(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2008-0098357
(43) 공개일자 2008년11월07일

(51) Int. Cl.
G06T 1/40 (2006.01) *G06N 3/08* (2006.01)
G06J 1/00 (2006.01)
(21) 출원번호 10-2008-7014170
(22) 출원일자 2008년06월12일
심사청구일자 없음
번역문제출일자 2008년06월12일
(86) 국제출원번호 PCT/AU2006/001708
국제출원일자 2006년11월15일
(87) 국제공개번호 WO 2007/056803
국제공개일자 2007년05월24일
(30) 우선권주장
2005906330 2005년11월15일 오스트레일리아(AU)

(71) 출원인
가너, 베르나데트
미합중국, 캘리포니아 95112, 센 조스, 아파트먼트 315, 560 노스 식스 스트리트
(72) 발명자
가너, 베르나데트
미합중국, 캘리포니아 95112, 센 조스, 아파트먼트 315, 560 노스 식스 스트리트
(74) 대리인
서만규, 서경민

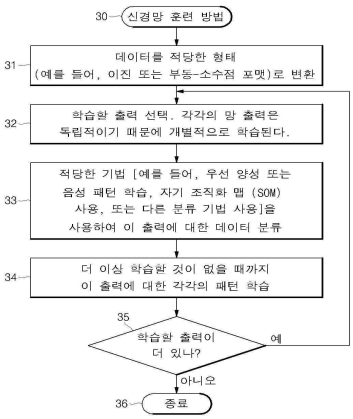
전체 청구항 수 : 총 39 항

(54) 신경망 훈련 방법

(57) 요 약

본 발명은 인공 신경망 (NN) 훈련 방법 (30)을 제공한다. 상기 방법 (30)은 다음 단계들을 포함한다: 훈련될 NN의 출력을 선택하고, 선택된 출력에 대한 NN의 입력 층에 있는 입력 뉴런들에 NN의 출력 뉴런을 연결하여 상기 NN을 초기화하는 단계; NN에 의해 학습될 데이터 세트를 준비하는 단계; 및 NN의 첫 번째 은닉 층, 또는 만일 NN이 은닉 층을 가지고 있지 않다면 NN의 출력 층에 상기 준비된 데이터의 입력 벡터를 적용하여 학습될 NN에 상기 준비된 데이터 세트를 적용하고, NN의 각 층에 있는 선택된 출력에 대한 적어도 하나의 뉴런이 상기 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 있는지 결정하는 단계. 만일 NN 층에 있는 뉴런들 중 어느 것도 상기 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 없다면, 새로운 뉴런이 그 층에 추가되어, 그 층에 있는 다른 뉴런에 의해 학습될 수 없는 연관 출력을 학습한다. 상기 새로운 뉴런은, 훈련되는 출력과 관련된 다음 층에 있는 모든 뉴런들에 연결된 그것의 출력을 갖는다. 만일 출력 뉴런이 상기 입력 벡터를 학습할 수 없다면, 현재의 출력 뉴런 및 모든 입력들이 동일한 층에 직접적으로 연결됨에 따라, 또 다른 뉴런이 상기 동일한 층에 추가된다. 이 뉴런은, 오래된 출력이 학습할 수 없는 입력을 학습한다. 추가 뉴런이 다음 층에 추가된다. 이 뉴런으로의 입력은 NN의 오래된 출력이고, 그 층에 새롭게 추가된 뉴런이다.

대 표 도 - 도2



특허청구의 범위

청구항 1

(i) 훈련될 신경망의 출력을 선택하고, 선택된 출력에 대한 신경망의 입력 층에 있는 입력 뉴런에 신경망의 출력 뉴런을 연결시켜 신경망을 초기화하는 단계;

(ii) 신경망에 의해 학습될 데이터 세트 준비 단계; 및

(iii) 준비된 데이터 세트의 입력 벡터를 신경망의 첫 번째 은닉 층, 또는 만일 신경망이 은닉 층을 갖고 있지 않다면 신경망의 출력 층에 적용하여 학습될 신경망에 준비된 데이터 세트를 적용하고, 신경망의 각 층에 있는 선택된 출력에 대한 적어도 하나의 뉴런이 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 있는지 판단하는 단계를 포함하는 인공 신경망 훈련 방법으로서,

만일 신경망의 각 층에 있는 선택된 출력에 대한 적어도 하나의 뉴런이 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 있고, 만일 학습할 준비된 데이터 세트의 입력 벡터가 더 있다면, 다음 입력 벡터를 위해 단계 (iii)를 반복하고, 만일 훈련될 출력이 더 있다면 신경망의 다음 출력을 위해 단계 (i) 내지 (iii)을 반복하고;

만일 신경망의 선택된 출력에 대한 은닉 층에 있는 어떠한 뉴런도 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 없다면, 새로운 뉴런을 그 층에 추가하여, 선택된 출력에 대한 그 층에 있는 다른 뉴런에 의해 학습될 수 없는 연관 출력을 학습하고, 만일 학습할 데이터 세트의 입력 벡터가 더 있다면 다음 입력 벡터를 위해 단계 (iii)을 반복하고, 만일 훈련될 출력이 더 있다면 신경망의 다음 출력을 위해 단계 (i) 내지 (iii)을 반복하고;

만일 신경망의 선택된 출력에 대한 출력 뉴런이 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 없다면, 출력 뉴런이 신경망의 은닉 층 뉴런이 되고, 새로운 뉴런을 이 은닉 층에 추가하여 출력 뉴런에 의해 학습될 수 없는 연관 출력을 학습하고, 새로운 출력 뉴런을 선택된 출력에 대한 신경망에 추가하고, 만일 학습할 데이터 세트의 입력 벡터가 더 있다면, 다음 입력 벡터를 위해 단계 (iii)를 반복하고, 만일 훈련될 출력이 더 있다면 신경망의 다음 출력을 위해 단계 (i) 내지 (iii)을 반복하는 인공 신경망 훈련 방법.

청구항 2

(i) 신경망에 의해 학습될 데이터 세트 준비 단계;

(ii) 훈련될 신경망의 출력을 선택하고, 선택된 출력에 대한 신경망의 입력 층에 있는 입력 뉴런에 신경망의 출력 뉴런을 연결하여 신경망을 초기화하는 단계; 및

(iii) 준비된 데이터 세트의 입력 벡터를 신경망의 첫 번째 은닉 층, 또는 만일 신경망이 은닉 층을 갖고 있지 않다면 신경망의 출력 층에 적용하여 학습될 신경망에 준비된 데이터 세트를 적용하고, 신경망의 각 층에 있는 선택된 출력에 대한 적어도 하나의 뉴런이 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 있는지 판단하는 단계를 포함하는 인공 신경망 훈련 방법으로서,

만일 신경망의 각 층에 있는 선택된 출력에 대한 적어도 하나의 뉴런이 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 있고, 만일 학습할 준비된 데이터 세트의 입력 벡터가 더 있다면, 다음 입력 벡터를 위해 단계 (iii)를 반복하고, 만일 훈련될 출력이 더 있다면 신경망의 다음 출력을 위해 단계 (ii) 내지 (iii)을 반복하고;

만일 신경망의 선택된 출력에 대한 은닉 층에 있는 어떠한 뉴런도 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 없다면, 새로운 뉴런을 그 층에 추가하여, 선택된 출력에 대한 그 층에 있는 다른 뉴런에 의해 학습될 수 없는 연관 출력을 학습하고, 만일 학습할 데이터 세트의 입력 벡터가 더 있다면 다음 입력 벡터를 위해 단계 (iii)을 반복하고, 만일 훈련될 출력이 더 있다면 신경망의 다음 출력을 위해 단계 (ii) 내지 (iii)을 반복하고;

만일 신경망의 선택된 출력에 대한 출력 뉴런이 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 없다면, 출력 뉴런이 신경망의 은닉 층 뉴런이 되고, 새로운 뉴런을 이 은닉 층에 추가하여 출력 뉴런에 의해 학습될 수 없는 연관 출력을 학습하고, 새로운 출력 뉴런을 선택된 출력에 대한 신경망에 추가하고, 만일 학습할 데이터 세트의 입력 벡터가 더 있다면 다음 입력 벡터를 위해 단계 (iii)를 반복하고, 만일 훈련될 출력이 더 있다면 신경망의 다음 출력을 위해 단계 (ii) 내지 (iii)을 반복하는 인공 신경망 훈련 방법.

청구항 3

제 1항 또는 제 2항에 있어서, 상기 신경망의 뉴런이 선형 임계 게이트 (LTG)인 것을 특징으로 하는 인공 신경망 훈련 방법.

청구항 4

제 3항에 있어서, 상기 단계 (iii)에서 LTG가 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 있는지 결정하는 것은, 상기 LTG가 이전에 학습한 것이 주어진다면, 상기 LTG의 가중치와 임계치 사이의 관계가 해를 갖는지 결정하는 것임을 특징으로 하는 인공 신경망 훈련 방법.

청구항 5

제 4항에 있어서, 상기 관계가 제약이고, 상기 입력 벡터와 상기 LTG의 가중치 벡터가 상기 신경망의 선택된 출력을 기초로 한 LTG의 임계치와의 관계를 형성하는 것을 특징으로 하는 인공 신경망 훈련 방법.

청구항 6

제 5항에 있어서, 제약을 학습하는 것은, LTG의 제약 세트에 상기 제약을 추가할 수 있는 것임을 특징으로 하는 인공 신경망 훈련 방법.

청구항 7

제 6항에 있어서, LTG의 제약 세트에 상기 제약을 추가할 수 있기 위해, 모든 제약들 사이에 해가 있어야만 함을 특징으로 하는 인공 신경망 훈련 방법.

청구항 8

제 6항 또는 제 7항에 있어서, 상기 신경망을 초기화하는 단계가, 상기 출력 LTG의 제약 세트를 제거하는 단계를 더 포함하여, 상기 출력 LTG의 제약 세트가 비워지는 것을 특징으로 하는 인공 신경망 훈련 방법.

청구항 9

제 1항 내지 제 8항 중 어느 한 항에 있어서, 상기 신경망에 의해 학습될 데이터 세트를 준비하는 단계가, 훈련 동안 데이터 세트가 신경망에 제공되기 전에 사전-정의된 데이터 포맷으로 데이터 세트를 변환시키는 단계; 훈련 동안 상기 데이터 세트가 신경망에 제공되기 전에 상기 데이터 세트에 불일치가 있는지 결정하는 단계; 훈련 동안 상기 데이터 세트가 신경망에 제공되기 전에 상기 데이터 세트를 분류하는 단계; 및 훈련 동안 상기 데이터 세트가 신경망에 제공되기 전에 상기 데이터 세트에 0 입력 벡터가 이용가능한지 결정하고, 여기서 만일 상기 0 입력 벡터가 상기 데이터 세트에 이용가능하다면, 상기 데이터 세트가 정렬되어, 상기 0 입력 벡터가 처음에 훈련될 신경망에 제공되는 단계를 포함하고, 이들 단계 각각이 임의의 순서로 수행될 수 있는 것을 특징으로 하는 인공 신경망 훈련 방법.

청구항 10

제 9항에 있어서, 상기 사전-정의된 데이터 포맷이 이진 또는 부동-소수점 데이터 포맷인 것을 특징으로 하는 인공 신경망 훈련 방법.

청구항 11

제 9항 또는 제 10항에 있어서, 상기 데이터 세트가 신경망에 제공되기 전에 상기 데이터 세트에 불일치가 있는지 결정하는 단계는, 다른 출력을 생산하는 두 개 이상의 동일한 입력 벡터가 있는지 결정하는 단계를 포함하는 것을 특징으로 하는 인공 신경망 훈련 방법.

청구항 12

제 11항에 있어서, 만일 두 개 이상의 동일한 입력 벡터가 다른 출력을 생산하는 것으로 결정된다면, 상기 입력 벡터 중 단 하나만 사용되는 것을 특징으로 하는 인공 신경망 훈련 방법.

청구항 13

제 9항 내지 제 12항 중 어느 한 항에 있어서, 훈련 동안 상기 데이터 세트가 신경망에 제공되기 전에 상기 데이터 세트를 분류하는 단계가, 상기 데이터 세트의 입력 벡터를 두 개의 세트로 분류하고, 그 출력에 대해 0을 생산하는 입력 벡터로부터 1을 출력하는 입력 벡터를 분리하고, 먼저 훈련될 두 개의 세트 중 하나를 선택하는 단계; 자기 조직화 맵 (SOM)으로 상기 데이터를분류하는 단계; 및/또는 다른 적합한 방법을 사용하여 상기 데이터를 분류하는 단계를 포함하는 것을 특징으로 하는 인공 신경망 훈련 방법.

청구항 14

제 13항에 있어서, 훈련 동안 상기 데이터가 신경망에 제공되기 전에, 각각의 입력 층에 대한 단일 리스트가 상기 분류된 데이터로부터 만들어지는 것을 특징으로 하는 인공 신경망 훈련 방법.

청구항 15

제 5항에 있어서, 만일 새로운 LTG가 층에 추가되어, 단계 (iii)에 따라 그 층에 있는 다른 LTG에 의해 학습될 수 없는 제약을 학습한다면: 상기 새로운 LTG가, 신경망의 선택된 출력에 기여하는 다음 층에 있는 모든 LTG들에 연결되고, 상기 새로운 LTG로부터 입력을 받는 다음 층에 있는 LTG들의 제약 세트가 갱신되어 상기 새로운 LTG로부터 입력을 수용하고; 만일 상기 새로운 LTG를 지닌 층이 신경망의 첫 번째 층이 아니라면, 상기 새로운 LTG가 신경망의 선택된 출력에 기여하는 이전 층에 있는 모든 LTG들에 연결되어 그것들로부터 입력을 수용하고; 상기 새로운 LTG의 제약 세트가 갱신되어 그 층에 있는 이전의 마지막 LTG의 변경된 제약 세트 및 그 층에 있는 다른 LTG에 의해 학습될 수 없는 제약의 사본을 포함하는 것을 특징으로 하는 인공 신경망 훈련 방법.

청구항 16

제 5항 또는 제 15항에 있어서, 새로운 출력 LTG가 단계 (iii)에 따라 신경망에 추가된다면: 상기 새로운 LTG가, 은닉 층에 있는 LTG들에 연결되어 그것들로부터 입력을 수용하고; 만일 상기 은닉 층이 신경망의 첫 번째 층이 아니라면, 상기 은닉 층에 있는 새로운 LTG가 신경망의 선택된 출력에 기여하는 이전 층에 있는 모든 LTG들에 연결되어 그것들로부터 입력을 수용하고; 상기 은닉 층에 추가된 새로운 LTG의 제약 세트가 갱신되어 그 층에 있는 이전의 출력 LTG의 변경된 제약 세트 및 상기 이전의 출력 LTG에 의해 학습될 수 없는 제약의 사본을 포함하고; 상기 새로운 출력 LTG가, 상기 이전의 출력 LTG에 의해 학습될 수 없는 것에 따라 사전-정의된 논리 관계로 그것의 입력을 결합시키는 것을 특징으로 하는 인공 신경망 훈련 방법.

청구항 17

제 16항에 있어서, 새로운 출력 LTG가 단계 (iii)에 따라 신경망에 추가될 때, 상기 새로운 출력 LTG로의 입력들 사이에 형성된 사전-정의된 논리적 관계가 논리 OR, 논리 AND, 또는 다른 적당한 논리 관계인 것을 특징으로 하는 인공 신경망 훈련 방법.

청구항 18

제 17항에 있어서, 만일 상기 이전의 출력 LTG에 의해 학습될 수 없는 입력 벡터가 출력 1을 생산한다면 논리 OR이 사용되고, 만일 상기 이전의 출력 LTG에 의해 학습될 수 없는 입력 벡터가 출력 0을 생산한다면 논리 AND가 사용되는 것을 특징으로 하는 인공 신경망 훈련 방법.

청구항 19

선택된 출력에 대한 그 층에 있는 다른 뉴런이, 학습되는 데이터 세트의 입력 벡터와 연관된 관계를 학습할 수 없을 때 새로운 뉴런을 신경망에 추가하는, 훈련 동안 신경망 층에 새로운 뉴런을 추가하는 방법으로서,

그 층에 있는 신경망의 선택된 출력에 기여하는 이전의 마지막 뉴런으로부터 변경된 모든 데이터 및 그 층에 있는 다른 뉴런에 의해 학습될 수 없는 관계의 사본으로 상기 새로운 뉴런을 갱신하는 단계; 및

출력 뉴런을 갱신하여, 상기 새로운 뉴런으로부터 입력을 수용하는 단계를 포함하는, 훈련 동안 신경망 층에 새로운 뉴런 추가 방법.

청구항 20

제 19항에 있어서, 상기 신경망의 뉴런이 LTG인 것을 특징으로 하는, 훈련 동안 신경망 층에 새로운 뉴런 추가 방법.

청구항 21

제 20항에 있어서, 상기 관계가 LTG의 가중치와 임계치 사이의 관계인 것을 특징으로 하는, 훈련 동안 신경망 층에 새로운 뉴런 추가 방법.

청구항 22

제 20항 또는 제 21항에 있어서, 상기 관계는 제약이고, 상기 데이터 세트의 입력 벡터와 상기 LTG의 가중치 벡터가 신경망의 출력을 기초로 한 상기 LTG의 임계치와 관계를 형성하는 것을 특징으로 하는, 훈련 동안 신경망 층에 새로운 뉴런 추가 방법.

청구항 23

이진 포맷 데이터 세트 이외의 데이터 세트를 신경망에 의해 학습될 이진 포맷 데이터 세트로 변환시키는 방법으로서,

(i) 상기 데이터 세트의 속성 각각을 이진수로 나타내기 위한 비트 수를 개별적으로 결정하는 단계;

(ii) 방정식 [범위 = (최대값 - 최소값) + 1]을 사용하여 상기 데이터 세트의 속성 범위를 계산하는 단계; 및

단계 (i)에서 결정된 비트 수를 사용하여 상기 데이터 세트의 속성 범위를 이진수로 부호화하는 단계를 포함하는 변환 방법.

청구항 24

제 1항 또는 제 2항의 신경망 훈련 방법에 따라 훈련될 데이터를 준비하기 위한, 제 23항의 방법의 사용.

청구항 25

신경망에 의해 훈련될 데이터 세트를 분류하는 방법으로서,

상기 데이터 세트의 입력 벡터들을 2개의 그룹으로 분류하는 단계; 0을 출력하는 입력 벡터들로부터 1을 출력하는 입력 벡터들을 분리하는 단계; 및 신경망에 의해 먼저 학습될 2개의 그룹 중 하나를 선택하는 단계를 포함하는 분류 방법.

청구항 26

제 1항 또는 제 2항의 신경망 훈련 방법에 따라 훈련될 데이터를 분류하기 위한, 제 25항의 방법의 사용.

청구항 27

입력 벡터가 뉴런에 의해 드러나는지 또는 드러나지 않는지 결정하는 방법으로서,

상기 입력 벡터로부터 제약 및 그것의 보수를 구성하는 단계; 뉴런의 제약 세트에 상기 제약 및 그것의 보수를 교대로 추가하는 단계; 상기 두 경우에 해가 있는지 결정하기 위해 제약 세트를 테스트하는 단계, 여기서: 만일 상기 제약 또는 그것의 보수에 대한 해가 없다면, 상기 입력 벡터가 뉴런에 의해 드러나는 것으로 결정되는 단계; 및 만일 상기 제약 및 그것의 보수가 상기 제약 세트에 교대로 추가될 때 해가 있다면, 상기 입력 벡터가 뉴런에 의해 드러나지 않는 것으로 결정되는 단계를 포함하는 결정 방법.

