그 유닛을 강하게 on이 되도록 할 것이다. 역으로, 두번째 유닛은 같은 박들에서의 쉼표-어택의 조합에 의해 on이 될 것이다. 따라서 이 망은 우리의 예에서 첫 박, 두번째 박 그리고 네번째 박이 가지는 엄청난 중요성을 인식한 것으로 보인다. 그렇기는 하지만 무작위 가중치들의 다른 집합을 가지고 학습을 시작하면 동일한 해결에 이르지 않는다는 점을 주목해야 한다. 이것은 가능한 최소의 편차를 찾는데 있어서 항상 전체적 최소점(global minimum)에 도달하는 것만은 아니라는 점을 지적해 준다.

가능성

신경망은 세련된 자동화된 행위들을 획득하는 문제에 대해 여지껏의 전통적인 인공지능 및 기호에 의한 프로그래밍 접근에 대한 멋진 대안을 제시한다. 어떤 문제들에 대해서는 사실 신경망이 훨씬 훌륭한 접근방법이라고 강하게 주장할 수 있다(Abu-Mustafa와 Psaltis 1987). 또 아직 판정이 내려지지 않은 문제들도 있다(Daedalus 1988). 신경망의 전망이 밝은 것으로 보이는 많은 분야에서 그것들이 관련 문제를 심각하게 단순화시킨 꼴에 대해서만 검토되었다는 점에 주목하는 것이 중요하다. 보다 규모가 큰 신경망들이 마찬가지로 쉽게 훈련되는지 또는 새로운 기법들이 훈련을 용이하게 하기 위해 개발될 수 있는지 여부가 아직 살펴야 할 문제로 남아있다.

이렇듯 아직 초보적 단계임에도 불구하고, 신경망이 컴퓨터 음악에 의미심장한 충격을 줄 것임은 거의 확실한 것으로 보인다. 최소한 (이 글에서 인용한 리듬 판단 신경망의 예가 암시하듯이) 그것들은 컴퓨터 매개의 작곡에서 분명한 어플리케이션을 가진다. 더욱이, 망이 더욱 세련되어지고 시뮬레이션 능력 및 컴퓨팅 능력이 지속적으로 신장됨에 따라, 신경망은 음색의 영역에도 실질적인 영향력을 행사할 것으로 보인다.

"상상가능한 모든 소리"를 합성하는 능력은 컴퓨터 음악에서 가장 지속적이고 가장 매혹적인 매력들 중 하나가 되어왔지만 이러한 음향적 자유의 보다 제한된 (그리고, 많은 사람들에게 보다 바람직한) 면모는 앞서도 언급했듯이 우리로부터 계속해서 도망치고 있다. 즉, "상상가능한 모든 악기의 소리"를 합성하는 능력이 분명히 가능할 것임에도 불구하고 결코 음악가들을 만족시킬만한 수준에 도달하고 있지 못한 것이다. 이 두 자유의 차이, 즉 상상가능한 모든 소리를 합성할 수 있는 음향상의 자유를 가지고도 상상가능한 모든 악기의 소리를 합성할 수 있는 음향적 자유를 구가하지 못함은 비전자 악기로부터 나오는 소리들이 갖는 "자연스러움"과 음높이 또는 음의 세기가 바뀌더라도 동일한 음원으로부터의 소리인 것으로 확신시키는 이른바 "원리적 변주와 통제"에 있다. 전자적 소리에 이러한 음향적 특질을 부여함에 있어서 어느 정도 성공을 거두었다. 그러나 그 성공은 대부분 현존 악기의 소리들을 섬세하게 모방하는 경우들에 국한되어 있다.

Dolson(1991)은 다음과 같이 몇가지 중요한 도전을 제시한다. (1) 이 모방들을 더욱 효과적인

것으로 하기(컴퓨테이션적으로 및 메모리의 사용에 있어서도), (2) 이러한 모방들이 자연주의적 통제에 더 잘 반응하도록 하기(예를 들어 활 쓰기 또는 공기 압력에 직접 상응하는 지속적 변화), (3) 아직 발명되지 않은 음향악기에 대한 수긍이 가는 시뮬레이션의 고안(가령 음고와 세기가 변하는 소리들을 항상 단일한 음향 원천으로부터의 출력인 것으로 들리도록 하기).