청구항 28

제 27항에 있어서, 상기 제약 세트는 LTG 뉴런으로 구성된 신경망 뉴런의 제약 세트인 것을 특징으로 하는 결정 방법.

청구항 29

제 1항 또는 제 2항의 신경망 훈련 방법에 따라 훈련된 신경망의 보이지 않는 입력 벡터의 출력을 결정하기 위한, 제 27항의 방법의 사용.

청구항 30

보이지 않는 입력 벡터에 대한 디폴트 출력이 상기 데이터 세트에 따라 1 또는 0이 되는 것을 특징으로 하는,

제 3항의 신경망 훈련 방법에 따라 훈련된 신경망의 LTG의 보이지 않는 입력 벡터를 결정하기 위한, 제 27항의 방법의 사용.

청구항 31

제약 세트의 최소 활성화 체적 (MAV)을 결정하는 방법으로서,

(i) 한 번에 하나씩 상기 제약 세트로부터 각각의 제약을 제거하며, 상기 제약 세트에 남아있는 제약들을 변하지 않게 두는 단계;

(ii) 상기 제거된 제약의 보수를 상기 제약 세트에 추가하는 단계;

(iii) 해가 있는지 보기 위해 새로운 제약 세트를 테스트하는 단계로서, 여기서:

만일 해가 있다면, 원래의 제약을 MAV를 정의하는 제약 세트에 추가하고, 상기 제약 세트에 추가된 제약의 보수를 제거하여 원래의 제약을 상기 제약 세트로 되돌리고, 만일 상기 제약 세트에 테스트할 제약이 더 있다면 단계 (i) 내지 (iii)을 반복하고, 그렇지 않다면 상기 MAV는 상기 MAV를 정의하는 제약 세트 내에 포함된 제약들의 세트인 것이고;

만일 해가 없다면, 상기 제약 세트에 추가된 세트의 보수를 제거하여 원래의 제약을 상기 제약 세트로 되돌리고, 만일 상기 제약 세트에 테스트할 제약이 더 있다면, 단계 (i) 내지 (iii)를 반복하고, 그렇지 않으면 상기 MAV는 상기 MAV를 정의하는 제약 세트 내에 포함된 제약들의 세트인 것인 테스트 단계를 포함하는, 최소 활성화 체적 결정 방법.

청구항 32

제 31항에 있어서, 상기 제약 세트는 LTG 뉴런으로 구성된 신경망 뉴런의 제약 세트인 것을 특징으로 하는 최소 활성화 체적 결정 방법.

청구항 33

제 1항 또는 제 2항의 신경망 훈련 방법에 따라 훈련된 신경에 있는 각각의 LTG에 대한 MAV를 결정하기 위한, 제 32항의 방법의 사용.

청구항 34

사실상 첨부된 도면을 참조하여 앞에서 설명한 것과 같은, 인공 신경망 훈련 방법.

청구항 35

사실상 첨부된 도면을 참조하여 앞에서 설명한 것과 같은, 훈련 동안 신경망 층에 새로운 뉴런을 추가하는 방법.

청구항 36

사실상 첨부된 도면을 참조하여 앞에서 설명한 것과 같은, 이진 포맷 데이터 세트 이외의 데이터 세트를 신경망에 의해 학습될 이진 포맷 데이터 세트로 변환시키는 방법.

청구항 37

사실상 첨부된 도면을 참조하여 앞에서 설명한 것과 같은, 신경망에 의해 훈련될 데이터 세트를 분류하는 방법.

청구항 38

사실상 첨부된 도면을 참조하여 앞에서 설명한 것과 같은, 제약 세트의 입력 벡터가 드러나는지 드러나지 않는지 결정하는 방법.

청구항 39

사실상 첨부된 도면을 참조하여 앞에서 설명한 것과 같은, 제약 세트의 최소 활성화 체적 (MAV)를 결정하는 방법.

명 세 서

기 술 분 야

<1> 본 발명은 일반적으로 인공 신경망 및 그들의 작동에 관한 것으로, 특히 이에 한정되지 않지만, 훈련 과정 동안 필요한 대로 뉴런들이 망 (network)에 추가되도록 하는 향상된 신경망 훈련 방법 및/또는 시스템에 관한 것이다.

배 경 기 술

<2> 지난 10년에 걸쳐 생성된 데이터 세트 (data set)의 크기 및 급증에 따라, 데이터 세트 내의 관계들을 찾기 위해 사용될 수 있는 장비 (tool) 개발에 많은 관심이 쏟아졌으나, 이러한 데이터 세트를 명확히 이해하지는 못했다. 데이터를 조사할 수 있는 도구를 통해, 고정된 시간 내에 매번 일관되게 데이터 세트를 학습하여, 입력 (input)과 출력 (output) 사이의 관계에 대한 뚜렷한 정보가 쉽게 결정되도록 하는 것이 바람직하다.

<3> 데이터를 조사하기 위해 사용된 하나의 도구는 피드-포워드 (feed-forward) 신경망이다. 피드-포워드 신경망은, 지난 40년에 걸쳐 큰 관심을 이끌며, 데이터 세트로 많은 다양하고 어려운 임무들을 수행하기 위해 사용되었다. 여기에는 패턴 분류 (pattern classification) 및 함수 근사 (function approximation)가 포함되는데, 이는 그들의 "일반화 (generalise)" 능력 때문이다. 따라서, 신경망 (이하, "NN들" 또는 "NN"이라 함)은 비선형 (non-linear) 시스템 모델링 (system modelling)과 이미지 압축 및 재구성과 같은 애플리케이션 (application)에 사용될 수 있다.

<4> NN은 과학, 상업, 의학 및 산업을 포함한 많은 분야에서 관심의 대상인데, 이는 NN이 어떠한 관계가 데이터에 내재하는지 알려지지 않은 주어진 데이터 세트일 수 있고, NN이 상기 데이터를 성공적으로 분류하는 방법을 학습할 수 있기 때문이다.

<5> 데이터가 사전에 분류될 수 없는 경우, 이러한 상황에서는, 자기-조직화 맵 (self-organising map)과 같은 자율 훈련 (unsupervised training)을 사용하여 데이터를 분류하는 것이 일반적이다. 반면 데이터가 분류된 데이터 샘플로 사전에 나눠질 수 있는 경우, 이러한 상황에서는, NN을 훈련시켜 추가 미분류 데이터를 분류할 수 있게 하는 것이 일반적이다. 후자의 경우, 지도 학습 알고리즘 (supervised learning algorithm)이 전통적으로 사용된다. 분류된 입력 데이터 예들은 연관된 출력을 갖고, 훈련 동안, NN은 입력 벡터 (input vector)와 연관된 원하는 출력을 재생산하는 것을 학습한다. 피드-포워드 NN은 전통적으로 지도 학습 방법을 사용하여 훈련된다.

<6> 인공 NN은 수 많은 뉴런들로 이루어지고, 이들은 때때로 유니트 (unit) 또는 노드 (node)라 불린다. 그들은 생물학적 뉴런에서 영감을 얻은 것이다. 뉴런들은 망 (network)을 형성하기 위해 서로 연결된다. 각각의 뉴런은 많은 다른 뉴런들로부터 나올 수 있는 입력을 갖는다.

<7> 뉴런은 발화 (firing)에 의해 또는 발화 없이, 입력에 반응하여 출력을 생산한다. 그런 다음, 뉴런의 출력은 많은 다른 뉴런들에 입력을 제공할 수 있다. 이것이 피드-포워드 NN의 기본 구조이다.

<8> 일반적으로, 뉴런은 층을 형성한다. 피드-포워드 NN에는 세 가지 유형의 층, 즉 입력, 은닉 (hiddin) 및 출력 층이 있다. 첫 번째 층은 입력 층으로, 층에 하나 이상의 뉴런들을 갖는다. 출력 층 또한 하나 이상의 뉴런들을 가질 수 있다. 또한, NN은 하나 이상의 은닉 층들을 가질 수 있다. 입력 층에 있는 모든 뉴런들은 다음 층에 그들의 출력을 제공하는데, 상기 다음 층은 출력 층이거나, 만일 하나 이상의 은닉 층들이 존재한다면 첫 번째 은닉 층일 수 있다. 만일 오직 하나의 은닉 층이 있다면, 상기 은닉 층에 있는 뉴런들은 교대로 그들의 출력을 출력 층에 보고할 것이다. 만일 하나 이상의 은닉 층들이 있다면, 마지막 은닉 층의 뉴론이 출력 층의 입력에 그들의 출력을 공급할 때까지, 그 뉴런들은 다음 은닉 층 등에 있는 뉴런들의 입력에 그들의 출력을 공급할 것이다.

<9> 다른 망 구조 또한 가능한데, 여기서 NN은 특히 특정 데이터 세트를 학습하도록 설계된다. 이런 구조는 특히 입력 벡터의 NN 학습 순서 (learning sequence)에 나타나고, 이 연결 (connections)에서 피드백 고리 (feedback loop)를 가질 수 있다. 이러한 NN은 순환 (recurrent) 피드-포워드 NN이라 불리고, 일반적으로 NN의 출력은 종종 NN의 입력으로의 피드백될 수 있다.

<10> 첫 번째 생물학적 뉴런 모델은 1943년 맥컬럭 및 피트 (McCulloch and Pitt)에 의해 개발되었다. 이 모델은 맥컬럭-피트 뉴런으로 알려졌다. 이러한 맥컬럭-피트 뉴런 모델 또는 선형 임계 게이트 (linear threshold gate) (이하, "LTG" 또는 "LTG들"라 함)는 수많은 입력 연결들을 갖는 것으로 정의되며, 각각의 연결은 그것과 연관된

가중치 (weight)를 갖는다. 입력은 수리적 (mathmatically)으로 벡터, $x_i \in \{0,1\}^n$으로 정의되는데, 여기서 n은 LTG로의 입력 수를 나타내는 양수 (positive integer)이고, i는 i^{th} 입력 벡터이다. n 입력 연결들이 있기 때문에, 연결 가중치는 수리적으로 벡터, w (여기서, $w \in R^n$)로 정의될 수 있다. LTG로의 입력 벡터 각각은 그것의 연관된 가중치로 곱해져, 수리적으로 $x_i.w$로 나타낼 수 있고, 그 결과는 LTG 임계치 (threshold value), T (여기서, $T \in R$)와 비교된다. 출력은 만일 $x_i.w \geq T$라면 1일 것이고, $x_i.w < T$라면 0일 것이다. 즉, LTG는, 뉴런의 활성화 함수로, 단계 (step), 헤비사이드 (Heaviside), 함수를 사용한다.

<11> 상기 LTG는 다음의 정의를 사용하여 수리적으로 정의될 수 있다:

<12> $w = \{w_1, w_2, \ldots w_n\}$ 및 $x_i = \{x_1, x_2, \ldots x_n\}$

<13> $net_n = x_i.w$ 및 $x_i \in \{0,1\}^n$ 및 $w \in R^n$이면, LTG의 작동은 다음과 같이 방정식 1.1로 요약될 수 있다:

<14> $x_i.w < T \rightarrow 0$ 및 $x_i.w \geq T \rightarrow 1$ (1.1)

<15> 따라서, LTG의 출력, *O*은 이진수 (binary){0,1}이다. LTG는, 만일 활성화되면 1을 출력할 것이고, 만일 그렇지 않다면 0을 출력할 것이다.

<16> 상기 LTG는 1962년에 1로 영원히 설정된 추가 바이어스 (bias) 입력으로 변경되었다. 상기 바이어스 입력은 임계치를 흡수한 후, 0으로 설정된다. 상기 변경된 LTG 모델은 퍼셉트론 (perceptron)으로 개명되었다. 상기 퍼셉트론 모델은 임계치, T가 $x_i.w$로부터 제거되도록 하고, 따라서 방정식이 $x_i.w < T \equiv x_i.w - T < 0$ 및 $x_i.w \geq T \equiv x_i.w - T \geq 0$이 된다. 이제, 임계치는 가중치 w_0와 함께, 뉴런으로의 또 다른 입력이 될 수 있고, 뉴론으로의 입력을 1로 고정하면, 항상 존재하게 되어, $T = 1 \cdot w_0$이 된다. 상기 가중치 w_0는 바이어스 가중치로 불린다. 따라서 방정식은 다음처럼 된다:

<17> $x_i.w - w_0 < 0$ 및 $x_i.w - w_0 \geq 0$.

<18> 1960년에, 로젠블라트 (Rosenblatt)는 퍼셉트론 모델을 사용하여 가중치의 숫자 값을 찾는 데 관심을 쏟았다. 그때부터 현재까지, 뉴론에 있는 가중치 각각의 단일 숫자 값을 찾는 것이 뉴런 및 NN의 확립된 훈련 방법이 되었다. 뉴런들에 의해 형성된 관계들이 명제 논리 (propositional logic)를 사용하여 나타낼 수 있다는 것이 인정됨에도 불구하고, 가중치와 임계치 사이의 기호적 (symbolic) 관계를 직접적으로 찾으려는 시도는 없었다. 훈련 동안 NN이 학습한 데이터 세트 내의 규칙은 숫자 값으로 부호화되고, 이것으로 규칙들을 압축 불가능하게 할 수 있다. 가중치와 임계치에 의해 찾아진 숫자로부터 NN에 의해 학습된 규칙을 찾으려는 시도가 계속되어 왔다. 이러한 방법들은 모두, 규칙이 NN으로부터 직접적으로 읽히지 못하도록 하는 훈련 이후의 추가 과정이다.

<19> 1962년에, 로젠블라트는 퍼셉트론 학습 알고리즘의 수렴도 (convergence), 즉 이것이 선형적으로 분리 가능한 데이터 세트를 만족시키는 숫자를 반복하여 찾는 것을 증명하였다. 특정 입력이 주어지면 원하는 출력을 생산하도록, 뉴런은 연결 가중치를 변경하여 학습한다. 방정식 1.2에서 볼 수 있는 것처럼, 로젠블라트의 훈련 규칙은, 가중치, w_j (여기서, $1 \leq j \leq n$이고, n은 퍼셉트론으로의 입력의 수이다)가 입력, x_i (t는 시간 단계이다) 및 양성 수득률 (positive gain rate), η (여기서, $0 \leq \eta \leq 1$이다)을 기초로 변경되는 것이다. 로젠블라트의 규칙은 이진 출력에 사용된다. 만일 특정 입력에 대한 퍼셉트론의 출력이 옳다면, 아무것도 할 것이 없다.

<20> $w_j(t+1) = w_j(t)$ (1.2)

<21> 그렇지 않으면, 만일 출력이 0이고 1이어야만 한다면:

<22> $w_j(t+1) = w_j(t) + \eta w_i(t)$ (1.3)

<23> 또는, 만일 출력이 1이고 0이어야만 한다면:

<24> $w_j(t+1) = w_j(t) - \eta w_i(t)$ (1.4)

<25> 가중치를 반복적으로 조절하고자 했던 사고가 현재 피드-포워드 NN의 확립된 훈련 방법이 되었다.

<26> 1969년, 로젠블라트의 학습 알고리즘이 더 복잡한 데이터 세트에는 적용될 수 없다는 것이 발견되었다. 민스키

(Minsky) 및 파페르트 (Papert)는, 단층 (single layer) 퍼셉트론이 유명한 배타적 (exclusive) 또는 (XOR) 문제를 풀 수 없다는 것을 증명하였다. 그것이 적용될 수 없는 이유는, 반복이 가중치-공간 (space)에서 단일 포인트 (single point)를 찾는 데 사용되었기 때문이다.

<27> 모든 부울 함수 (Boolean function)가 단일 LTG에 의해 학습될 수 있는 것은 아니다. 거기에는 n 입력 변수의 2^n 조합 (combination)이 있고, 가능한 출력과 결합됐을 때, 이것은 2^{2n} 유일한 부울 함수 [스위칭 함수 (switching function)라고도 알려짐]가 존재한다는 것을 의미한다. 상기 2^{2n} 함수들 가운데 오직 일부만이 단일 n-입력 LTG로 나타낼 수 있다. 입력 공간이 선형적으로 분리 가능한 부울 함수들은 단일 LTG로 나타낼 수 있지만, 선형적으로 분리 가능하지 않은 부울 함수들을 학습하기 위해서는 추가 LTG들이 필요하다. XOR은, 선형적으로 분리 가능하지 않아, 단일 LTG에 의해 학습될 수 없는 부울 함수의 일 예이다.

<28> LTG의 추가 층들을 사용하여, 선형적으로 분리 가능하지 않은 문제들이 NN에 의해 학습되도록 할 수 있었지만, LTG의 다층 (multiple layer)이 그때에 훈련되도록 이용할 수 있는 훈련 규칙은 없었다.

<29> 결과적으로, LTG의 다층이 훈련되도록 하는 가중치 및 임계치의 숫자 값을 찾기 위한 반복적인 방법이 없었기 때문에, 뉴런의 맥컬럭-피트 모델은 폐기되었다. 이 모델은 역전파 (backpropagation)가 개발될 때 비로소 사용되었다.

<30> 1974년, 웨보스 (Werbos)는 오차 역전파 (error backpropagation)[또는, "역전파"]라는 개념을 생각해냈다. 1986년 후반에 루멜하트 (Rumelhart) 및 힌톤 (Hinton), 1986년에 윌리암 (William) 및 1985년에 파커 (Parker) 역시 동일한 알고리즘을 생각해냈고, 그것은 다층 NN 모델이 가중치의 숫자 값을 반복적으로 찾도록 훈련되도록 하였다. 이를 통해, 단층 퍼셉트론이 해결할 수 없었던 많은 다른 문제들뿐만 아니라 XOR 문제가 해결되었다. 맥컬럭-피트의 뉴런 모델은, 그것의 활성화 함수로서, 계단 (step) 함수가 아닌 시그모이드 (sigmoid) 함수를 사용하기 위해 변경되었다. 상기 시그모이드 함수의 수리적 정의는 방정식 1.5에 나와있다.

<31> $0 = 1/(1 + e^{-kx \cdot w})$ (1.5)

<32> 퍼셉트론은 일반적으로 퍼셉트론의 활성화 함수로서 시그모이드 함수를 사용한다. 상기 단어 k는 곡선의 퍼짐 (spread)을 조절하고, 시그모이드 함수는 $k \rightarrow \infty$, 출력, $O \rightarrow$ 계단 함수로서 계단-함수에 가까워진다. 하지만, $\tanh(kx \cdot w)$와 같은 다른 활성화 함수를 사용하는 것 또한 가능하다. 만일 NN이 -1 내지 +1의 함수 범위로 음수 (negative number)를 출력할 수 있는 것이 필요하다면, 이 활성화 함수가 사용된다.

<33> 역전파는 로젠블라트의 학습 알고리즘을 기초로 하는데, 이것은 방정식 1.2 내지 1.4로 설명된다. 그것은 지도 학습 알고리즘이고, 입력 벡터를 NN의 입력 층에 적용함으로써 작동한다. 입력층은 첫 번째 은닉 층에 이러한 입력을 분배한다. 한 층에 있는 뉴런 각각의 출력은 방정식 1.5에 따라 계산되고, 이것이 그 다음 층으로의 입력이 된다. 그 다음 층에 입력이 되는 뉴런 층의 출력 (또는 활성화)을 계산하는 과정은, NN의 출력이 계산될 수 있을 때까지 반복된다. 실제 출력과 원하는 출력 사이에 일부 오차가 있을 것이고, 가중치는 오차의 양에 따라 변경된다. NN에서의 오차를 줄이기 위해, 출력 층으로의 연결로부터 은닉 층 상의 연결까지 차례대로 연결 가중치를 조절하여, NN을 통해, 출력에서의 오차가 피드백 또는 역전파된다. 조절된 가중치 양은 직접적으로 유니트 (unit)에 있는 오차 양에 비례한다.

<34> 역전파 델타 규칙은 방정식 1.6에 나와있고, 여기서 i는 층이고, j는 층 i-1에서 연결이 유래한 퍼셉트론이고, k는 층 i에서 연결이 시작된 퍼셉트론이다.

$$w_{ijk}^{new} = w_{ijk}^{old} + \Delta w_{ijk} \qquad \textbf{(1.6)}$$

여기서, $\Delta w_{ijk} = \eta \delta_{ijk} o_{ijk}$

<35>

<36> $\triangle w_{ijk}$는, NN에 있는 가중치 상의 숫자 값에서 오차를 줄이기 위해 변경된 가중치가 양이다. 가중치가 변경된 양은 뉴런의 출력, o_{ijk}, 획득 용어 (gain term) η를 기초로 하고, 이것은 또한 출력, δ_{ijk}에서의 오차 및 학습율 (learning rate)이라 불린다. NN의 오차는 NN의 실제 출력과 원하는 출력 사이의 차이이다.

<37> NN이 완전히 훈련될 때, NN에서의 오차가 최소이기 때문에, 그것은 오차 함수의 대역 최소치 (global minimum)

에 있는 것으로 나타난다. 오차에 잠재적으로 많은 국소 최소치 (local minima)가 있기 때문에, 상기 오차는 면(surface)으로 여겨질 수 있고, 이는 그것이 함수일 수 있다는 것을 암시한다. 하지만, 오차 함수는 어떠한 NN에도 드러나지 않는다. 상기 오차 함수는, NN에 적용된 모든 입력 벡터에 대한 실제 출력과 원하는 출력 사이의 차이를 기초로 하기 때문에, 오직 경험적으로 계산될 수 있다. 상기 용어 δ_{ijk}는 오차 함수의 1차 미분(derivative)[미분은 출력에서의 오차 차이를 기초로 한다]이다. 그것은, 역전파가 NN에서의 오차를 최소화하려고 함에 따라 최소화될 오차 함수이다. 기울기 (gradient)(1차 미분)를 이용하여, NN에서의 오차를 최소화하기 위해 가중치를 어떻게 변경할지 결정하는 것이 가능하다. 이것을 기울기-하강 (gradient descent)이라 부른다.

<38> NN으로부터 뉴런을 추가하거나 제거하기 위한 알고리즘에 허용차 (allowances)가 없기 때문에, 역전파는 고정된-크기 (fixed-sized)의 NN 상에서 작동해야 한다. 데이터 세트를 학습하기 위해 NN을 훈련시킬 때, 얼마나 많은 층들과 층 각각에 있는 얼마나 많은 뉴런들이 상기 데이터를 학습하는 데 필요한지 추측한다. 훈련 이후, 필요하지 않은 뉴런을 제거하여, 상기 훈련된 NN의 수행 능력을 향상시키기 위한 시도들이 있을 수 있다. 하지만, 훈련 동안, 뉴런의 수는 고정된 채로 있어야만 한다.

<39> 전통적인 역전파 알고리즘은 다음과 같이 요약될 수 있다: (a) 초기화 (initialization): NN에 있는 층의 수 및 각 층에 있는 뉴런의 수를 정의하며, NN 가중치를 랜덤 값 (random value)으로 초기화하고; (b) 훈련 세트로부터의 입력 벡터를 NN에 적용한다. 입력 층 다음의 첫 번째 층에 있는 뉴런 각각에 대해 방정식 1.5를 사용하여 출력을 계산하고, 이 출력을 입력으로서 다음 층에 사용한다. 출력이 계산될 때까지 NN의 층 각각에 대해 이 과정을 반복하고; (c) 방정식 1.6을 사용하여 NN에 존재하는 오차량에 따라 가중치를 변경하고; (d) NN이 훈련된 것으로 간주될 때까지 단계 b)와 c)를 반복한다. 상기 오차가, 훈련 세트에 있는 일부 수의 입력 벡터들에 대한 얼마간의 임의 값 (arbitrary value) 이하로 떨어졌을 때, NN은 훈련된 것으로 간주된다.

<40> 역전파를 사용하여 데이터 세트를 학습하도록 NN을 훈련시키는 것과 연관된 많은 이점이 있지만, 역전파는 그것의 한계점을 갖는다. 역전파를 사용하면, NN은 데이터 세트를 학습하는 데 장시간이 걸리거나, 더 심각하게는, 데이터 세트를 전혀 학습할 수 없다는 것이다. 일부 경우, NN이 데이터 세트를 학습할 수 없는 이유를 규명하는 것이 불가능할 수 있고/있거나, NN이 상기 데이터 세트를 학습할 수 있을지 또는 학습하는 데 장시간이 걸릴지 훈련 동안 구별하기란 불가능하다.

<41> 역전파를 사용하면, NN은 너무 작아 데이터를 학습할 수 없을 것이다. 전통적으로, NN 설계자는 각각의 은닉 층에 몇 개의 뉴런들이 사용되는지와, 데이터 세트를 학습하기 위해 필요한 은닉 층의 수를 추측해야만 한다. 만일 NN이 너무 크다면, 그것은 알맞게 일반화할 수 없을 것이다. 따라서, 이 문제를 해결하기 위해, 뉴런들이 때때로 NN으로부터 제거된다. NN은 오차 공간의 국소 최소치에 고정되어 있을 수 있다. NN이 상기 데이터 세트를 학습할 때, NN은 오차 공간의 대역 최소치에 있는다. 오차 함수의 형태가 알려지지 않기 때문에, 그것은 높은 오차 및 낮은 오차 영역 (area)을 갖는다. 역전파가 오차 함수의 1차 미분을 조사하여 오차를 최소화하기 위해 이동하기 때문에, 그것은 오직 국소 영역만을 조사한다. 은닉 층에 있는 뉴런을 훈련시키는 목적은, 상기 데이터 세트에 있는 서로 다른 특징들을 학습하기 위한 것이다. 하지만, 역전파가 NN을 통해 오차를 역으로 전파할 때, 모든 가중치들이 어느 정도 변경되어, 그 결과 아마도 상기 데이터 세트에 있는 특정한 특징들과 뉴런 각각의 독특한 관련성을 줄일 수 있다. 뉴런이, 동일한 층에 있는 다른 뉴런이 동일한 특징들을 학습하는지 결정할 수 없기 때문에 이것은 가능하다. 이로써, 특정한 데이터 특징을 학습한 가중치가 상기 특징을 잊을 수 있게 된다.

<42> 역전파로 NN을 훈련시키는 데 있어 주요 문제점은, 상기 이유들 중 어느 것이 NN이 데이터 세트를 학습하지 못하도록 하는 원인인지 구별하는 것이 불가능하다는 것이다. 그것은 상기 데이터 세트를 학습하지만 꽤 느릴 수 있거나, 또는 NN이 너무 작아서 상기 데이터 세트를 결코 학습할 수 없거나, 또는 국소 최소치에 고정될 수 있다. 역전파를 이용하는 데 있어 또 다른 중요한 문제점은, NN이 데이터 세트를 학습할 때, NN이 학습한 것이 숫자로서 가중치 및 임계치로 이해 불가능하게 부호화되는 것이다.

<43> 역전파로 NN을 훈련시키는 데 따른 어려움들 때문에, 피드-포워드 NN을 훈련시키기 위한 대체 알고리즘들을 개발하는 데 많은 연구들이 행해지고 있다.

<44> 여러 알고리즘들이 피드-포워드 NN 훈련을 위해 역전파의 대안으로서 개발되고 있다. 현재 두 부류 (class)의 대체 알고리즘들이 있다: (1) NN에 고정된 수의 뉴런 또는 자원 (resources)을 필요로 하는 알고리즘들; 및 (2) 뉴런이 NN에 동적으로 (dynamically) 할당되도록 하는 알고리즘들이다.

<45> 대부분의 이런 알고리즘들은 고정된-크기의 NN을 갖는 데 의존하기 때문에, 그 결과 역전파가 경험한 것과 동일

한 문제들을 겪는다. 알려진 한 방법은 유전자 알고리즘 (genetic algorithm)을 사용하여, 가중치 값을 찾는 것이다. 유전자 알고리즘은 국소 최소치 문제를 해결할 수 있지만, 훈련시키는 데 무한정한 시간이 걸리고, 또한 NN이 너무 작기 때문에 적절히 훈련시킬 수도 없다. 또 다른 대체 방법은, 원형 기준 함수 (Radial Basis Functions: RBF)를 사용하는 것으로, 이것은 오직 단층을 사용하여, NN을 학습하는 것이지만, 역전파가 필요로 하는 것보다, 데이터 세트를 학습하기 위해 더 많은 입력 벡터들을 필요로 한다. 고정된-크기의 NN과 연관된 문제의 결과로서, NN이 상기 데이터 세트를 학습하기 위해 필요한 대로 성장하도록 허용하는 것이 유용하다.

<46> 동적으로 뉴런을 추가하는, 피드-포워드 NN 훈련 알고리즘은 사전-정의된 (pre-defined) 구조 문제들에 대한 해(solution)로 도입되는데, 이는 오직 상기 데이터에 있는 특징들이 확실히 학습될 수 있게 할 필요가 있을 때만 뉴런을 추가하는 데 유연성 (flexibility)을 주기 때문이다. 따라서, 다른 뉴런이 상기 데이터 세트에 있는 특정한 특징들을 학습할 수 없을 때 뉴런이 추가되고, 그 결과 상기 훈련된 NN은, 훈련 동안 어떠한 규칙이 NN에 의해 학습되는지 확인하는 데 더 효과적으로 사용될 수 있다. 사전-정의된 망 구조는 데이터를 학습하는 NN의 능력을 제한한다. NN은 그들의 가중치를 변경하여 학습하는데, 이러한 가중치는 생물학적 NN에서 시냅스 가중치 (synaptic weight)에 해당한다. 앞서 논의한 대로, 피드-포워드 NN은 생물학적 NN에서 영감을 얻었다. 하지만, 생물학적 NN은 필요에 따라, 뉴런에 대한 연결들을 동적으로 만들어낸다.

<47> 현재, 구조적으로 동적인 알고리즘에 두 가지 접근법 (approach)이 있다: (1) NN으로부터 뉴런을 제거하는 접근법. NN으로부터 뉴런을 제거하기 위한 두 가지 접근법은: (i) 오차 최소화 과정에 벌칙 (penalty)을 추가하는, 루멜하트의 가중치 감소 (Rumelhart's Weight Decay)와 같은 훈련 동안 작동하는 접근법; 그리고 (ii) NN으로부터 가중치를 제거한 후 대역 오차에 대한 영향력을 계산하는, 최적의 뇌 수술 (Optimal Brain Surgeon)과 같은 훈련 후에 뉴런을 제거하는 더 흔한 접근법; 및 (2) 캐스케이드-상관 망 (Cascade-Correlation Network)[이하, "CCN"이라 함], 동적 노드 생성 (Dynamic Node Creation)[이하, "DNC"라 함], 감수분열 (Meiosis), 및 예를 들어 제한된 쿨롱 에너지 분류기 (Restricted Coulomb Energy Classifiers)[이하, "RCEC"라 함] 및 다항식-시간-훈련된 초구 분류기 (Polynomial-Time-Trained Hyperspherical Classifiers)[이하, "PTTHCs"라 함]과 같은 초구 분류기 부류와 같은, NN에 뉴런을 추가하는 접근법.

<48> 비록 훈련 동안 NN에 동적으로 뉴런을 할당하여 작동하는 NN 훈련 알고리즘을 제공하려는 여러 시도들이 있었지만, 어느 것도 다양한 상황에서 데이터를 효과적으로 및/또는 정확하게 분류하는 데 이상적이지 않은 것으로 여겨진다.

<49> NN이 과학 및/또는 산업의 관심 대상이 된 주된 이유는, 데이터 내의 관계를 찾는 그들의 능력 때문인데, 이로써 상기 데이터가 분류되어, 입력 벡터, 또는 패턴 (pattern)을 성공적으로 분류할 수 있고, NN이 훈련 동안 노출되지 않는다. 이러한 강력한 특성은 종종 NN의 "일반화" 능력이라고 불린다. NN이 훈련 동안 노출되지 않았던 입력 벡터는 흔히 보이지 않는 (unseen) 패턴 또는 보이지 않는 입력 벡터라고 불린다. NN이 일반화할 수 있기 위해서는 훈련이 필요하다.

<50> 훈련 동안, NN은 그것을 훈련시키는 데이터 세트에 있는 두드러진 특징들을 학습하고, 보이지 않는 입력 벡터의 출력을 '예측'할 수 있다. NN이 분류할 수 있는 것은 상기 NN이 무엇으로 훈련되는가에 따라 다르다.

<51> NN이 데이터에 있는 노이즈 (noise)를 다루도록 하는 것을 일반화하는 것이 NN의 능력이다.

<52> 뛰어난 일반화를 확보하기 위해, NN에서 훈련될 가중치 수보다 더 많은 훈련 입력 벡터가 이용될 수 있어야만 한다고 여겨진다.

<53> NN은, 그것이 학습한 높은 비율 (high ratio)의 입력 벡터 및 테스트 세트를 성공적으로 분류할 수 있을 때 훈련된 것으로 간주된다. 하지만, NN을 훈련시키고 테스트하는 데 이용가능한 분류된 데이터 패턴의 수는 한정적일 수 있기 때문에, 상기 데이터 세트를 분리하는 방법이 고려되어야만 한다. NN이 얼마나 잘 훈련되어 테스트될 수 있는지 결정하기 위해, 데이터 세트를 분리하는 방법에 관한 접근법이 많이 있다.

<54> NN이 훈련됐는지 결정하는 일반적인 방법은, 역전파로 훈련된 NN을 사용할 때 각각의 입력 벡터에 있는 오차 수를 계산하는 것이다. 당업자는, 이전에 NN에 있는 오차를 확인하기 위해 사용되었던 상기 접근법들을 잘 이해할 것이고, 따라서 그에 대한 자세한 논의는 본 명세서에 제공되지 않을 것이다.

<55> 하지만, 이제 훈련 알고리즘들 사이의 비교 근거로 사용될 수 있는 속성들 (attributes)을 논의할 것이다.

<56> 학습 알고리즘을 비교할 때 고려될 수 있는 많은 요인들이 있어, 수행 능력의 객관적 척도 (objective measure)가 된다.

<57> 일반적으로, 비교 시에, 학습 알고리즘의 다음 4가지 속성들이 고려된다: (1) 정확성: 이것은 훈련 동안 학습된 규칙의 신뢰도이다; (2) 속도: 이것은 입력 벡터가 분류되는 데 걸린 시간의 척도이다; (3) 학습 시간: 이것은 입력 벡터를 학습하는 데 걸린 시간의 척도이다; 및 (4) 설명력 (comprehensibility): 이것은 학습된 규칙을 해석할 수 있는 능력으로서, 그 규칙은 대체 방법에 적용될 수 있다. 이러한 전략은 정량화하기 어렵다.

<58> 이러한 속성들 중 2가지, 즉 데이터 세트를 학습하는 데 필요한 학습 알고리즘의 시간 및 NN에 의해 학습된 것의 설명력을 더 조사할 것이다.

<59> 앞서 논의한 대로, NN이 데이터 세트를 결코 학습할 수 없을 수도 있기 때문에, 역전파로 데이터 세트를 학습하도록 NN을 훈련시키는 것은, 장시간이 필요할 수 있다. 고정된-크기의 NN을 훈련시키는 데 걸리는 시간은 기하급수적일 수 있다고 여겨진다. 이러한 이유로, NN을 훈련시키는 데 걸리는 시간은 대체 훈련 알고리즘들 사이의 비교 기준이 된다. 이상적인 훈련 알고리즘은, 훈련 입력 벡터에 최소한의 노출을 필요로 할 것이다. 최적의 상황에서 훈련 입력 벡터에 대한 가능한 최소한의 노출은, NN을 완전히 훈련됐을 때 오직 한 번 각각의 입력 벡터에 노출시키는 것이다. 그러한 훈련 알고리즘은 단일 패스 (single pass) 훈련 알고리즘이라고 불릴 수 있다.

<60> 피드-포워드 NN을 훈련시키는 알고리즘들 사이의 비교를 위한 기초 (basis)로서 흔히 사용되는 4가지 속성들 가운데 설명력은, 특히 숫자 값으로 훈련된 피드-포워드 NN에 대해 정량가능성 (quantifiable)이 가장 적은데, 이는 훈련 동안 NN에 의해 학습된 규칙이 숫자 값으로 이해 불가능하게 부호화되기 때문이다. 훈련 동안 학습된 규칙을 추출 (extraction)할 수 있는 방법 중 하나는 민감도 분석 (sensitivity analysis)을 수행하는 것이다. 민감도 분석은 오차에 대한 견고성 (robustness)의 척도라고 불릴 수 있다.

<61> 규칙 추출은, 사용자가 이 시스템에 의해 생산된 결과를 신뢰할 수 있기 때문에 관심의 대상이었고, 이것은, 의료 수술, 항공 교통 관제 및 원자력발전소 감시와 같은 중요한 문제 영역에 NN이 사용될 때, 또는 천문학적 데이터의 경우와 같은, NN을 훈련시켜 수집된 데이터로부터 이론을 추론할 때 매우 중요하다.

<62> 설명력을 보장하기 위한 바람직한 규칙은 입력을 함께 관련시키는 명제 논리 규칙의 형태에 있다.

<63> 민감도 분석은, NN 내에 저장된 정보가 무엇인지 찾는 하나의 방법이기 때문에, 종종 NN에 수행된다. 이로써, NN에 의해 학습된 규칙이 무엇인지 찾는 것이 종종 바람직하기 때문에, 상기 규칙이 숫자 값으로 종종 이해 불가능하게 부호화됨에 따라, 민감도 분석 수행이 NN에 매우 귀중한 것이 된다.

<64> NN에 민감도 분석을 수행하는 데 이용될 수 있는 두 가지 접근법이 있다: (1) 가중치를 변경하는 효과 (effect); 및 (2) NN에 노이즈 입력을 적용하는 효과.

<65> 만일 입력 공간이 잘 알려지면, 필요한 만큼 많은 데이터 포인트들 (data points)을 생성시켜, 다음 3가지 방법에 의해 선택된 입력 벡터에 대한 NN의 출력을 찾는 것이 가능하다: (1) 데이터 공간에 있는 모든 포인트들에 대한 출력 찾기. 만일 NN이 이진 (binary) 데이터로 훈련된다면, 상기 데이터 세트는 반드시 한정된다; (2) 상기 입력 공간으로부터 데이터 포인트들을 무작위로 선택; 또는 (3) 상기 입력 공간에 있는 모든 n^{th} 데이터 포인트 (여기서, $n>1$) 선택. 이로써 상기 입력 공간에 대해 고른 분배가 이루어진다.

<66> 또한, 데이터 포인트는 입력 공간의 영역으로부터 선택될 수 있는데, 이 입력 공간은 어떤 원하는 NN 반응이 일어날지 알려지지 않은 곳이다. 이 경우, 미지의 (unknown) 데이터가 주어졌을 때 NN이 어떻게 반응할지를 보여줄 것이다.

<67> 지금껏, 상기 입력-공간을 검사하는 방법이 조사되었고, 이제, NN에 있는 뉴런의 가중치-공간을 조사할 것이다.

<68> 시스템은, 지정한 대로 수행하는 데 필요한 수많은 구성요소들을 갖고 있으며, 이 구성요소들을 통해 상기 시스템이 필요한 대로 수행된다. 각각의 구성요소가 지정한 대로 수행할 때, 상기 구성요소들은 그들의 최적의 범위에 있다고 여겨진다.