오늘날, 소리 합성에 대한 접근 중 이러한 점에서 장래가 밝은 것은 3가지가 있다. 자연주의적 전자음향을 얻어내는 한 가지 명백한 방법은 실제 악기소리의 녹음된 조각들을 연결시키는 것이다. 이 접근은 소위 샘플링 기법으로서 유명한 많은 전자 신디사이저에서 사용되고 있는 것인바, 매우 성공적인 것으로 판명되었으나 자료량이 너무 많으며, 지속적으로 변하는 제어 신호에 대한 응답도가 제한적이고 낯선 악기들로의 확장이 어렵다는 한계를 지닌다. (샘플링 기법에 대한 자세한 설명은 허영한 앞의 글 169-178쪽 참조.) 이 접근의 반대쪽 극단에 해당하는 것은 음향 악기를 컴퓨터로 이뮬레이션한 물리적 모델을 근거로 한 전자적 소리의 합성이다(Wawrzynek 1989). 이 접근은 위에서 인용한 세가지 도전 모두를 만족시킬 것으로 매우 기대가 크다. 그러나 아직 초기 단계에 있으며, 수학적 물리학적으로 상당히 세련되어야 지속적 발전이 이루어질 것이다.

이 양극단의 중간에 위치한 흥미있는 것이 모델에 기초한 분석-합성법이다(Moorer 1985). 이 접근법의 성패는 소리합성을 위한 적절한 내재적 모델의 선택 그리고 원래의 녹음된 소리를 재합성하도록 할 그 모델의 시간에 따라 변화하는 파라미터들을 결정하기 위한 녹음된 소리들의 분석에 달려있다. 예를 들면, 덧셈 합성(허영한, 앞의 글 187-89쪽 참조)에서 그 모델은 사인파의 합이며, 분석은 각 사인파의 시간에 따라 변하는 진폭과 주파수를 결정하여 그 합한 결과가 원래의 소리가 되도록 한다. 역으로 뺄셈 합성에서는 그 모델이 주기적 충격 줄기에 의해 활성화되는 필터이고, 분석은 충격 줄기의 주기와 필터의 주파수 반응을 결정한다. 덧셈과 뺄셈 모델 모두가 자연 악기음에 매우 가까운 소리를 효과적으로 합성하는 데 사용될 수 있다.

이렇게 어렵고 복잡하기는 하지만 그나마 겨우 단일음의 합성에서 어느 정도 만족도를 기대할 수 있을 뿐, 임의의 악절을 합성하는 데는 둘 모두 그다지 적합하지 못하다. 그 이유는 그 모델들 자체가 실제 악기음들이 음고와 세기의 변화에 대한 함수로서 자연스럽게 변화하는 방식에 관한 정보를 얻을 수도 활용할 수도 없기 때문이다. 그 모델들은 완전히 정적이다. 그것들은 다만 분석 자료에 의해 구체화된 순간적 스펙트럼을 재생산할 뿐이기 때문이다. 거기에는 소리를 생산하는 물리적 체계의 내재적 역동성을 붙잡을 어떠한 메카니즘도 없다.

그와는 대조적으로, 음악적 소리 생산의 신경망 모델은 음향적 구조에서 이와 같은 보다 깊은 층에 접근할 수 있는 능력을 가지고 있다. 어떤 악기가 현재 모방되고 있으며 (끊임없이 변하는) 어떤 음높이와 세기가 현재 요구되는가를 구체화하는 입력들을 가지고 있으며, 합성된 파형을 일련의 샘플 값으로 내놓는 단 한개의 출력을 가지는 일종의 유토피아적 신경망을 상상할 수 있다. 그 망은 다양한 실제 음악 연주에 대해 훈련받을 것이며, 그후 본질적으로는 범용악기의 역할을 맡게 될 것이다. 그 망이 각 음향 악기의 특성을 결정하는 내재적 역동적 관계들을 파악하는 데

성공한다면, 낯선 제어 파라미터들에도 훌륭하게 반응할 것이다.

이런 새로운 영역에서 유용한 직관을 개발하기 위해서는 직접 손으로 해보는 시뮬레이션 경험이 거의 필수적이다. 그러나 심지어 VAX에서도 훈련망이 이를테면 1,000개 이상의 연결들을 가지면 컴퓨트 비율이 참을 수 없을 정도로 느려지게 된다는 점에 주의해야 한다.