<69> 민감도 분석은, 상기 시스템의 구성요소들에 대한 최적의 값 또는 범위로부터 벗어나는 효과의 조사이다. 이 경우, 상기 최적의 범위는 훈련된 NN에 있는 가중치에 대한 것이다. 상한 및 하한 (upper and lower limit)이 설정되어, 이러한 경우 가중치가 NN의 작동을 변화시키지 않고 변경될 수 범위 (또는 간격)를 찾는다. 민감도 분석을 수행하기 위해, 상기 시스템에 있는 각각의 구성 요소는 차례대로 테스트되고, 모든 다른 구성 요소들은 고정된 상태에 있는다. 테스트되는 구성 요소는 모든 가능한 값으로 설정되어, 상기 시스템이 수행하는 방식을 결정할 것이다. 이 과정 동안, 상한 및/또는 하한이, 상기 시스템이 최적으로 작동하도록 하는 구성 요소를 확인하고, 상기 구성 요소가 이러한 범위를 벗어날 때, 상기 시스템이 어떻게 작동하는지 관찰될 수 있다. 이 과

정을 레인징 (ranging)이라고 부른다. 상기 상한 및 하한이 제약 (constraint)으로 표현될 수 있다.

<70> 알려진 민감도 분석은, NN이 학습한 것을 이해 가능하게 만들 입력 변수들을 함께 연관시키는 명제 논리 규칙을 생성하지 않는 것으로 여겨진다.

<71> 민감도 분석의 목적은, 민감도 분석이 구성 요소의 작동을 정확히 정의하기 때문에, 체적 (volume)의 형태를 결정할 수 있는 것이다. 하지만, 알려진 NN 훈련 방법의 한계로 인해, 뉴런을 활성화시키는 체적 면 (surface)을 찾는 것은 불가능하다. 유일한 방법으로는, 통계적 방법을 사용하여 각각의 가중치 범위를 결정하여, 상기 면을 조사하는 것이 가능하였다. 상기 체적의 실제 면이 가중치들 사이에 존재하는 관계를 정의하기 때문에 상기 체적의 실제 면을 아는 것이 이상적일 것이고, 필요시, 이것으로부터 가중치의 범위가 결정될 수 있다.

<72> 훈련 동안 피드-포워드 NN이 학습한 것을 결정할 수 있는 것이 매우 바람직하고, 결과적으로, 어떠한 관계가 데이터 내에 존재하고 NN에 의해 학습되는지 확인하기 위해 많은 연구가 수행되었다. 이것이 설명력이라 불리고, 학습 알고리즘이 얼마나 유용한지 결정하는 데 기여한 하나의 속성인 것이다. 현재 NN으로부터 규칙을 추출하기 위해 사용된 방법은 훈련이 완료된 후에 수행된다.

<73> 찾을 필요가 있는 바람직한 관계의 유형은 명제 논리로서 주어진다. 이러한 요건 (requirement)은 다음으로 요약될 수 있다: 상기 훈련 조건을 만족시키는 모든 수치 해 (numeric solution)를 정의하여, 따라서 민감도 분석이 NN에서 쉽게 수행되도록 하는 요건; 및 (b) 데이터 세트를 분류하기 위해 훈련 동안 NN에 의해 학습된 규칙이 상기 NN으로부터 쉽게 읽히도록 할 요건.

<74> 다양한 동적 알고리즘에 관하여 상기 언급된 알려진 훈련 알고리즘 가운데, 규칙이 NN으로부터 직접적으로 거의 읽히도록 하는 유일한 알고리즘은 초구 분류기로, 그것은 영역들 사이의 OR 관계를 형성한다. 따라서, 입력 공간의 영역들이 특정 범주에 속하거나 속하지 않기 때문에, 영역들이 AND로 결합될 수 없다. 만일 그들이 상기 영역에 속하지 않는다면, 활성화되지 않아야만 하는 뉴런의 활성화를 억제하기 위해 구 (sphere)가 추가되어, 따라서 OR이 상기 입력 공간을 표현하는 데 적절하다. 초구를 정의하는 반지름 (radius)은, 입력 공간이 복잡해짐에 따라 0이 되기 쉽고, 궁극적으로 초구가 각각의 입력 벡터에 추가된다. 비록 은닉 층에 있는 뉴런에 의해 정의된 영역들이 상기 입력 공간에 있는 영역들에 가까워지지만, 데이터 포인트들만큼 많은 초구들이 있는 최악의 경우를 제외하고는, 상기 영역들은 그것을 정의하지 않는다. PTTHC들은 입력 공간의 적용 범위를 증가시키고자 하며, 그 결과 계산 복잡성 (computational complexity)을 훨씬 늦춰, 일반화 수행 능력을 향상시킨다.

<75> CCN, 감수분열 및 DNC 모두 가중치를 숫자로 훈련하기 때문에, 훈련 동안 상기 데이터 내에 어떠한 관계들이 찾아지는지 결정하는 것은 쉽지 않다.

<76> 이러한 모든 알고리즘은, 일반화에 관한 수행 성공 정도를 변화시키며, NN에 뉴런을 동적으로 할당한다. 일부 알고리즘은 다른 것들보다 일부 데이터 세트에서 우수하고, 초구 분류기를 제외한 모든 것이 가중치-공간의 경계 조건 (boundary condition) 정보를 손실시켜, 규칙 추출에 그다지 유용하지 못한다.

<77> 일부 알고리즘들은, 심지어 역전파보다 더 느리게 되는 경향이 있는 어닐링 (annealing)을 기초로 한 감수분열 알고리즘과 같은 다른 것들보다, 일부 데이터 세트를 더 빨리 학습한다.

<78> CCN 및 DNC는 특정 데이터 세트에 대한 빠른 훈련 시간을 갖는 것으로 보고되지만, 이들은, 뉴런이 NN에 추가되기 전에 상기 시스템에서 오차 양을 줄이기 위한 반복에 의존하기 때문에, 단일 패스 알고리즘이 아니다.

<79> 하지만, 필요한 대로 NN에 뉴런을 추가하는 단일 패스로 학습하여, 규칙이 상기 NN으로부터 직접적으로 읽히도록 하는 NN 훈련 알고리즘은 없다.

<80> 따라서, 본 발명의 목적은, 뉴런이 데이터 세트를 학습하는 데 필요한 대로 NN에 할당될 수 있다는 의미에서, 상관적 (relational)이고 동적인 NN 훈련 방법을 제공하는 것이다.

<81> 본 발명의 또 다른 목적은, 단일 패스로 데이터 세트를 학습할 수 있는 NN 훈련 방법을 제공하는 것이다.

<82> 본 발명의 또 다른 목적은, 규칙들이 NN으로부터 직접적으로 읽히도록 하는 NN 훈련 방법을 제공하는 것이다.

발명의 상세한 설명

<83> **발명의 요약**

<84> 본 발명의 일 양태에 따라, 인공 NN 훈련 방법이 제공되는데, 상기 방법은 다음 단계들을 포함하다: (i) 훈련될 NN의 출력을 선택하고, 선택된 출력에 대한 NN의 입력 층에 있는 입력 뉴런에 NN의 출력 뉴런을 연결하여 NN을

초기화하는 단계; (ii) NN에 의해 학습될 데이터 세트를 준비하는 단계; 및 (iii) 준비된 데이터 세트의 입력 벡터를 NN의 첫 번째 은닉 층, 또는 만일 NN이 은닉 층을 갖고 있지 않다면 NN의 출력 층에 적용하여 학습될 NN에 준비된 데이터 세트를 적용하고, NN의 각 층에 있는 선택된 출력의 적어도 하나의 뉴런이 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 있는지 판단하는 단계, 여기서: 만일 NN의 각 층에 있는 선택된 출력에 대한 적어도 하나의 뉴런이 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 있고, 만일 학습할 준비된 데이터 세트의 입력 벡터가 더 있다면, 다음 입력 벡터를 위해 단계 (iii)를 반복하고, 만일 훈련될 출력이 더 있다면 NN의 다음 출력을 위해 단계 (i) 내지 (iii)을 반복하고; 만일 NN의 선택된 출력에 대한 은닉 층에 있는 어떠한 뉴런도 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 없다면, 새로운 뉴런이 그 층에 추가되어, 선택된 출력에 대한 그 층에 있는 다른 뉴런에 의해 학습될 수 없는 연관 출력을 학습하고, 만일 학습할 데이터 세트의 입력 벡터가 더 있다면 다음 입력 벡터를 위해 단계 (iii)을 반복하고, 만일 훈련될 출력이 더 있다면 NN의 다음 출력을 위해 단계 (i) 내지 (iii)을 반복하고; 만일 NN의 선택된 출력의 출력 뉴런이 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 없다면, 출력 뉴런이 NN의 은닉 층 뉴런이 되고, 새로운 뉴런이 이 은닉 층에 추가되어 출력 뉴런에 의해 학습될 수 없는 연관 출력을 학습하고, 새로운 출력 뉴런이 선택된 출력에 대한 NN에 추가되고, 만일 학습할 데이터 세트의 입력 벡터가 더 있다면, 다음 입력 벡터를 위해 단계 (iii)를 반복하고, 만일 훈련될 출력이 더 있다면 NN의 다음 출력을 위해 단계 (i) 내지 (iii)을 반복한다.

<85> 본 발명의 다른 양태에 따라, 인공 NN 훈련 방법이 제공되는데, 상기 방법은 다음 단계들을 포함한다: (i) NN에 의해 학습될 데이터 세트 준비; (ii) 훈련될 NN의 출력을 선택하고, 선택된 출력에 대한 NN의 입력 층에 있는 입력 뉴런에 NN의 출력 뉴런을 연결시켜 NN을 초기화하는 단계; 및 (iii) 준비된 데이터 세트의 입력 벡터를 NN의 첫 번째 은닉 층, 또는 만일 NN이 은닉 층을 갖고 있지 않다면 NN의 출력 층에 적용하여 학습될 NN에 준비된 데이터 세트를 적용하고, NN의 각 층에 있는 선택된 출력에 대한 적어도 하나의 뉴런이 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 있는지 판단하는 단계, 여기서: 만일 NN의 각 층에 있는 선택된 출력에 대한 적어도 하나의 뉴런이 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 있고, 만일 학습할 준비된 데이터 세트의 입력 벡터가 더 있다면, 다음 입력 벡터를 위해 단계 (iii)를 반복하고, 만일 훈련될 출력이 더 있다면 NN의 다음 출력을 위해 단계 (ii) 내지 (iii)을 반복하고; 만일 NN의 선택된 출력에 대한 은닉 층에 있는 어떠한 뉴런도 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 없다면, 새로운 뉴런이 그 층에 추가되어, 그 층에 있는 다른 뉴런에 의해 학습될 수 없는 연관 출력을 학습하고, 만일 학습할 데이터 세트의 입력 벡터가 더 있다면 다음 입력 벡터를 위해 단계 (iii)을 반복하고, 만일 훈련될 출력이 더 있다면 NN의 다음 출력을 위해 단계 (ii) 내지 (iii)을 반복하고; 만일 NN의 선택된 출력에 대한 출력 뉴런이 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 없다면, 출력 뉴런이 NN의 은닉 층 뉴런이 되고, 새로운 뉴런이 이 은닉 층에 추가되어 출력 뉴런에 의해 학습될 수 없는 연관 출력을 학습하고, 새로운 출력 뉴런이 선택된 출력에 대한 NN에 추가되고, 만일 학습할 데이터 세트의 입력 벡터가 더 있다면 다음 입력 벡터를 위해 단계 (iii)를 반복하고, 만일 훈련될 출력이 더 있다면 NN의 다음 출력을 위해 단계 (ii) 내지 (iii)을 반복한다.

<86> 상기 정의된 NN 훈련 방법의 실제적인 바람직한 실시예에서, NN의 뉴런은 선형 임계 게이트 (LTG)이다.

<87> 준비된 데이터 세트를 학습될 NN에 적용하는 상기 단계 (iii)에서, LTG가 입력 벡터에 대한 연관 출력을 생산하는 것을 학습할 수 있는지 결정하는 것은, LTG의 임계치와 가중치 사이의 관계가 상기 LTG가 이전에 학습한 것이 주어진 해인지를 결정하는 것이 바람직하다. 실제적인 바람직한 실시예에서, 상기 관계는 제약 (constraint)이며, 여기서 입력 벡터와 LTG의 가중치 벡터가 NN의 선택된 출력을 기초로 한 LTG의 임계치와 관계를 형성한다. 상기 바람직한 실시예에서, 제약을 학습하는 것은 상기 제약을 LTG의 제약 세트에 추가할 수 있는 것이다. 제약을 LTG의 제약 세트에 추가할 수 있기 위해, 모든 제약들 사이에 해가 있어야만 한다.

<88> NN을 초기화하는 단계는, 출력 LTG의 제약 세트를 제거하여, 출력 LTG의 제약 세트가 비워지는 단계를 포함하는 것이 바람직하다.

<89> NN에 의해 학습될 데이터 세트를 제조하는 단계가 적어도 다음 단계들을 포함하는 것이 바람직한데, 이들 각각의 단계는 임의의 순서대로 수행될 수 있다: 훈련 동안 데이터 세트가 NN에 제공되기 전에 데이터 세트를 사전-정의된 데이터 포맷 (format)으로 변환시키는 단계; 훈련 동안 데이터 세트가 NN에 제공되기 전에 상기 데이터 세트에 불일치 (inconsistencies)가 있는지 결정하는 단계; 훈련 동안 데이터 세트가 NN에 제공되기 전에 데이터 세트를 분류하는 단계; 및 훈련 동안 데이터 세트가 NN에 제공되기 전에 0 입력 벡터가 데이터 세트에 이용가능한지 결정하고, 만일 상기 0 입력 벡터가 상기 데이터 세트에 이용가능하다면, 상기 데이터 세트가 정렬되어, 상기 0 입력 벡터가 처음에 훈련될 NN에 제공되는 단계. 훈련 동안 데이터 세트가 NN에 제공되기 전에 데이터 세트를 사전-정의된 데이터 포맷으로 변환시키는 상기 단계의 바람직한 실시예에서, 상기 사전-정의된 포맷

은 이진 (binary) 또는 부동-소수점 (floating-point) 데이터 포맷이다. 데이터 세트가 NN에 제공되기 전에 상기 데이터 세트에 불일치가 있는지 결정하는 상기 단계는, 다른 출력을 생산하는 두 개 이상의 동일한 입력 벡터가 있는지 결정하는 단계를 포함하는 것이 바람직하다. 데이터 세트에 불일치가 있는지 결정하는 상기 단계의 바람직한 실시예에서, 만일 두 개 이상의 동일한 입력 벡터가 다른 출력을 생산하는 것으로 결정된다면, 상기 입력 벡터 중 단 하나만 사용된다. 훈련 동안 데이터 세트가 NN에 제공되기 전에 데이터 세트를 분류하는 상기 단계는 다음 단계들을 포함하는 것이 바람직하다: 데이터 세트의 입력 벡터를 두 개의 세트로 분류하고, 그러한 출력에 대해 0을 생산하는 입력 벡터들로부터 1을 출력하는 입력 벡터들을 분리하고, 먼저 훈련될 두 개의 세트 중 하나를 선택하는 단계; 자기 조직화 맵 (SOM)으로 데이터를 분류하는 단계; 다른 적합한 방법을 사용하여 데이터를 분류하는 단계. 또한, 훈련 동안 데이터가 NN에 제공되기 전에 각각의 입력 층에 대한 단일 리스트 (list)가 상기 분류된 데이터로부터 만들어지는 것이 바람직하다.

<90> 실제적인 바람직한 실시예에서, 새로운 LTG가 층에 추가되어, 단계 (iii)에 따라 그 층에 있는 다른 LTG에 의해 학습될 수 없는 제약을 학습할 때: 새로운 LTG가, NN의 선택된 출력에 기여하는 다음 층에 있는 모든 LTG들에 연결되고, 새로운 LTG로부터 입력을 받는 다음 층에 있는 LTG의 제약 세트가 갱신 (update)되어 새로운 LTG로부터 입력을 수용하고; 만일 새로운 LTG를 지닌 층이 NN의 첫 번째 층이 아니라면, 새로운 LTG가 NN의 선택된 출력에 기여하는 이전 층 (preceding layer)에 있는 모든 LTG들에 연결되어 그것들로부터 입력을 받고; 새로운 LTG의 제약 세트가 갱신되어 그 층에 있는 이전의 마지막 LTG의 변경된 제약 세트 및 그 층에 있는 다른 LTG에 의해 학습될 수 없는 제약의 사본 (copy)을 포함한다.

<91> 실제적인 바람직한 실시예에서, 새로운 출력 LTG가 단계 (iii)에 따라 NN에 추가될 때: 새로운 LTG가, 은닉 층에 있는 LTG에 연결되어 그것으로부터 입력을 받고; 만일 은닉 층이 NN의 첫 번째 층이 아니라면, 은닉 층에 있는 새로운 LTG가 NN의 선택된 출력에 기여하는 이전 층에 있는 모든 LTG들에 연결되어 그것들로부터 입력을 받고; 은닉 층에 추가된 새로운 LTG의 제약 세트가 갱신되어 그 층에 있는 이전의 출력 LTG의 변경된 제약 세트 및 이전의 출력 LTG에 의해 학습될 수 없는 제약의 사본을 포함하고; 새로운 출력 LTG가, 이전의 출력 LTG에 의해 학습될 수 없는 것에 따라 사전-정의된 논리 관계 (logical relationship)로 그것의 입력을 결합한다. 새로운 출력 LTG가 단계 (iii)에 따라 NN에 추가될 때, 이 새로운 출력 LTG로의 입력들 사이에 형성된 사전-정의된 논리적 관계가 논리 OR, 논리 AND, 또는 다른 적당한 논리 관계이다. 이러한 바람직한 실시예에서, 만일 이전의 출력 LTG에 의해 학습될 수 없는 입력 벡터가 출력 1을 생산한다면 논리 OR이 사용되고, 만일 이전의 출력 LTG에 의해 학습될 수 없는 입력 벡터가 출력 0을 생산한다면 논리 AND가 사용된다.

<92> 본 발명의 또 다른 양태에 따라, 훈련 동안 NN 층에 새로운 뉴런을 추가하는 방법이 제공되는데, 여기서 선택된 출력에 대한 그 층의 다른 뉴런이 학습되는 데이터 세트의 입력 벡터와 연관된 관계를 학습할 수 없을 때, 새로운 뉴런이 NN에 추가되며, 상기 방법은 다음 단계들을 포함한다: 그 층에 있는 NN의 선택된 출력에 기여하는 이전의 마지막 뉴런으로부터 변경된 모든 데이터 및 그 층에 있는 다른 뉴런에 의해 학습될 수 없는 관계의 사본으로 새로운 뉴런을 갱신하는 단계; 및 출력 뉴런을 갱신하여, 새로운 뉴런으로부터 입력을 수용하는 단계.

<93> 실제적인 바람직한 실시예에서, NN의 뉴런은 LTG이다.

<94> 상기 관계가 LTG의 임계치와 가중치 사이의 관계인 것이 바람직하다. 실제적인 바람직한 실시예에서, 상기 관계는 제약이고, 여기서 데이터 세트의 입력 벡터와 LTG의 가중치 벡터는 NN의 출력을 기초로 한 LTG의 임계치와 관계를 형성한다.

<95> 본 발명의 또 다른 양태에 따라, 이진 포맷 (binary format) 데이터 세트 이외의, 데이터 세트를 NN에 의해 학습될 이진 포맷 데이터 세트로 변환시키는 방법이 제공되는데, 상기 방법은 다음 단계들을 포함한다: (i) 데이터 세트의 속성 각각을 이진수로 나타내기 위한 비트 (bit)의 수를 개별적으로 결정하는 단계; (ii) 방정식 [범위 = (최대값 - 최소값) + 1]을 사용하여 데이터 세트의 속성 범위를 계산하는 단계; 및 단계 (i)에서 결정된 비트 수를 사용하여 데이터 세트의 속성 범위를 이진수로 부호화하는 단계.

<96> 데이터 세트를 이진 포맷 데이터 세트로 변환시키는 방법은, 상기 정의된 NN 훈련 방법의 데이터 준비 단계 [단계 (ii) 또는 단계 (i)]에 따라 사용되는 것이 바람직하다.

<97> 본 발명의 또 다른 양태에 따라, NN에 의해 훈련될 데이터 세트를 분류하는 방법이 제공되는데, 상기 방법은 다음 단계들을 포함한다: 데이터 세트의 입력 벡터를 2개의 그룹으로 분류하는 단계; 0을 출력하는 입력 벡터들로부터 1을 출력하는 입력 벡터들을 분리하는 단계; 및 NN에 의해 먼저 학습될 2개의 그룹 중 하나를 선택하는 단계.

<98> 본 발명의 또 다른 양태에 따라, 입력 벡터가 뉴런에 의해 드러나는지 (known) 또는 드러나지 않는지 (unknown) 결정하는 방법이 제공되는데, 상기 방법은 다음 단계들을 포함한다: 입력 벡터로부터 제약 및 그것의 보수 (complement)를 구성하는 단계; 뉴런의 제약 세트에 제약 및 그것의 보수를 교대로 추가하는 단계; 만일 상기 두 경우에 해가 있는지 결정하기 위해 제약 세트를 테스트하는 단계, 여기서: 만일 제약 또는 그것의 보수에 대한 해가 없다면, 입력 벡터가 뉴런에 의해 드러난 것으로 결정되고; 만일 상기 제약 및 그것의 보수가 제약 세트에 교대로 추가될 때 해가 있다면, 입력 벡터가 뉴런에 의해 드러나지 않은 것으로 결정된다.

<99> 상기 제약 세트는 LTG 뉴런으로 구성된 NN 뉴런의 제약 세트인 것이 바람직하다. 또한, 상기 방법은 상기 정의된 NN 훈련 방법에 따라 훈련된 NN의 보이지 않는 입력 벡터의 출력을 결정하는 데 사용되는 것이 바람직하다. 실제적인 바람직한 실시예에서, 상기 방법은 본 발명의 NN 훈련 방법에 따라 훈련된 NN의 LTG의 보이지 않는 입력 벡터를 결정하는 데 사용되어, 보이지 않는 입력 벡터에 대한 디폴트 (default) 출력은 데이터 세트에 따라 1 또는 0이다.

<100> 본 발명의 또 다른 양태에 따라, 제약 세트의 최소 활성화 체적 (minimum activation volume: MAV)을 결정하는 방법이 제공되는데, 상기 방법은 다음 단계들을 포함한다: (i) 한 번에 하나씩 상기 제약 세트로부터 각각의 제약을 제거하며, 상기 제약 세트에 있는 나머지 제약들을 변하지 않게 두는 단계; (ii) 상기 제거된 제약의 보수를 상기 제약 세트에 추가하는 단계; (iii) 해가 있는지 보기 위해 새로운 제약 세트를 테스트하는 단계, 여기서: 만일 해가 있다면, 제약 세트로부터 제거된 원래의 제약이 MAV를 정의하는 제약 세트에 추가되고, 제약 세트에 추가된 제약의 보수가 제거되고 원래의 제약이 제약 세트로 되돌아가고, 만일 제약 세트에 테스트할 제약이 더 있다면 단계 (i) 내지 (iii)을 반복하고, 그렇지 않다면 MAV는 상기 MAV를 정의하는 제약 세트 내에 포함된 제약들의 세트이고; 만일 해가 없다면, 제약 세트에 추가된 세트의 보수는 제거되고 원래의 제약이 상기 제약 세트로 되돌아가고, 만일 제약 세트에 테스트할 제약이 더 있다면, 단계 (i) 내지 (iii)를 반복하고, 그렇지 않으면 MAV는 상기 MAV를 정의하는 제약 세트 내에 포함된 제약들의 세트이다.

<101> 상기 제약 세트는 LTG 뉴런으로 구성된 NN 뉴런의 제약 세트인 것이 바람직하다. 실제적인 바람직한 실시예에서, 상기 방법은 상기 정의된 NN 훈련 방법에 따라 훈련된 NN에 있는 각각의 LTG에 대한 MAV를 결정하는 데 사용된다.

<102> **발명의 이점**

<103> 일 양태에서 본 발명은 뉴런을 훈련시키는 데 신규한 접근법을 제공한다. 이러한 접근법은 뉴런으로의 입력 연결과 출력 사이의 관계를 정의하여, 규칙 추출을 단순하게 만들다. 본 발명에 따라 뉴런을 훈련시키는 방법은 일반화를 허용하고, 학습된 데이터가 종래 방법으로 훈련된 뉴런으로 회상 (recall)되도록 한다. 또한, 간단한 테스트를 사용하여, 뉴런이 입력 벡터를 학습할 수 있는지 없는지를 결정한다. 이러한 테스트는, 하나 이상의 뉴런을 NN에 추가하여 데이터 세트에 있는 특징을 학습하기 위한 자연적 기준 (natural criterion)을 형성한다. 뉴런이 은닉 층에 할당될 수 있거나 또는 새로운 출력 층이 데이터 세트의 복잡성에 따라 추가될 수 있다.

<104> 따라서, 본 발명의 NN 훈련 방법은 동적 상관적 (Dynamical Relational)[이하, "DR"이라 함] 훈련 방법이라고 부를 수 있다.

<105> 본 발명의 DR 훈련 방법에 따라 훈련된 NN은, 입력 벡터가 학습될 수 있고, 뉴런이 오직 필요시에 NN에 동적으로 할당될 수 있는지 결정하기 위해 테스트될 수 있기 때문에, 데이터가 단일 패스로 NN에 제공됨에 따라 학습될 수 있다.

<106> 피드-포워드 NN을 훈련시키는 종래의 접근법은 고정된-크기의 NN을 필요로 하고, 뉴런의 은닉 층 수 및 NN이 데이터 세트를 학습하는 데 필요한 은닉 층 각각의 뉴런 수를 추측할 필요가 있다. NN의 크기를 추측하는 것은 중요한 문제인데, 이는 만일 NN이 너무 작다면 데이터 세트를 학습할 수 없을 것이고, 만일 NN이 너무 크다면 NN의 수행 능력을 떨어트릴 수 있기 때문이다. NN의 크기 추측 문제에 대한 최고의 해결책은, 필요한 대로, 오직 필요시에 NN에 뉴런을 동적으로 할당할 훈련 방법을 사용하는 것이다. 따라서, 뉴런을 NN에 동적으로 할당하여, 고정된-크기의 NN과 연관된 문제들을 해결한다. 본 발명의 DR 훈련 방법에 따라, 오직 NN이 입력 벡터를 학습할 수 없는 경우에만, 뉴런이 NN에 할당될 수 있다. 새로운 뉴런이 NN에 할당될 때, 그것은 NN에 이미 있는 뉴런과의 명제 논리 관계를 형성하여, 그 결과 본 발명의 훈련 방법이 상관적이다.

<107> 각각의 입력 벡터는, NN에 제공됨에 따라 학습된다. 이는, 본 발명의 DR 훈련 방법이 단일 패스로 데이터를 학습할 수 있다는 것을 의미한다.

<108> 본 발명에 다른 뉴런 훈련 방법은 데이터 세트에 관한 정보가 학습되도록 하고, 뉴런이 NN에 추가되도록 하는 입력 벡터를 확인할 뿐만 아니라, 입력 벡터가 데이터 세트 분류에 필수적인 것을 보여준다. 또한, 본 발명의 방법은 입력 공간에 뚜렷한 경계를 제공하면서, 피드-포워드 NN을 훈련시키는 다른 알고리즘의 이점들 중 전부는 아니더라도 대부분을 제공한다.

<109> 또 다른 양태에서 본 발명은 훈련 동안 데이터가 NN에 제공되기 전에 데이터를 적당한 포맷으로 변환시키는 방법을 제공한다. 훈련 전에 데이터를 변환시키는 것과 관련된 이점들이 많다. 그 중 하나의 이점은, 데이터 변환이 훈련 동안 NN에 제공된 입력 수를 최소화하면서, 데이터를 정확하게 부호화한다는 것이다. 본 발명에 따른 DR 훈련 방법의 경우, NN에 제공된 입력 수의 최소화가 훈련 시간을 단축하고, 매번 입력 벡터가 NN에 의해 학습된다면, 입력 벡터가 NN에 의해 학습될 수 있는지를 결정하기 위해 제약이 테스트되어야만 한다.

<110> 또 다른 양태에서 본 발명은 훈련 동안 데이터가 NN에 제공되기 전에 데이터를 분류하는 방법을 제공한다. 훈련 전에 데이터를 사전에 분류하는 것이 데이터 분류의 효율성을 증가시킨다. 사전-분류는, 훈련된 NN이 불완전하게 수행되는 상황에서 사용되는 것이 바람직하다. 사전-분류는, NN에 의해 학습될 데이터 세트가 상당히 복잡하여 NN에 뉴런을 추가할 필요가 있을 때마다 추천된다.

<111> 훈련 전에 데이터를 적당한 포맷으로 변환시키는 방법과 훈련 동안 데이터가 NN에 제공되기 전에 데이터를 분류하는 방법은 모든 NN 훈련 방법에 유용한 것으로 여겨진다. 따라서, 본 발명의 이러한 양태는 독립적이고, 본 발명의 DR 훈련 방법에 한정되지 않는다.

<112> 테스트 동안, 중요한 이점은, NN에 의해 학습된 데이터가 100% 정확하게 회상될 수 있다는 것이다. 또 다른 양태에서 본 발명은, NN이 보이지 않는 입력 벡터에 대한 출력을 알고, 드러나지 않은 입력 벡터를 분명히 확인할 수 있는지 결정하기 위해 사용될 수 있는 방법을 제공한다. 따라서, NN이 데이터 세트의 특징을 알지 못할 때를 알려주고, 추가 훈련이 필요한 입력 벡터를 확인할 수 있다.

<113> 훈련된 NN에 민감도 분석을 수행하는 목적은, 가중치가, 뉴런 및 NN이 학습한 것을 결정하기 위해 취할 수 있는 값의 범위를 결정하기 위한 것이다.

<114> 본 발명의 DR 훈련 방법의 중요한 이점은, 가중치-공간 영역 (region)의 경계가 NN이 훈련된 후에 결정될 수 있다는 것이다. 이를 위해, 또 다른 양태에서 본 발명은, 단순히 각각의 가중치가 취할 수 있는 값의 범위라기보다는, 가중치-공간의 실제 면을 찾는 방법을 제공한다. 이것으로부터, 뉴런을 지배하는 것, 즉 훈련 동안 NN이 학습한 것이 결정될 수 있다.

<115> 본 발명의 DR 훈련 방법은 종래의 피드-포워드 NN 훈련 방법의 이점 및 유용성 전부는 아니지만 그것의 대부분을 유지하면서, NN이 고정된-크기로 되는 주요 한계를 제거하고, 또한 그것이 단일 패스로 학습하기 때문에 국소 최소치 문제를 겪지 않는다. 본 발명의 모든 다른 양태에 따른 본 발명의 DR 훈련 방법은, 피드-포워드 NN을 사용하는 모든 애플리케이션에 매우 유용할 것이다. 인공 NN의 목적은, 생물학적 학습을 따라하는 것이지만, 현재까지 알려진 시스템들은 이러한 목적을 달성하는 데 실패하였다. 본 발명의 DR 훈련 방법은, 신경과학, 신경생물학, 뉴런의 생물학적 모델링, 및 아마도 세포 생물학에 관련된 그럴듯한 생물학적 학습 방법을 제공하는 것으로 여겨진다.

<116> 본 발명의 규칙 추출 방법, 즉 가중치-공간의 실제 면을 결정하는 방법, 및 입력 벡터가 드러나는지 또는 드러나지 않는지 결정하는 방법은 NN에 한정되지 않는다. 또한, 이러한 방법들은, 제약-만족 문제 (Constraint-Satisfaction Problems)[이하, "CSP" 또는 "CSP들"이라 함], 최적화 (optimization) 또는 운영 연구 유형 문제 (operational research type problem), 또는 예를 들어 DNA와 같은 데이터 문자열 (string) 분석과 같은 제약 시스템들을 사용하는 다른 분야에 유용할 수 있을 것이다. 따라서, 본 발명의 이러한 양태들은 독립적이고, 본 발명의 DR 훈련 방법에 한정되지 않는다.

<117> 마지막으로, 본 발명의 DR 훈련 방법은 규칙들이 NN으로부터 추출되도록 하기 때문에, NN이 학습하는 것과 NN에 의해 생산된 출력에 더 신뢰성을 준다. 따라서, 본 발명의 방법은 피드-포워드 NN을 사용하는 시스템에 대한 신뢰성을 증가시킬 것이다.

실 시 예

<140> **바람직한 실시예의 상세한 설명**

<141> 이제, 본 발명의 상세한 바람직한 구성이 첨부된 도면을 참조하여 설명될 것이다. 오직 바람직한 실시

예에 의해, LTG 뉴런 모델이 다음의 논의를 위해 사용될 것이다. 다른 많은 뉴런 모델들이 이용가능하고, 따라서 본 발명의 DR 훈련 방법 또는 알고리즘에 따라 NN을 구성하는데 사용될 수 있다는 것이 이해되어야만 한다. 그러므로, 본 발명은 첨부된 도면에 도시된 특정 뉴런 모델에 한정되지 않는다. 따라서, 상세한 설명 전체에 걸쳐, 용어 "LTG" 또는 "LTG들"은, 단순히 "어느 적당한 뉴런 모델"을 의미하는 것으로 구성되어야만 한다.

<142> 이제 설명될 본 발명의 다른 양태와 함께, 본 발명의 DR 훈련 알고리즘은 어느 적당한 컴퓨터 소프트웨어 및/또는 하드웨어를 사용하여 실행될 수 있다. 따라서 본 발명은 어느 특정한 실제적 실행에 한정되지 않는다. 본 발명의 DR 훈련 알고리즘 수행 능력을 평가하는 목적은, 실험을 통해, 본 발명의 다른 양태들 및 알고리즘이 기대한 대로 작동했고, 알고리즘이 코드대로 프로그램되었고, 컴퓨터 소프트웨어를 사용하여 실행되었다는 것을 증명하는 것이다.

<143> NN은 망을 형성하기 위해 다양한 구성으로 함께 연결된 뉴런의 결합이다. 2-입력 피드-포워드 NN (10)의 기본 구조는 예로 도 1에 개략적으로 도시된다. NN (10)은 첫 번째 또는 입력 층 (16)에 배치된 2개의 입력 뉴런 (12),(14)을 포함한다. 각각의 입력 뉴런 (12),(14)은 그것의 출력을 은닉 층 (24)에 배치된 3개의 뉴런 (18),(20),(22)에 제공한다. 그런 후, 은닉 층 뉴런 (18),(20),(22)은 각각 그들의 출력을 출력 층 (28)에 배치된 단일 뉴런 (26)에 제공한다.

<144> NN은 데이터 세트 내의 관계를 결정하기 위해 사용된다. NN을 훈련시키기 전에, 이용가능한 데이터를 2개의 세트로 나누는데, 그 하나는 훈련을 위해 사용될 것이고, 다른 하나는 테스트를 위해 사용될 것이다. 상기 훈련 세트는 NN을 훈련시키기 위해 사용된다. 상기 테스트 데이터 세트는, NN이 훈련될 때까지 보존될 것이다. 훈련 이후, 테스트 데이터 세트를 NN에 제공하여, NN이 상기 데이터를 충분히 잘 학습하였는지, 또는 NN이 상기 데이터의 일부 양상 (aspect)을 놓치고 있는지를 판단한다. 상세한 설명 전체에 걸쳐, 용어 "NN 훈련" 또는 "훈련"은 훈련 데이터 세트로 NN을 훈련시키는 것을 의미하고자 한다.

<145> 대부분의 NN 훈련 알고리즘은, 훈련 조건을 만족시키고자 한 단일 숫자 값을 찾고, NN의 원하는 출력과 실제 출력 사이의 오차를 기초로 한 가중치 값을 반복적으로 변경하여 학습한다.

<146> 본 발명의 DR 훈련 알고리즘은 뉴런을 훈련시키는 데 다른 접근법을 사용한다. 이 훈련 알고리즘은 상기 훈련 조건을 만족시키는 가중치에 대한 단일 값 (single value)을 찾는 데 기초를 두지 않는다. 대신, 본 발명의 DR 훈련 알고리즘은, 상기 훈련 조건을 만족시킬 모든 가중치 값을 찾는다. 이를 위해, 입력 가중치를 임계치와 연결시키는 제약 (constraint)으로 입력 벡터를 변환시키는 것이 바람직하다. 각각의 뉴런은 입력 가중치 세트를 갖고 있으며, 이러한 입력 가중치들이 서로 관계를 맺고, 상기 훈련 조건을 만족시키는 임계치를 형성한다. 제약을 뉴런에 추가함으로써, 뉴런을 활성화시킬 가중치-공간 영역에 또 다른 제약이 놓여진다.

<147> 비록 본 발명이 상기 제약으로 변환되는 입력 벡터에 관해 설명하지만, 본 발명은 단지 제약의 사용에만 한정되지 않는다는 것이 이해되어야만 한다. 상기 제약은 오직 상기 입력 가중치와 임계치 사이의 관계를 나타낸다. 동일한 관계가 다른 방식, 예를 들어 전자적으로 (electronically) 또는 자기적으로 (magnetically) 표현될 수 있고, 그것에 의해 본 발명은 제공된 특정 예에 한정되지 않는다.

<148> 이와 같은 뉴런 훈련 방법을 사용함으로써, 뉴런을 NN에 동적으로 추가하여 NN이 형성된다. 이 방법은 뉴런을 NN에 추가하기 위한 정확한 기준을 제공하고, 뉴런이 앞서 설명한 표준 피드-포워드 위상 (topology)에 추가될 수 있다.

<149> 이러한 뉴런 훈련 방법을 통해, 데이터 세트가 단일 패스로 학습된다. 또한, 상기 제약이 뉴런 각각의 임계치와 가중치 사이의 관계를 정의하고, 뉴런이 NN에 추가될 때, 명제 논리에 따라 추가되기 때문에, 훈련 동안 학습된 규칙을 추출하는 것이 단순한 문제가 된다.

<150> NN이 본 발명의 DR 훈련 알고리즘으로 데이터 세트를 학습하기 위해, 그 과정의 순서가 정해지는 것이 바람직하다. 각각의 데이터 세트는 독특하고, 많은 출력들을 가지고 있으며, 이것은 상기 데이터 세트에 따라 다르다. 이 순서를 다음과 같이 간단히 요약할 수 있다: (1) NN의 초기화; (2) NN에 의해 학습될 데이터 준비; (3) NN에 의해 학습될 데이터 적용; 및 (4) 필요한 대로 NN에 뉴런을 할당.

<151> 이미 논의된 대로, 본 발명의 바람직한 실시예에 따라 사용되는 뉴런은 LTG 또는 맥컬럭-피트 뉴런이고, 이것은 제약 세트를 포함하기 위해 변경된다. LTG에 대한 상기 초기화 단계 (1)는 상기 입력 뉴런을 출력 뉴런에 연결시키고, 처음 훈련될 출력을 선택하는 것을 수반한다.