나가면서

음악가들이 컴퓨터에 대해 호의적이지 못한 또 다른 이유와 책임은 한편으로 음악가들 자신에게도 있다. 대부분의 음악가들이 컴퓨터의 복잡한 작동원리에 대한 지식의 부족으로 컴퓨터의 활용가능성에 대한 이해가 모자라는 것도 부인할 수 없는 사실이다. 그렇다고 하여 컴퓨터 엔지니어 또는 프로그래머들이 음악에 대한 넓고 깊은 전문적 지식을 소유하고 있으며 또한 관심과 애정을 가지고 있다면, 음악가들에게 환영받을 프로그램을 만들어 낼 수도 있겠으나, 이러한 수동적 입장마저도 기대하기 어려운 편이다. 말하자면, 음악가는 컴퓨터에 관한 아이디어가 모자라고 컴퓨터 프로그래머는 음악적 아이디어가 모자라기 때문에, 둘 사이에 이해의 공통부분이 형성되지 않아 성공적인 커뮤니케이션이 어려운 것이다. 아니 오히려 교감이 모자라다는 편이 옳을 것이다. 물론 음악가가 전문적인 컴퓨터 프로그래머가 되는 것이 최선의 길이기는 하지만 현실적으로는 거의 불가능한 일이다. 따라서 필자의 단견으로는, 양쪽 방향에서의, 즉 음악가는 컴퓨터에 대한, 그리고 프로그래머는 음악에 대한 접근노력이 있어야 그야말로 만족의 수준에 가까이 올 수 있는 컴퓨터 프로그램이 나올 수 있을 것이다. 이 글도, 비록 한 쪽 방향에서일 뿐이지만, 그러한 노력의 일환이다.

〈참고 문헌〉

여기에 열거한 참고문헌들은 이 글에 직접 인용된 것 및 참조한 것, 그리고 앞으로의 연구에 도움이 될 것들을 모은 것이다.

석 봉래 옮김, 『물질과 의식: 현대심리철학입문』 (P.M. Churchland, *Matter and Consciousness*, 1988) 서울: 서광사, 1992.

허 영한, 〈컴퓨터 음합성, 미디, 인터엑티브 시스템: 컴퓨터음악의 현재〉 『음악과 민족』 제 6호, 민족음악연구소, 1993, 161 - 215쪽.

Bharucha, J. J. and P. M. Todd. 1991. "Modeling the Perception of Tonal Structure with Neural Nets." in Loy D.G. & Todd P.M. ed. *Music and Connectionism*, MIT, pp. 128-37.

Buxton, B., W. Reeves, R. Baecker, and L. Mezei. 1978. "The Use of Hierarchy and Instance in a Data Structure for Computer Music." *Computer Music Journal* 2(4): 10-20.

Buxton, B., R. Sniderman, W. Reeves, S. Patel, and R. Baecker. 1979. "The Evolution of the SSSP Score-editing Tools." *Computer Music Journal* 3(4): 14-25.

Byrd, D. 1984. "Music Notation by Computer." Ph.D. diss., Indiana University Department of Computer Science.

Cage, J. 1961. *Silence*. Cambridge, Massachusetts: MIT Press.

Chafe, C., B. Mont-Reynaud, and L. Rush. 1982. "Toward an Intelligent Editor of Digital Audio: Recognition of Musical Constructs." Computer Music Journal 6(1): 30-41.

Churchland P.S. & Sejnowski T.J. ed. *The Computational Brain*, MIT, 1992.

Clynes, M. 1984. "Secrets of Life in Music." Proceedings of the International Computer Music Conference. San Francisco: Computer Music Association, pp. 225-232.

Cope, D. 1990. "Pattern Matching as an Engine for the Computer Simulation of Musical Style." Proceedings of the International Computer Music Conference. San Francisco: Computer Music Association, pp. 288-291.

Dannenberg, R., and G. Bloch. 1985. "Realtime Computer Accompaniment of Keyboard Performances." Proceedings of the International Computer Music Conference. San Francisco: Computer Music Association, pp. 279-290.

Ebcioglu, K. 1984. "An Expert System for Schenkerian Synthesis of Chorales in the Style

of J. S. Bach." Proceedings of the International Computer Music Conference. San Francisco: Computer Music Association, pp. 135-142.

Fux, J. J. 1725. *Gradus ad Parnassum.* Reprinted in 1943 as *Steps to Parnassus.* New York: W.W. Norton.

Gabor, D. 1947. "Acoustical Quanta and the Theory of Hearing." Nature 159 : 591-594.

Gill, S. 1963. "A Techniquc for the Composition of Music in a Computer." *The Computer Journal* 6: 129- 133.