<152> 상기 데이터 준비 단계 (2)는, NN에 의해 학습될 데이터를 준비하기 위한 많은 단계들을 포함하는데,

이들은 다음과 같다: (i) 만일 NN에 제공된 데이터가 훈련에 맞는 적당한 포맷에 있지 않는다면, 상기 데이터 세트를 NN에 제공하기 전에 적당한 포맷으로 변환시키는 것이 바람직하다. 본 발명의 바람직한 실시예에 따라, 상기 적당한 데이터 포맷은 이진 데이터이다. 따라서, 본 발명의 DR 훈련 방법으로 훈련될 데이터 세트는, NN에 제공되기 전에 이진수로 변환된다. 데이터를 디지털화 (digitising)하는 적합한 방법이 사용될 수 있다. 본 발명의 다른 양태에 따라, 데이터를 디지털화하는 적당한 방법은, 후에 실험 결과가 논의된 곳에서 논의한다. 비록 이진 데이터가 훈련 동안 바람직한 데이터 포맷으로 제공되지만, 본 발명의 DR 훈련 알고리즘은 또한, 예를 들어 부동-소수점 데이터와 같은 다른 데이터 포맷으로 작동할 수 있다는 것이 이해되어야만 하고, 그것에 의해, 본 발명은 주어진 특정 실시예에 한정된 것으로 해석되지 않아야만 한다; 및 (ii) 본 발명의 DR 훈련 알고리즘은 단일 패스 알고리즘이기 때문에, 입력 벡터의 제공 순서에 유의하는 것이 바람직한데, 이 순서에 따라 NN이 어떠한 규칙을 학습하고, 얼마나 잘 수행되는지 영향받는다. 비록 고려할 가치가 있지만, 상기 입력 벡터의 제공 순서는, 본 발명의 DR 훈련 알고리즘이, 입력 벡터가 뉴런을 NN에 추가되도록 하는 것을 탐지하고 보고할 수 있는 NN을 구성하는 것만큼 중요한 것은 아니다. 이 단계는, 훈련된 NN이 불완전하게 수행되는 상황에 사용되는 것이 바람직하다. 이 단계는, 상기 데이터 세트가 상당히 복잡하여 LTG를 NN에 추가할 필요가 있을 때마다 추천된다.

<153> 다음 단계 (3)는 데이터를 NN 입력에 적용하는 것인데, 상기 NN 입력은 학습될 LTG의 임계치와 가중치 사이의 관계인 제약으로 변환되는 것이 바람직하다. 만일 은닉 층이 있다면, 상기 LTG의 출력은 다음 층 LTG로의 입력이 되고, 그런 다음 상기 LTG는, 그것이 받은 입력 벡터를, 학습할 수 있는 제약으로 변형시킨다. NN이 원하는 출력을 생산할 때까지 이 과정을 반복한다. 만일 그것이 NN에 의해 학습될 수 있다면, 훈련이 다음 입력 벡터로 계속되고, 그렇지 않으면 상기 과정은 하나 이상의 LTG를 NN에 추가하는 다음 단계 (4)로 이동한다.

<154> 입력 벡터가 학습될 수 없는 적어도 2개의 가능한 시나리오가 있다: (i) 만일 은닉 층이 상기 입력 벡터를 학습할 수 없다면, 새로운 LTG를 상기 은닉 층에 추가하고; 및 (ii) 만일 출력 층이 그것의 입력 벡터를 학습할 수 없다면, 이 경우 새로운 층을 NN에 추가하여 오래된 출력이 상기 은닉 층에서 LTG가 된다. 그런 다음, 또 다른 LTG를 이 새로운 은닉 층에 추가하여, 오래된 출력 유니트가 학습할 수 없는 것을 학습하고, 이러한 상기 LTG들을 새로운 출력에 연결시켜 그 출력을 결합시킨다.

<155> LTG를 NN에 할당한 후, 다음이 중요하다: (a) 새로운 LTG를 NN에 있는 기존의 LTG에 연결시킨다; (b) 새롭게 추가된 LTG의 제약 세트가 비워지도록 설정하거나, 이전의 마지막 LTG로부터의 제약을 새로운 LTG에 복사한다; (c) 새로운 LTG의 추가로 인해, NN은 이전에 학습한 것을 확실히 기억한다. NN은 단일 패스 훈련 알고리즘이기 때문에, 새롭게 추가된 LTG를 통해, NN이 이전에 학습한 것을 계속해서 생산하는 것이 중요하다.

<156> 비록 본 발명의 DR 훈련 알고리즘이 번호 (1)에서 (4)의 과정 순서로 제공되지만, 이러한 단계 또는 적어도 이러한 단계 각각의 양태들은 제공된 것 이외의 순서로 수행될 수 있다는 것이 이해되어야만 한다. 예를 들어, 단계 (1) 및 (2)의 경우에, 이용가능한 데이터 세트를, 훈련될 NN의 출력이 선택되기 적당한 데이터 포맷으로 변환시킬 수 있다 (도 2 참조). 마찬가지로, 단계 (4)의 경우에, 새로운 출력 LTG를 추가하기 전에 새로운 은닉 층 LTG를 NN에 추가할 수 있고, 그 역도 마찬가지이다. 따라서, 본 발명의 DR 훈련 알고리즘은 제공된 단계의 특정 순서에 한정되지 않는다.

<157> 이제, 본 발명의 다른 양태와 함께, 본 발명의 DR 훈련 알고리즘의 바람직한 실시예가 상기 설명된 단계에 따라 제공될 것이다: (1) NN의 초기화; (2) 데이터 준비; (3) 학습될 NN에 데이터 제공; 및 (4) 최종적으로, 필요시 LTG 할당.

<158> **DR 훈련 알고리즘 설명**

<159> 도 2는, 본 발명의 바람직한 실시예에 따라 만들어진 NN 훈련 방법 또는 알고리즘 (30)의 흐름도를 도시한다.

<160> 상기 훈련 과정은 LTG의 입력 층으로 시작한다. 이제, LTG를 NN에 동적으로 추가하기 위한 DR 훈련 알고리즘 (30)이 다음 단계로 요약 및 제공된다:

<161> **(1) NN의 초기화**

<162> DR 훈련 알고리즘 (30)에 따른 NN의 초기화는 일반적으로 도 2의 블록 (block)(32)으로 나타낸다. 블록 (32)에서, NN을 초기화하는 과정은 다음 단계를 포함하는 것이 바람직하다:

<163> a) 출력 벡터의 차원 (dimension) 각각을 개별적으로 훈련시킨다. 학습될 차원 O_j 선택.

<164> b) 출력 LTG O_j의 제약 세트를 비움.

<165> c) 마지막으로, 출력 LTG O_j를 입력 층에 연결.

<166> **(2) NN에 의해 학습될 데이터 준비**

<167> DR 훈련 알고리즘 (30)에 따라 NN에 의해 학습될 데이터를 준비하는 과정은 일반적으로 도 2의 블록 (31) 및 (33)으로 나타낸다. NN에 의해 학습될 데이터 준비 과정은 적어도 다음 단계들을 포함하는 것이 바람직하다:

<168> a) 본 발명의 DR 훈련 알고리즘 (30)은 이진 데이터로 작동하는 것이 바람직하기 때문에, 도 2의 블록 (31)에 도시된 것처럼, 훈련 전에 데이터 세트를 이진수로 변환시키는 것이 필요할 수 있다. 본 발명의 다른 양태에 따라, 훈련 동안 NN에 제공되기 전에 다양한 유형의 데이터 세트를 이진수로 변환시키는 적당한 기법은 후에 논의될 것이다. 다른 데이터 포맷이 본 발명의 DR 훈련 알고리즘 (30)에 따라 사용될 수 있고, 그것에 의해 블록 (31)은 단순히, 이용가능한 데이터 세트를 적당한 데이터 포맷으로 변환시키는 것을 의미하는 것으로 이해되어야만 한다.

<169> b) 상기 훈련 세트에 불일치한 데이터가 있는지 결정. 불일치한 데이터는, 다른 출력을 생산하는, 두 개 이상의 동일한 입력 벡터, x_i가 있는 곳에서 발생한다. 불일치한 데이터의 예는 $x_i \rightarrow 0$ 및 $x_i \rightarrow 1$이고, 여기서 동일한 입력 벡터는 상기 데이터 세트에서 한 번 이상 나타나고 다른 출력을 생산한다. 만일 불일치가 존재한다면, 오직 하나의 입력 벡터 x_i가 사용되어야만 한다. NN이 이 데이터를 학습할 수 있을 것이지만, 잘 수행되지는 않을 것이다. 만일 NN이 불일치한 데이터를 학습한다면, 그것은 모든 입력에 대해 출력 0일 것이다. 불일치한 출력이 있는지 결정하기 위해 입력 벡터에 점검 (check)을 수행하여, 이러한 상황을 피하는 것이 바람직하다. 상기 훈련 세트에 불일치한 데이터가 있는지 결정하는 이 과정은 도 2에 구체적으로 도시되지 않았지만, 동일한 것이 블록 (31) 또는 (33)의 일부로서 수행될 수 있다.

<170> c) 도 2의 블록 (33)에 도시된 것처럼, 적당한 분류 기법을 사용하여 학습될 데이터를 분류하는 것이 바람직하다. DR 훈련 알고리즘 (30)이 데이터를 무작위로 학습하는 것이 가능하지만, 생산된 NN 결과물은 상기 데이터를 효과적으로 분류할 수 없을 것이다. 따라서 바람직한 분류 기법은 다음을 포함한다:

<171> - 상기 입력 벡터들을 2개의 그룹, 즉 출력에 대해 0을 생산하는 입력 벡터들과 1을 출력하는 입력 벡터들로 분류. 상기 입력 벡터를 2개의 세트, 즉 1을 출력하는 입력 벡터들과 0을 출력하는 입력 벡터들로 분류. 이 2개의 세트 중 어느 하나가 먼저 학습될 수 있다; 또는

<172> - SOM (자기 조직화 맵)으로 상기 데이터 분류.

<173> 앞서 논의된 대로, 본 발명은 특정 분류 기법에 한정되지 않는다.

<174> d) 입력 벡터 세트로부터 단일 리스트를 만든다. 이 단계는 블록 (33)으로 나타낸 상기 분류 단계의 일부이다.

<175> e) 상기 0 입력 벡터가 학습될 데이터 세트에 이용가능한지 결정. 이 0 벡터는 0으로 설정된 모든 입력 차원들을 갖는다. 만일 상기 입력 벡터가 이용가능하다면, 이 입력 벡터를 그것의 출력과 상관없이 먼저 학습되도록 분류한다. 상기 0 벡터가 훈련 동안 이용가능하고, 그것이 출력과 상관없이 먼저 학습되는 것이 바람직하지만, 만일 그것이 이용가능하지 않다면, 중요하지는 않다. 다시 한번, 이 단계는 블록 (33)으로 나타낸 상기 분류 단계의 일부이다.

<176> **(3) NN에 의해 학습될 데이터 적용**

<177> DR 훈련 알고리즘 (30)에 따라 NN에 의해 학습될 데이터를 적용하는 과정은 일반적으로 도 2의 블록 (34)로 나타낸다. 블록 (34)에서, 각각의 패턴 (또는, 입력 벡터와 그것의 연관 출력의 결합)이 더 이상 학습할 것이 없을 때까지 NN의 출력에 대해 학습되는 것을 볼 수 있다. DR 훈련 알고리즘 (30)에 따라 출력에 대한 단일 패턴을 학습하는 과정 (40)의 바람직한 실시예는 도 3에 제공된다.

<178> 과정 (40)은 블록 (41)에서 NN의 첫 번째 층으로 시작한다. 그런 후, 훈련 세트에 있는 각각의 입력 벡터에 대해:

<179> a) 블록 (42)에서, 상기 입력 층에 적용되는 입력 벡터를 기초로 한 제약은 다음 층에 있는 각각의 LTG

가 되도록 구성된다. 상기 제약을 만들기 위해, 상기 LTG의 정의가 사용되고 (이는, 앞서 LTG를 정의한 곳에서 논의되었다), 상기 입력 벡터 x_i 및 상기 LTG의 가중치 벡터, w는 NN 출력을 기초로 한 상기 LTG의 임계치, T와의 관계를 형성한다. 따라서 만일 상기 LTG가 1을 생산하는 것이라면, 구성된 제약은 다음과 같다:

<180> $x_i.w \geq T \rightarrow 1$

<181> 또는, 만일 상기 출력이 0이라면, 생산된 제약은 다음과 같다:

<182> $x_i.w < T \rightarrow 0$

<183> b) 또한, 블록 (42)에서, 상기 입력 벡터 x_i로부터 구성된 제약이 이 층에 있는 LTG에 의해 학습될 수 있는지를 결정하기 위해 테스트를 수행한다. 제약을 학습하는 것은 LTG의 제약 세트에 제약을 추가할 수 있는 것이다. 만일 수치 해가 찾아진다면, 제약을 추가할 수 있다. 어떠한 수치 해가 상기 알고리즘인지는 관심 대상이 아니고, 수치 해를 찾을 수 있다는 것만이 중요하다. 이것은, 제약들 사이에 교차 구역 (intersection)이 있어야만 하는 것과 동일하다. 이 테스트는, LTG를 NN에 추가하기 위한 기준을 구성한다. 만일 상기 LTG의 어느 것도 상기 입력 벡터로부터 형성된 제약을 학습할 수 없다면, 이것이 새로운 LTG를 NN에 할당하기 위한 기준이 된다.

<184> - 만일 상기 LTG가 상기 입력 벡터를 학습할 수 있다면, 블록 (43)에서 제약을 상기 LTG의 제약 세트에 추가한다. 새로운 제약의 추가함으로써, 상기 LTG가 활성화되도록 할 LTG의 가중치-공간 영역이 감소된다. 그런 후, 이 층으로부터의 출력을 블록 (45)에서 다음 층에 적용하고, NN이 정확한 출력을 출력할 때까지 과정 (40)을 반복한다 [블록 (42)로 되돌아감]. 블록 (44)에서, 현재 층이 출력 층인지 결정되고, 만일 그렇다면 과정 (40)은 블록 (46)에서 종결되는데, 여기서 만일 학습할 패턴이 더 있다면 다음 패턴이 학습된다. 만일 어느 시점에서, 입력 벡터가 층에서 학습될 수 없다면, 그것은 LTG를 할당하기 위한 토대 (ground)가 된다 [도 3의 블록 (47) 내지 (49)로 나타냄- 아래의 단계 4 참조]. 각각의 층은, NN의 원하는 출력을 출력할 수 있는 LTG를 가져야만 한다. 이전 층으로부터 입력을 받는 층의 목적은, 상기 이전 층의 출력을 결합하여, NN의 원하는 출력을 생산하는 것이다.

<185> - 상기 언급된 대로, 블록 (44)에서 점검한 후, 블록 (45)에서 NN이 정확한 반응을 출력하고 학습될 입력 벡터가 더 있다면, 과정 (40)이 단계 3의 처음으로 되돌아간다 [블록 (42)].

<186> - 만일 블록 (44)에서 NN이 정확한 반응을 생산하고, 학습될 입력 벡터가 더이상 없다면, 이 NN의 훈련 출력은 훈련을 끝내고, 과정 (40)이 블록 (46)에서 종결되는데, 여기서 만일 학습할 패턴이 더 있다면 다음 패턴이 학습된다.

<187> - 만일 블록 (35)에서, DR 훈련 알고리즘 (30)에 의해, 훈련될 NN의 출력이 더 있는지가 결정된다면, 도 2에 도시된 것처럼, 상기 DR 훈련 과정은 초기화 단계 1 [블록 (32)]으로 되돌아간다.

<188> **(4) 필요한 대로 NN에 새로운 LTG 할당**

<189> 필요한 대로 새로운 LTG를 NN에 할당하는 과정은 일반적으로 도 3의 블록 (47) 내지 (49)로 나타낸다. 블록 (47)은 새로운 LTG를 NN의 은닉 층에 할당하는 것을 도시하고, 블록 (49)는 새로운 출력 LTG를 NN에 할당하는 것을 도시한다. 새로운 은닉 층 LTG를 NN에 할당하기 위한 과정 (50)의 바람직한 실시예가 도 4의 흐름도에 도시된다. 마찬가지로, 새로운 출력 LTG를 NN에 할당하기 위한 과정 (60)의 바람직한 실시예가 도 5의 흐름도에 도시된다. 새로운 LTG를 NN에 할당하는 과정 (50),(60)에 대한 이해를 돕기 위해, 도 6a 및 6b가 참조될 것인데, 상기 도 6a 및 6b는 본 발명의 DR 훈련 알고리즘 (30)의 과정 (50),(60)에 따라 NN (70)의 구조를 개략적으로 도시한다.

<190> 도 3에 도시된 출력에 대한 단일 패턴을 학습하는 바람직한 과정 (40)에서, NN에 새로운 은닉 층 LTG을 할당하는 것 [블록 (47)]은, 새로운 출력 LTG의 할당 [블록 (49)] 이전에 수행되는 것으로 도시된다. 도 3에서, 만일 LTG가 블록 (42)에서 상기 입력 벡터 (또는 패턴)를 학습할 수 없다면, 블록 (47)에서 새로운 LTG를 현재 (은닉) 층에 추가하여 상기 입력 벡터를 학습한다. 새로운 LTG를 블록 (47)에서 현재 층에 추가한 후, 블록 (48)에서 테스트를 수행하여, 상기 현재 층이 NN의 출력 층인지 결정한다. 만일 블록 (48)에서, 상기 현재 층이 NN의 출력 층이 아니라고 결정되면, 과장 (40)은 블록 (45)에서 계속되는데, 여기서 이 (현재) 층으로부터의 출력이 다음 층에 적용된다. 그런 후, 앞서 논의된 대로, NN이 정확한 출력을 출력할 때까지 과정 (40)을 반복한다 [블록 (42)로 되돌아감]. 만일 블록 (48)에서, 상기 현재 층이 NN의 출력 층으로 결정되면, 과정 (40)을 블

록 (49)에서 계속하는데, 여기서 새로운 출력 LTG를 NN에 추가한다. 블록 (49)에서 새로운 출력 LTG를 NN에 할당한 후, 과정 (40)은 블록 (46)에서 계속되는데, 여기서 만일 학습할 패턴이 더 있다면 새로운 패턴이 학습된다.

<191> 비록 도 3의 과정 (40)이, 새로운 출력 LTG를 NN에 할당하기 [블록 (49)] 전에 새로운 LTG를 은닉 층에 할당하는 것 [블록 (47)]을 도시하지만, 새로운 은닉 층 LTG의 할당 이전에 새로운 출력 LTG가 NN에 할당될 수 있다는 것이 이해되어야만 한다. 따라서, 본 발명은 제공된 특정 실시예에 한정되지 않는다. 본 발명의 DR 훈련 알고리즘 (30)에 따른 새로운 은닉 층 LTG의 할당 이전에, 새로운 출력 LTG가 NN에 할당될 수 있다는 것을 보여주기 위해, 이제 도 6a 및 6b의 NN (70)으로의 LTG 할당이 도 3의 바람직한 과정 (40)에 도시된 것에 역순으로 제공될 것이다.

<192> 이제, NN (70)에 새로운 출력 LTG를 할당하는 과정 (60)(도 6a 및 6b)이 도 5를 참조하여 설명할 것이다.

<193> - 만일 출력 LTG가 블록 (42)[도 3]에서 입력 벡터에 대한 필요한 출력을 생산할 수 없다면, 도 6a 및 도 3의 블록 (49)에 도시된 것처럼 새로운 출력 LTG를 NN (70)에 할당한다.

<194> Ⅰ. 도 6a(i) 참조, 현재 출력 LTG, LTG A는 층 N에 있다.

<195> 도 6a(ii) 참조, 또 다른 LTG, LTG B를 층 N에 추가한다.

<196> LTG B에 대한 제약을 빈 (empty) 세트로 초기화하는 것이 바람직하다.

<197> NN (70)의 층 N에 새로운 LTG, LTG B를 할당하는 것은

<198> 도 5의 흐름도에 도시되지 않았지만, 이제 설명될 블록 (61)의

<199> 일부로서 이해되어야만 한다. 마찬가지로, 층 N에 새로운 LTG,

<200> LTG B를 할당하는 것은, 층 N+1에 새로운 출력 LTG, LTG C를

<201> 할당한 후 일어날 수 있다.

<202> Ⅱ. 블록 (61)에서, 출력 O_j에 대한 이 층에 있는 단 하나의 새로운 LTG,

<203> LTG C를 새로운 출력 층, 층 N+1에 추가한다. 그런 후, LTG C의

<204> 제약 세트를 단계 V 및 VI에 따라 초기화하는 것이 바람직하다.

<205> Ⅲ. 만일 블록 (62)에서 층 N ＞1라면, LTG A에 연결된 이전 층,

<206> 층 N-1 (미도시)에 있는 LTG로부터, 새로운 LTG, LTG B에 연결이

<207> 추가된다.

<208> Ⅳ. 또한 블록 (62)에서, 층 N에 있는 LTG A 및 LTG B 각각의 출력이,

<209> 층 N+1에 있는 새로운 출력 LTG, LTG C의 입력에 연결된다.

<210> Ⅴ. 만일 학습될 입력 벡터가 출력 0을 생산한다면, 블록 (63)에서:

<211> a) 층 N에 있는 새로운 LTG B를 훈련시켜, 이 층의 제약으로의 입력을

<212> 학습한다. LTG A가 상기 입력을 학습할 수 없기 때문에, 이러한 LTG,

<213> LTG B 및 LTG C를 추가한다.

<214> b) 모든 제약을 LTG B의 ≥ 임계치로 설정하면서, LTG A로부터의

<215> 제약을 새로운 LTG, LTG B의 제약 세트로 복사한다.

<216> c) 층 N에 있는 LTG A와 LTG B 사이의, 층 N+1에 있는 새로운 출력

<217> LTG, LTG C에 AND를 형성하는 제약을 추가한다.

<218> Ⅵ. 만일 학습될 입력 벡터가 출력 1을 생산한다면, 블록 (64)에서:

<219> a) 층 N에 있는 새로운 LTG B를 훈련시켜, 이 입력의 제약을

<220> 학습한다.

<221> b) 모든 제약을 LTG B의 < 임계치로 설정하면서, LTG A로부터의

<222> 제약을 새로운 LTG, LTG B의 제약 세트로 복사한다.

<223> c) 층 N에 있는 LTG A와 LTG B 사이의, 층 N+1에 있는 새로운 출력

<224> LTG, LTG C에 OR를 형성하는 제약을 추가한다.

<225> - 만일 층 N에 있는 LTG들 중 어느 것도 블록 (42)[도 3]에서 필요한 출력을 생산하는 것을 학습할 수 없다면, 도 6b 및 도 3의 블록 (47)에 도시된 것처럼, 새로운 LTG, LTG D를 NN (70)의 그 층, 층 N에 할당한다.

<226> 이제, NN (70)[도 6a 및 6b]에 새로운 은닉 층 LTG를 할당하는 과정 (50)을 도 4를 참조하여 설명할 것이다.

<227> Ⅰ. 블록 (51)에서, LTG의 어느 것도 상기 데이터를 학습할 수 없는

<228> 층 N에 추가 LTG, LTG D를 추가한다. 그런 후, 제약 세트를 단계 V 및

<229> VI에 따라 초기화하는 것이 바람직하다. 나머지 단계, 단계 Ⅱ 내지

<230> VI는 일반적으로 호환적이고, 따라서 이러한 절차 단계의 순서는

<231> 도 4에 도시된 것으로 달라질 수 있다.

<232> Ⅱ. 블록 (53)에서, NN 출력 O_j에 대해, LTG D의 출력으로부터의 연결이,

<233> 층 N에 대한 출력 층을 형성하는 층 N+1에 있는 모든 LTG에

<234> 만들어진다. 블록 (54)에서, 층 N+1에 있는 LTG (이 경우, LTG C)를

<235> 갱신하여, 층 N에 있는 다른 LTG, LTG A 및 B에 의해 학습되지

<236> 않을 것을 기초로, 새로운 LTG, LTG D로부터의 입력을 어떻게

<237> 할지 안다.

<238> Ⅲ. 만일 층 N >1이라면, 블록 (52)에서, NN 출력 O_j에 대해, 이전 층,

<239> 층 N-1 (미도시)에서 입력을 형성하는 모든 LTG들에서 LTG D로 입력

<240> 연결을 추가한다.

<241> Ⅳ. 블록 (55)에서, 새로운 LTG, LTG D를 훈련시켜, 층 N에 의해

<242> 학습될 수 없는 입력 벡터를 학습한다. 새로운 LTG, LTG D를

<243> 훈련시켜, 층 (N)에 있는 다른 LTG들에 의해 학습될 수 없는

<244> 입력 벡터를 학습하는 과정 [블록 (55)]에 대한 이해를 돕기 위하여,

<245> 관련된 바람직한 절차의 더 자세한 설명을 포함하는 또 다른 블록,

<246> 블록 (56)이 제공된다.

<247> Ⅴ. 만일 학습될 입력 벡터가 출력 0을 생산한다면, 블록 (57) 및

<248> (58)에서:

<249> a) 모든 제약을 ≥ 새로운 임계치로 설정하면서 [블록 (58)],

<250> 이 층, 층 N에 있는 이전의 마지막 LTG, LTG B의 제약들을

<251> 새로운 LTG, LTG D의 제약 세트로 복사한다.

<252> b) LTG C는, 층 N에 있는 LTG D 및 다른 LTG들로부터의 입력에 대해

<253> 그것의 제약 세트에서 AND를 형성한다 [블록 (54) 참조].

<254> 이 논리는 (A ... B) AND D이다.

<255> VI. 만일 학습될 입력 벡터가 출력 1을 생산한다면, 블록 (57) 및

<256> (59)에서:

<257> a) 모든 제약을 < 새로운 임계치로 설정하면서 [블록 (59)],

<258> 이 층, 층 N에 있는 이전의 마지막 LTG, LTG B의 제약들을

<259> 새로운 LTG, LTG D의 제약 세트로 복사한다.

<260> b) LTG C는, 층 N에 있는 LTG D 및 다른 LTG들로부터의 입력에 대해

<261> 그것의 제약 세트에서 OR을 형성한다 [블록 (54) 참조].

<262> 이 논리는 (A ... B) OR D이다.

<263> - 도 3을 참조하여, 만일 블록 (47)에서 새로운 LTG, LTG D를 할당한 후, NN (70)이 정확한 반응을 출력하고, 학습할 입력 벡터가 더 있다면 [블록 (48)], 과정 (40)은 단계 3의 처음으로 되돌아간다 [블록 (42) 대 블록 (45)].

<264> - 다시 도 3을 참조하여, 만일 블록 (47)에서 새로운 LTG, LTG D를 할당한 후, NN (70)이 정확한 반응을 출력하고, 학습될 입력 벡터가 더 이상 없지만, 학습될 출력이 더 있다면 [블록 (48)], 과정 (40)은 단계 1의 초기화로 되돌아간다 [도 2의 블록 (32)].

<265> 층 N에 있는 이전의 마지막 LTG, LTG B로부터의 제약을 새로운 LTG, LTG D에 복사할 때, 다른 논리 결합이 가능하다는 것이 이해되어야만 한다. 제공된 특정 실시예는 단순히, LTG를 NN (70)에 할당하는 두 가지 경우 [과정 (50),(60)]에서 작동하는 바람직한 논리 배열 (logic arrangement)이다. 사용된 특정 학습 논리는 본 발명에서 중요하지 않으나, 데이터는 임의의 형식으로 교차 복사되어야만 한다. 그렇지 않으면 NN (70)은 전에 학습한 모든 것을 완전히 잊어버릴 것이다.

<266> 이제, DR 훈련 알고리즘 (30)에 따른 NN (70)으로의 LTG 할당이 임의의 순서로 수행될 수 있다는 것이 이해되어야만 한다. 따라서, 도 6a 및 6b에 따라, LTG C가 새로운 출력 층, 층 N+1에 추가되기 전에, LTG D가 층 N에 추가될 수 있었다. 마찬가지로, NN (70)에 LTG를 할당하는 두 가지 경우 [과정 (50),(60)]에서, 절차 단계, 단계 I 내지 VI는 일반적으로 호환적이고, 그것에 의해 본 발명이 제공된 특정 단계 순서에 한정되지 않는다는 것 또한 이해되어야만 한다.

<267> **DR 훈련 알고리즘 단계의 상세한 설명**

<268> 이제, 본 발명의 DR 훈련 알고리즘 (30) 및 다른 양태에 대한 이해를 돕기 위해, 알고리즘 (30)의 단계 각각에 대한 더 상세한 설명이 제공될 것이다. 이러한 훈련 알고리즘 (30)은 피드-포워드 NN 구조를 기초로 하는데, 이 구조는, 훈련 조건을 만족시키기 위한 단일 숫자 값을 찾기 위한 LTG의 NN 훈련에 대한 종래의 접근법으로부터 변화한다. 앞서 간단히 논의했듯이, 이러한 접근법은 대신, 훈련 데이터 세트의 단일 패스로 학습하는, 각각의 LTG에 대한 훈련 조건을 만족시키는 가중치-공간에서 영역을 찾고, LTG가 NN에 동적으로 할당되도록 하고, 입력 벡터가 분류될 수 있는지 결정할 수 있고, 훈련 동안 학습된 규칙이 NN으로부터 쉽게 추출되도록 한다.

<269> **초기화:** 특히, 도 2의 DR 훈련 알고리즘 (30)의 블록 (32) 참조. 출력 훈련의 개시, 훈련시킬 출력 선택, 및 그 출력에 대한 출력 LTG의 추가는 적어도 두 개의 단계를 통해 실행되는 것이 바람직하다.

<270> 학습할 출력을 선택하는 첫 번째 단계: NN은 데이터 세트에 대한 NN의 출력 각각을 개별적으로 생산하는 것을 학습한다. 각각의 출력, 또는 입력 벡터의 차원은 개별적인 학습 과정으로 처리된다. 이는, 출력 벡터의 차원 각각이 독립적이기 때문에 정당화된다.

<271> 상기 훈련 알고리즘과 상관없이, 각각의 LTG는, 공통의 입력을 공유함에도 불구하고, NN의 은닉 또는 출력 층인 그 층에 있는 다른 것들과 관계없이 독자적으로 작동한다. 이것은 역전파로 훈련된 NN에 문제를 일으킬 수 있는데, 즉, NN에 있는 오차가 NN을 통해 피드백될 때, 가중치가 증가돼야 하는지 또는 감소돼야 하는지

결정하기란 불가능하다. 팰만 외 (Fahlman et al.)는 이것을 신용 배정 문제 (credit assignment problem)라고 불렀다. 이 새로운 알고리즘 (30)은 LTG 각각의 독립적 작동을 활용하여, 각각의 출력 LTG를 개별적으로 훈련시킨다. 이것이, NN의 구성에 대한 주요 원칙을 형성한다. 일단 상기 출력 벡터의 차원이 선택되면, 상기 출력 LTG를 NN에 추가한다.

<272> 상기 출력 LTG를 추가하는 두 번째 단계: 처음에, 입력 층의 LTG를 출력 층 LTG에 완전히 연결시킨다. 또한, 상기 출력 LTG에 있는 제약 세트는 비워진 것이 바람직하다. 이제, 데이터를 상기 출력 LTG에 의해 학습되도록 준비할 수 있다.

<273> **데이터 준비:** 특히, 도 2의 DR 훈련 알고리즘 (30)의 블록 (31) 및 (33) 참조. 이것이 단일 패스 훈련 알고리즘이기 때문에, 학습될 NN에 데이터를 제공하는 순서가 중요하다. 상기 데이터를 준비하는 데 적어도 네 개의 단계가 관여하는 것이 바람직하다.

<274> 첫 번째 단계, 만일 NN에 제공될 데이터가 훈련에 맞는 적당한 포맷에 있지 않다면, 도 2의 블록 (31)으로 나타난 대로, 데이터 세트를 NN에 제공하기 전에 적당한 포맷으로 변환시키는 것이 바람직하다.

<275> 두 번째 단계, 데이터의 불일치를 점검한다: 불일치한 데이터는 NN을 훈련시킬 때 문제를 일으킨다. 상충 (conflicting)하는 출력을 생산하는 입력 벡터 x_i의 2개 이상의 경우가 있기 때문에, 데이터의 불일치를 점검할 필요가 있다. 즉, x_i는 출력 0뿐만 아니라 1을 생산한다. 비록 NN은 이 데이터를 학습할 수 있지만, 이 데이터는 불일치하고, 이러한 문제를 피하기 위해, 독특한 입력 벡터로 NN을 훈련시키는 것이 바람직하다. 이 단계는 도 2의 DR 훈련 알고리즘 (30)의 블록 (31) 또는 블록 (32)에서 수행될 수 있다.

<276> 세 번째 단계, NN을 훈련시키기 전에 훈련 세트에 있는 데이터 정렬: 0 벡터를 학습하는 것은 많은 시스템에 불안정성을 야기하는데, 이러한 시스템에는 음성 제약 (negative constraint)을 지닌 심플렉스법 (simplex method) 및 데이터 세트를 학습하기 위해 역전파로 훈련된 피드-포워드 NN (여기서, 입력 벡터 0은 1이 될 출력을 필요로 한다)이 포함된다. 그것이 NN에 문제가 되는 이유는, 뉴런에 있는 임계치가 음 (negative)이 될 필요가 있기 때문이다. 본 발명의 DR 훈련 알고리즘 (30)은, NN을 훈련시키기 전에 훈련 세트에 있는 데이터를 정렬하여 이러한 상황을 피한다. 상기 0 벡터는, 모든 입력 차원이 0인 입력 벡터로 정의된다. 예를 들어, 3개의 입력을 지닌 NN 또는 뉴런에 대한 0 벡터는 [0 0 0]이다. 상기 0 벡터가 상기 훈련 세트에서 이용가능할 때, 이것이 만일 상기 0 벡터로 인해 NN이 1을 출력한다면 NN이 경험할 수 있는 불안정성을 피하기 때문에, 상기 벡터가 처음에 NN에 의해 학습된다.

<277> 네 번째 단계, 만일 상기 입력 벡터 0이 드러난다면, 그것이 이용가능하다면, 처음에 학습되어야만 한다. 그런 후, 블록 (33)에서, 특히 만일 훈련 동안 LTG가 NN에 할당될 필요가 있다면, NN을 훈련시키기에 앞서 상기 입력 벡터를 임의의 순서로 분류시키는 것이 바람직하다. 본 발명의 알고리즘 (30)은, 데이터가 NN에 제공됨에 따라 학습되는 것과 마찬가지로, 내장된 서치 메커니즘 (in-built search mechanism)을 갖지 않는다. 이것은 DR 학습이 NN을 단일 패스로 훈련시켜, 제공된 대로 모든 데이터 세트를 학습할 수 있어야만 하기 때문이다.

<278> 본 발명의 또 다른 양태에 따라, 바람직한 분류 방법 [블록 (33)]은 1과 0 각각을 생산하는 세트로 상기 데이터를 분류하고, 상기 데이터 세트에 따라 1 또는 0을 처음에 생산하는 벡터들을 학습하는 것이다. 대략적인 가이드라인 (guide line)에 따라, 만일 0→0라면, 0을 처음에 출력하는 벡터들을 학습하고, 그렇지 않으면 1을 처음에 출력하는 벡터들을 학습한다.

<279> 또 다른 가능한 분류 방법은 SOM, 자기 조직화 맵을 사용하는 것인데, 이 SOM은 생물학적 뇌의 분류 기법 중 하나를 따라한 것이다. 상기 생물학적 뇌는 피질 (cortex) 면 (surface) 위의 SOM과 유사한 메커니즘을 사용한다. SOM은, 상기 데이터 세트에 있는 특징들의 2차원 집합 표시 (class representation)로 입력 벡터를 분류하거나 조직화하여 작동한다. 상기 특징들은 그들의 출력과 관계없이 입력 벡터를 분류한다. 상기 SOM에 있는 어떤 특징들에 속함에 따라 분류된 입력 벡터는 특징들로 분리 및 분류될 수 있고, 수집되어 훈련 메커니즘으로 공급될 수 있다. 이러한 방식으로, DR 훈련은, 상기 데이터 세트에 있는 특정한 특징들을 분류하는 방법을 학습하는 LTG들을 함께 하나의 무리 (cluster)로 지을 수 있다.

<280> 이러한 구조는 3차원, 2차원 SOM으로 구상될 수 있고, 상기 3차원은 DR 훈련된 NN이다. 대략 돔-형태 (dome-shaped)인 생물학적 뇌를 매우 단순화한 도에서, SOM은 상기 면 위에 있고, 피드포워드 NN은 상기 면으로부터 나온다. 가능한 하나의 모델은, 상기 피질이 피드포워드 뉴런과 함께 연결된 SOM 층으로 구성된 것일 수 있다.

<281> 본 발명의 DR 훈련 알고리즘 (30)으로 사용될 수 있는 잠재적으로 많은 다른 분류 기법이 있다는 것과, 그것에 의해 본 발명은 제공된 특정 실시예에 한정되지 않는다는 것이 이해되어야만 한다.

<282> **NN에 데이터 적용:** 특히, 도 2의 DR 훈련 알고리즘 (30)의 블록 (34) 및 (35), 더 자세히는 도 3의 과정 (40)을 참조. 본 발명의 DR 훈련 알고리즘 (30)은, 앞서 설명된 LTG를 사용하는 것이 바람직하다. 또한, DR 훈련 알고리즘 (30)은 그것의 전달 함수 (transfer function)로, 헤비사이드 (Heaviside), 또는 계단 (step) 함수를 사용하는 것이 바람직하다. 이러한 유형의 게이트 (gate)가 사용되는 것이 바람직한데, 왜냐하면, 그것이 입력 공간에 있는 집합들 사이에 뚜렷한 경계를 제공하고 입력 벡터의 집합들을 완벽하게 나누기 때문이다.

<283> 본 발명의 DR 훈련 알고리즘 (30)은 상기 LTG와 잘 작동하는데, 그 이유는, 그것이 상기 훈련 조건을 만족시키는 임계치 및 가중치에 대한 단일 숫자 값을 찾지 못하고, 대신에 상기 LTG의 가중치-공간을 제약하여 학습하기 때문이다. 따라서, 상기 가중치 및 임계치를 만족시키기 위해 상기 가중치-공간에서 단일 값을 찾는 대신에, 공간 영역을 찾는다. 지도 훈련 (supervised training) 개념에 따라, 입력 벡터 및 원하는 출력으로 제약이 형성된다.

<284> 상기 LTG에 있는 임계치, T는 상수 (constant)로 처리된다. 코하비 (Kohavi)가, 상기 LTG의 임계치가 그렇게 처리될 수 있다는 것을 보여주었다. 코하비는 보수 (complement) 가중치 (논리적 NOT 사용)를 사용하여, $T \leq 0$ 및 $T > 0$인 임계치를 설명한다. 본 발명의 훈련 알고리즘 (30)에서, 상기 데이터가 훈련 동안 $T \leq 0$ 또는 $T > 0$을 결정한다.