Gjerdingen, R. O. 1991. "Using Connectionist Models to Explore Complex Musical Patterns." in Loy D.G. & Todd P.M. ed. *Music and Connectionism,* MIT, pp. 138-49.

Grossberg, S. 1982. *Studies of Mind and Brain: Neural Principles of Learning. Perception. Development, Cognition, and Motor control.* Boston: Reidel/Kluwer.

Grout, D. J. 1980. *A History of Western Music.* New York: W.W. Norton.

Hiller, L., and L. Isaacson. 1959. *Experimental Music.* New York: McGraw-Hill.

Hinton G.E. 1992. "How Neural Networks Learn from Experience" in *Scientific American,* September : 105-09.

Kirchmeyer, H. 1963. "On the Historical Constitution of a Rationalistic Music." *Die Reihe* 8 : 11-24.

Koenig, G. M. 1970a. "Project One." *Electronic Music Reports* 2. Utrecht: Institute of Sonology.

Koenig, G. M. 1970b. "Project Two." *Electronic Music Reports* 3. Utrecht: Institute of Sonology.

Kohonen, T., P. Laine, K. Tiits, and K. Torkkola, 1991. "A Nonheuristic Automatic Composing Method." *Music and Connectionism,* MIT. 229-42.

Kugel, P. 1990. "Myhill's Thesis: There's More than Computing in Musical Thinking." *Computer Music Journal* 14(3): 13-25.

Lapedes, A., and R. Farber. 1987. "Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling." Technical Report LA-UR-87-2662. Los Alamos National Laboratory.

Laske, O. 1973. "Towards a Musical Intelligence System: OBSERVER." *Numus-West* 1(4):11ff.

Lerdahl, F., and R. Jackendoff. 1983. *A Generative Theory of Tonal Music.* Cambridge, Massachusetts: MIT Press.

Lewis, J. P. 1991. "Creation By Refinement and the Problem of Algorithmic Music

Composition." in Loy D.G. & Todd P.M. ed. *Music and Connectionism,* MIT, pp. 212-28.

Loy, G. 1989. "Composing With Computers--A Survey of Some Compositional Formalisms and Music Programming Languages." In M.V. Matthews and J.R. Pierce, eds. *Current Directions in Computer Music Research.* Cambridge, Massachusetts: MIT Press, pp. 291-396.

Loy D.G. & Todd P.M. ed. *Music and Connectionism,* MIT, 1991.

Mathews, M. V., and L. Rosler. 1968. "Graphical Language for the Scores of Computer-generated Sounds." *Perspectives of New Music* 6 : 92-118.

McIntyre, M.E., R.T. Schumacher, and J. Woodhouse. 1983. "On the Oscillations of Musical Instruments." *Journal of the Acoustical Society of America* 74 : 5.

Meyer, L. 1956. *Emotion and Meaning in Music.* Chicago: Chicago University Press.

Moore, F. R. 1990. *Elements of Computer Music.* Englewood Cliffs, N.J. : Prentice-Hall.

Moorer, J. A. 1975. "On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer." Ph.D. diss., Stanford University Department of Computer Science.

Moorer, J.A. 1985. "Signal Processing Aspects of Computer Music: A Survey." In J. Strawn, ed. *Digital Audio Signal Processing.* Los Altos, California: William Kaufmann, Inc.

Myhill, J. 1952, "Some Philosophical Implications of Mathematical Logic: Three Classes of Ideas." *Review of Metaphysics* 6(2) : 165-198.

Papert, S. 1988. "One AI or Many?" *Daedalus* 118(1): 1-14.

Perle, G., and P. Lansky. 1981. *Serial Composition and Atonality.* Los Angeles: University of California Press.

Potter, G. M. 1971. "The Role of Chance in Contemporary Music." Ph.D. diss., Indiana University Department of Music. Available through University Microfilms.

Roads, C. 1979. "Grammars as Representations of Music." *Computer Music Journal* 3(1): 48-55.

Rodet, X., Y. Potard, and J.-B. Barriere. 1984. "The CHANT Project: From the Synthesis of the Singing Voice to Synthesis in General." *Computer Music Journal* 8(3): 15-31.

Rumelhart, D. E., and J.L. McClelland, eds. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Cambridge, Massachusetts: MIT Press.

Scarborough, D. L., B. O. Miller, and J. A. Jones. 1991. "Connectionist Models for Tonal Analysis." in Loy D.G. & Todd P.M. ed. *Music and Connectionism*, MIT, pp. 54-63.

Scheidt, D. J. 1985. "A Prototype Implementation of a Generative Mechanism for Music Composition." M.S. Thesis. Kingston, Ontario, Canada: Queen's University Department of Computer and Information Science.

Schenker, H. 1906. *Neue Musikalische Theorien und Phantasien.* Universal Editions. Published between 1906 and 1935 in two volumes.

Schillinger, J. 1948. *The Mathematical Basis of the Arts.* New York: The Philosophical Library.

Schillinger, J. 1978. *The Schillinger System of Musical Composition.* New York: Da Capo Press.

Schottstaedt, B. 1984. "Automatic Species Counterpoint." Technical Report STAN-M-19. Stanford: Stanford University, Center for Computer Research in Music and Acoustics.

Sera, X., and J.O. Smith. 1990. "A Sound Decomposition System Based on a Deterministic Plus Residual Model." *Journal of the Acoustical Society of America* 87 : s97 (abstract).

Sundberg, J., and A. Friberg. 1986. "A Lisp Environment for Creating and Applying Rules for Musical Performance." Proceedings of the International Computer Music Conference. San Francisco: Computer Music Association, pp. 1-4.

Tenney, J., and L. Polansky. 1980. "Temporal Gestalt Perception in Music." *Journal of Music Theory* 24(2) : 205-241.

Thomas, M. T. 1985. "VIVACE: A Rule-based AI System for Composition." Proceedings of the International Computer Music Conference. San Francisco: Computer Music Association, pp. 267-274.

Todd, P.M. 1990. "Hierarchical Sequential Networks for Music Composition." *Journal of the Acoustical Society of America* 87 : s18 (abstract).

Todd, P.M. 1991. "A Connectionist Approach to Algorithmic Composition." in Loy D.G. & Todd P.M. ed. *Music and Connectionism*, MIT, pp. 173-94.

Vercoe, B., and M. Puckette. 1985. "Synthetic Rehearsal: Training the Synthetic Performer." Proceedings of the International Computer Music Conference. San Francisco: Computer Music Association, pp. 275-278.

Wawrzynek, J. 1989. "VLSI Models for Sound Synthesis." In M. Matthews and J.R.

Pierce, eds. *Current Directions in Computer Music Research. Cambridge,* Massachusetts: MIT Press.

Weigend, A., B. Huberman, and D.E. Rumclhart. 1990. "Predicting the Future: A Connectionist Approach." Technical Report PDP-90-01. Stanford: Stanford University, PDP Research Group, Department of Psychology.

Williams, R.J., and D. Zipser. 1989. "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks." Neural Computation 1 : 270-280. Winograd, T. 1968. "Linguistics and the Computer Analysis of Tonal Harmony." *Journal of Music Theory* 12 : 2-49.

Xenakis, I. 1971. *Formalized Music.* Bloomington: Indiana University Press.

Zarlino, G. 1558. *Institutioni Harmoniche.* Republished in 1968 as *The Art of Counterpoint.* New York: W. W. Norton.

Abstract

Music and Connectionism: The New Concept of Computer Music

As one of our highest expressions of thought and creativity, music has always been a difficult realm to capture, model, and study. The problem it poses have vexed a variety of scholars using traditional approaches from psychology and artificial intelligence. But the connectionist paradigm, now beginning to provide insights into many realms of human behavior, offers a new and unified viewpoint from which to investigate the subtleties of musical experience. Connectionist systems employ "brain-style" computation, capitalizing on the emergent power of a large collection of individually simple interconnected processors operating and co-operating in a parallel distributed fashion. Models of many aspects of musical behavior require the learning, constraint satisfaction, feature abstraction, and intelligent generalization properties that connectionist approaches embody, at the same time demanding further advancement of these techniques.

This article includes a brief survey on the history of computer music and connectionism, an introduction to connectionism and the traditional back-propagation principle, and a simple musical example to clarify how a neural network performes a certain task in a fahion resembling human judgements. In our country, many a musician does not have much interest in computer music, partly because the concept of von Neuman architecture has a serious limit in capturing, modelling the sounds (and finally the music) of acoustic instruments (involving human voices). This article introduces a fascinating alternative, connectionism which in principle does not have such a limit. But another reason that makes the computer hard to approach for musicians is that musicians are not (does not even want to be) familiar to the basic principles and concepts on which any operation of a computer is based. But to know them is significant and essential to the use of computer and to the development of computer music. This article is also meant to be an endeavour to let the musicians be aware of the importance and significance of those principles and concepts.