<285> 원래의 LTG에 만들어진 단 하나의 변형은, 비워진 상태로 초기화되는 제약 세트의 포함이다. 이 변형은, LTG가 가중치와 뉴런의 임계치 사이의 관계에 대해 학습한 것을 저장하기 위해 사용되기 때문에, 실행 목적에 필요하다. 앞서 논의된 대로, 본 발명은 제약 단독의 사용에 한정되지 않는다. 제약은 오직 입력 가중치와 임계치 사이의 관례를 나타내고, 단지 본 발명의 바람직한 특징이다.

<286> 이제, 제약이 어떻게 만들어져, 단일 LTG를 훈련시키는지에 대해 논의할 것이다.

<287> **단일 LTG에 대한 제약 형성:** 특히, DR 훈련 알고리즘 (30)에 따라 출력에 대한 단일 패턴을 학습하기 위한, 도 3의 바람직한 과정 (40)을 참조. 상기 LTG는, 블록 (42)에서 학습하기 위한 입력 벡터 (또는 패턴)를 제공한다. 이러한 입력 벡터는, 상기 LTG의 제약 세트로 기록된 제약으로 변환된다. 학습에 앞서, 상기 제약 세트를 비워진 상태로 초기화하는 것이 바람직하다.

<288> 훈련을 시작하기 위해, 제약을 구성하는 첫 번째 단계 [블록 (42)]는, 각각의 입력 벡터, x_i를 상기 LTG의 유입 (incoming) 가중치 벡터 w에 적용하여, $x_i.w$를 생산하는 것이다. 생산물 $x_i.w$는, LTG의 작동을 정의하는 방정식 1.1을 기초로, 상기 LTG의 임계치 T와 적어도 2개의 가능한 관계를 갖는다. 이 2개의 가능한 관계 각각은 연관 출력 (associated output)을 생산한다. T와의 관계 및 연관 출력은 방정식 2.1 및 2.2로 나타낸다.

<289> $$x_i.w \geq T \rightarrow 1 \qquad (2.1)$$

<290> 또는

<291> $$x_i.w < T \rightarrow 0 \qquad (2.2)$$

<292> 본 명세서 앞 부분에서 지도 훈련을 설명하였다. 지도 훈련의 원리를 사용하여, 만일 필요한 출력이 1이라면, 필요한 제약은 $x_i.w \geq T$이다. 마찬가지로, 만일 생산될 출력이 0이라면, w는 $x_i.w < T$로 제약된다. 이제, 새로운 제약을 상기 LTG의 제약 세트에 추가하는데, 이것은 이러한 제약이 이전에 추가된 다른 제약들과 함께 해를 갖는다는 조건을 갖는다. 이것은 후에 더 상세히 논의될 것이다. 상기 입력 벡터 및 LTG의 가중치로부터 구성된 제약을 LTG의 제약 세트에 추가하는 과정을, 훈련 세트에 있는 모든 n 입력 벡터에 대해 반복한다.

<293> 도 7은 2-입력 LTG NN (80)의 실시예를 도시한다. 이 실시예에서, 가중치 벡터는 $[w_1\ w_2]^T$이고, 입력 벡터는 $[x_1\ x_2]$이고, $[O_j]$는 LTG의 출력이 된다. 만일 상기 출력이 입력 [0 1]에 대해 1이 되면, LTG (82) 상의 제약은 $w_2 \geq T$가 될 것이다. 만일 또 다른 입력 벡터 [1 1]이 출력 0을 생산할 것으로 예상된다며, 제약 $w_1 + w_2 < T$가 또한 추가된다. 이러한 2개의 출력은 $\{w_2 \geq T,\ w_1 + w_2 < T\}$의 LTG (82)에 대한 제약 세트가 될 것이다. 제약 세트를 형성하여, LTG (82)는 상기 입력/출력을 분류하는 것을 학습한다.

<294> 지금껏, 상기 LTG를 훈련시킬 입력 벡터로부터 제약을 구성하는 방법을 설명하였다. 하지만, 새로운 입력 벡터가 학습될 수 있고, 그것의 연관 제약이 상기 LTG에 대한 제약 세트에 추가되기 전에, 상기 입력 벡터가 형성하는 제약을 상기 LTG가 학습할 수 있는지 검증되어야만 한다. 이제, 새로운 입력 벡터가 학습될 수 있는지, 그 결과 제약이 LTG의 제약 세트에 추가되는지 결정할, 입력 벡터를 학습하기 위한 기준에 대해 논의할 것이다.

<295> **입력 벡터 학습 기준:** 특히, 도 3의 과정 (40)의 블록 (42) 참조. LTG의 제약 세트에 제약을 추가할 때 가장 기본적으로 고려할 점은, 그것이 배우는 것을 학습할 수 있는지 결정하는 것이다. 이것은 본 명세서 앞 부분에 더 자세히 설명되었다.

<296> LTG가 입력 벡터를 학습할 수 있는지에 대한 테스트는 LTG의 NN을 만드는 데 기본적인 것이다. 만일 단일 LTG가 모든 데이터를 학습할 수 없다면, 추가 LTG가 필요하다.

<297> 본 발명의 DR 훈련 알고리즘 (30)이 상기 입력 벡터와 연관 출력을 제약으로 변환시키기 때문에, 상기 LTG가 이미 학습한 모든 제약들과 함께 가중치-공간에 해가 있는지 결정하기 위해, 새로운 제약을 테스트할 수 있다. 상기 심플렉스법을 사용하여 이 테스트를 시행할 수 있다. 이 테스트를 통해, 비록 각각의 가중치 및 임계치에 대한 특정 숫자를 찾는 것이 필요하지 않더라도, 상기 가중치 및 임계치에 대한 숫자 값이 확실히 찾아질 수 있다. 상기 제약을 만족시키는, 수치 해를 찾을 수 있다는 것을 아는 것만으로도 충분하다.

<298> 이제, 각각의 가중치 및 임계치에 대한 특정한 수치 해를 찾는 것이, 상기 학습 문제에 대한 일반 해 찾기에 바람직한 이유를 논의할 것이다.

<299> 전통적으로, 각각의 가중치에 대한 단일 숫자 값이 찾아졌는데, 이것은 일반 해를 찾는 것이 불가능하기 때문이었다.

<300> 만일 종래의 훈련 방법, 예를 들어 역전파로 훈련된 NN에 있는 가중치에 대한 해를 찾을 수 있다면, 상기 가중치가 $w \in R^n$으로부터 선택될 때, 모든 가중치에 대한 해의 무한수 (infinite number)가 있다. 이러한 해의 무한수는 각각의 LTG에 대한 가중치-공간 영역을 형성한다. 일반적으로 역전파로 선택된 해는, 미리-지정된 (pre-designated) 오차 허용 내에서 NN을 생산하는, 찾아진 숫자 값의 첫 번째 세트가 되는 경향이 있다. 각각의 LTG에 대해 찾아진 w는, 상기 LTG의 가중치-공간 영역의 단일 값이다.

<301> 상기 가중치 값에 대한 이러한 해는, 훈련 동안 NN에 적용된 입력 값을 기초로 한 일종의 평균치가 되도록 하는데, 여기서 상기 값의 범위가 단일 해를 찾기 위해 손실되기 때문에 극값 (extreme)이 손실된다.

<302> 본 발명에 따른 뉴런 훈련 방법은, 일반 해가 찾아짐에 따라, 훈련 동안 학습된 모든 정보가 보존되도록 한다. 일반 해는, 상기 가중치-공간 영역이 분석되어, 상기 LTG를 활성화시키는 가중치와 그렇게 하지 못하는 가중치 사이의 경계를 찾도록 한다. 모든 것들이 서로 연관되고, 그 밖의 모든 것에 관련하여 이해될 수 있는 것처럼 보임에 따라, 일반 해는 상대적 관계를 정의한다. 일반 해를 찾음으로써, 가중치들 사이의 관계가 분석되고, 그 결과, 상기 입력들 사이의 관계가 분석된다. 마지막으로, 만일 그것이 절대적으로 필요하다면, 상기 데이터 세트 내의 관계를 명확히 구체화하는 특정 숫자 값이 찾아질 수 있다.

<303> 따라서, 본 발명의 DR 훈련 알고리즘 (30)으로 훈련된 LTG에 의해, 새로운 제약이 학습될 수 있는지 결정하는 데 이용가능한 테스트가 있다. 또한, 이 테스트는 NN에 LTG를 추가하기 위한 기준이다. 만일 입력 벡터가 학습될 수 없다면, LTG를 NN에 추가한다.

<304> **NN에 LTG 할당:** 특히, 도 3의 과정 (40)의 블록 (47) 및 (49), 및 본 발명의 DR 훈련 알고리즘 (30)에 따라 NN에 LTG를 할당하기 위한, 도 4 및 5의 바람직한 과정 (50),(60) 참조. 본 명세서 앞에서, 단일 LTG를 훈련시키는 방법을 설명하였다. 단일 LTG의 작동이 LTG의 NN을 만들기 위한 기반을 형성한다. 단일 LTG의 훈련은 다음 3개의 단계를 포함한다:

<305> 1) 입력 벡터를 상기 가중치와 임계치 사이의 관계로 변환;

<306> 2) 상기 LTG가 새로운 입력 벡터를 학습할 수 있는지 결정. LTG가 입력 벡터를 학습할 수 있는지에 대한 기준은, 제약이 상기 LTG의 제약 세트에 추가될 때, 수치 해가 찾아질 수 있는지에 의해 결정된다; 및

<307> 3) 만일 상기 LTG가 상기 입력 벡터를 학습할 수 있다면, 상기 입력 벡터로부터 구성된 제약을 상기 LTG의 제약 세트에 추가한다.

<308> 만일 블록 (42)에서 상기 LTG가 상기 입력 벡터를 학습할 수 없다면, 추가 LTG가 필요하다. 따라서, 단계 2는 NN에 LTG를 할당하기 위한 근본적인 기준을 형성한다. 이제, LTG를 추가해 NN을 만드는 방법을 설명할 것이다.

<309> 오직 입력 벡터가 필요한 출력을 생산할 수 없을 때만 LTG를 할당한다. 이 과정은 다음 단계들을 포함한다:

<310> 1) 새로운 LTG와 NN에 이미 있는 LTG 사이의 연결 형성 [도 4의 블록 (52) 및 (53), 도 5의 블록 (62)]; 및

<311> 2) 확실히 상기 새로운 LTG가, NN이 이미 학습한 것을 잊지 않도록 함.

<312> 이제, 각각의 상기 과정을 더 자세히 논의할 것이다. 우선, NN 구성에 대한 일반적인 접근법을 설명한 후, 출력을 선택하고, 그 출력을 개별적인 문제들로 분리하기 위한 정당화를 설명할 것이다.

<313> **NN 구성:** 처음에, 상기 입력 LTG를 상기 출력 LTG에 완전히 연결한다. 입력및 출력 LTG의 수는 NN에 의해 학습되는 데이터 세트에 따라 다르다. 각각의 출력은 다른 출력의 독립적 학습 임무로 간주된다 [도 2의 블록 (32)].

<314> 이러한 논의를 위해, 단일 출력을 지닌 NN을 우선 조사할 것이다. 본 발명의 DR 훈련 알고리즘 (30)은 위상 (topology)을 가로질러 위로 상승하고, 역전파를 사용할 때 형성된 종래의 NN과 유사한 NN을 형성한다. 그것은 은닉 층 [도 4의 과정 (50)]에 유니트를 할당할 수 있고, 단일 LTG를 포함하는 새로운 출력 층 [도 5의 과정 (60)]을 추가할 수 있다. 그런 후, 이전 출력 층이 NN에서 은닉 층이 된다. 이 새로운 은닉 층은 그것에 할당된 추가 LTG를 가져 [도 3의 블록 (47)], 그 층에 있는 다른 LTG가 학습할 수 없는 것을 학습한다. 3개의 출력 O_1, O_2, O_3을 지닌 NN (90)의 예가 도 8에 나와있다. NN (90)은 모듈로 (Modulo)-8 문제 데이터 세트와 함께 본 발명의 DR 훈련 알고리즘 (30)으로 훈련되었다.

<315> 도 8에 도시된 NN (90)에서, O_3은 은닉 층이, 완전히 훈련한 후, 필요한 해를 생산할 수 있도록 요구하지 않았던 것처럼 보일 수 있다. 하지만, O_1 및 O_2는 은닉 층이 필요하였다. 훈련 초기에, 임계치 T_{11} 및 T_{13}을 지닌 LTG가 원래의 출력이었다. 하지만, 그들은 데이터를 학습할 수 없었고, 그래서 T_{12}를 은닉 층 (92)에 추가하였고, T_{21}은 O_1에 대해 출력이 되었다. 마찬가지로, T_{13}이 O_2를 생산할 수 없을 때, T_{14}를 추가하였다.

<316> LTG를 명명하기 위해 본 명세서에서 사용된 규정 (convention)은, LTG가 임계치 T를 갖고, 층 L에 속하는 것이다. 모든 함수가 단지 3개 층에서만 학습될 수 있기 때문에, 오직 한 자리 숫자를 할당하여, 그 층을 확인한다. 각각의 층은 그것에 있는 LTG의 숫자 N을 갖고, 이들은 그 층에서 k = 1..N에 걸친 수이다. 각각의 LTG는 LTG_{LK}로 언급될 수 있고, 그것과 연관된 임계치, T_{LK}를 갖는다. 각각의 LTG는 입력 연결 가중치 세트를 갖는다. 가중치 벡터의 개별 구성요소는 w_{LKj}로 언급되는데, 여기서 j는, 상기 입력을 제공한 이전 층에 있는 LTG이다.

<317> **NN 형성:** 입력 벡터가 NN에 의해 학습될 수 없을 때, LTG를 NN에 오직 추가한다 [참조: 블록 (42) 형성]. LTG를 NN에 추가할 필요가 있을 때는 두 가지 경우이다: (1) 출력이 필요한 출력을 생산할 수 없는 경우 [블록 (49)]; 및 (2) 은닉 층에 있는 어떤 LTG도 입력 벡터를 학습할 수 없을 경우 [블록 (47)]. 앞서 논의된 대로, NN에 출력 LTG 및 은닉 층 LTG을 할당하는 것은 임의의 순서로 일어날 수 있다.

<318> LTG를 NN에 추가할 때 바람직하게 고려될 필요가 있는 것은 다음과 같다:

<319> a) NN에 있는 기존의 LTG로부터 만들어질 필요가 있는 모든 연결들이 새로운 LTG에 만들어진다 [도 4의 블록 (52) 및 (53), 및 도 5의 블록 (62)];

<320> b) NN에 새로운 LTG를 추가한 후, 새롭게 추가된 LTG가 NN에 의해 이전에 학습된 모든 것을 학습하는 것이 중요하다 [블록 (55)]. 이로써, NN에 의해 이전에 학습된 것을 잊어버릴 수 있다는 것을 의미하기 때문에, 망각 (forgetfulness)이라 불리는 상태가 예방된다. 이것을 피할 수 있는 방법은, 본 명세서 후반에서 학습 논리를 설명할 때 논의할 것이다.

<321> c) 새롭게 할당된 LTG로부터 입력을 받는, 이미 NN 내에 존재하는 LTG는 입력을 수용할 준비가 되있어야만 한다 [블록 (54)]. 만일 상기 새롭게 할당된 LTG로부터 이 새로운 입력을 받을 LTG가 준비되지 않는다면,

그들을 상기 새로운 LTG의 출력을 무시할 것이다.

<322> **은닉 층에 LTG 추가:** 특히, 본 발명의 DR 훈련 알고리즘 (30)에 따라 NN의 은닉 층에 새로운 LTG를 할당하기 위한, 도 4의 과정 (50)을 참조. 이 논의는, 입력 벡터가 오직 하나의 출력을 지닌 NN에 처음에 적용되는 상황을 조사할 것이다. 제약은 첫 번째 은닉 층에 있는 첫 번째 LTG로 형성된다. 만일 그 층에 있는 첫 번째 LTG가 앞서 논의된 테스트에 의해 결정된 제약을 학습할 수 없다면 [도 3의 블록 (42)], 그 층에 있는 LTG들 중 하나가 그것을 학습할 때까지, 그 층에 있는 다음 LTG가 그것의 출력 등과 함께 입력 벡터로부터 형성된 제약을 학습하려고 한다. 하지만, 만일 그 층에 있는 LTG들 중 어느 것도 제약을 학습할 수 없다면, 추가 LTG를 상기 층에 추가해 그것을 학습해야만 한다 [블록 (47)]. 이는, LTG D를 층 N에 추가한 도 6b의 NN (70)에 도시되어있다.

<323> 예를 들어, 도 6b에 있는 LTG A 또는 LTG B가 새로운 입력 벡터를 학습할 수 없을 때, 도시된 대로 LTG D를 층 N에 추가한다. LTG D는 이 층, 층 N로의 입력을 기초로 한 새로운 제약을 학습한다 [블록 (55) 또는 (56)]. 또한, 이 새로운 LTG, LTG D의 출력은, 출력 O_j를 갖는 NN (70)의 출력 층, 층 N+1에 있는 출력 LTG, LTG C로의 입력 [블록 (53) 및 (54)]이 된다.

<324> 이제, 임의의 은닉 층이 고려될 것이다. 입력 벡터를 NN에 적용할 때, 첫 번째 은닉 층에 있는 각각의 LTG는 활성되어, 또는 LTG의 훈련에 의존하지 않고 반응할 것이다. 이러한 LTG의 반응들은 다음 층에 입력 벡터 역할을 하고, 이들은 그들의 훈련 등이 주어지면 반응할 것이다. 만일 은닉 층이, 입력 벡터의 결과로서, 받은 입력을 분류하는 것을 학습할 수 없다면 [블록 (42)], 새로운 LTG를 그 층에 추가한다 (참조: 도 4). 모든 입력 데이터가 학습될 때까지, 은닉 층에 LTG를 추가하는 과정을 반복한다.

<325> 설정된 은닉 층에 LTG를 동적으로 할당하는 과정 (50) 또는 바람직한 형식화 (formalization)는 다음 알고리즘에 주어진다:

<326> a) 층 N-1에 있는 모든 LTG들에 연결 형성 [블록 (52)]. 이러한 연결은 새롭게 할당된 LTG에 입력 역할을 한다.

<327> b) 학습된 출력에 대해, 층 N+1에 있는 모든 LTG들에 연결 형성 [블록 (53)]. 이러한 연결은 새롭게 할당된 LTG로부터의 출력 역할을 한다.

<328> c) 층 N+1에 있는 LTG는, 층 N에 있는 기존의 LTG와 새로운 LTG 사이의 논리 관계를 형성한다 [블록 (54)]; 및

<329> d) 층 N에 있는 다른 LTG가 학습한 것으로, 새롭게 할당된 LTG를 준비한다 [블록 (55) 내지 (56)].

<330> 상당히 복잡한 데이터로 훈련된 후 NN (100)이 만들어진 도 9에, 상기 연결 형성이 도시된다.

<331> LTG H를 층 N에 있는 NN (100)에 할당한다 [블록 (51)]. LTG H로부터의 출력 연결은, 출력 층이 아닌, 다음 은닉 층, 층 N+1에 있는 LTG F 및 G의 입력에 형성된다 [블록 (53)]. 이것은, 각각의 출력이 개별적인 학습 임무로 해결되는 것을 논의한 본 명에서 앞 부분에 나와있다. 입력 연결은 LTG들, 이전 층, 층 N-1에 있는 LTG A, B 및 D로부터 설정된다 [블록 (52)].

<332> 요약하자면, 만일 은닉 층 LTG의 어느 것도 그 층으로의 입력에 의해 형성된 제약을 학습할 수 없다면, 즉 해가 없다면, LTG를 은닉 층에 추가한다. 새로운 LTG는, O_j와 관련된 다음 층에 있는 모든 LTG들에 연결된 그것의 출력을 갖는다. 만일 상기 출력 LTG가 입력 제약을 학습할 수 없다면, 현재의 출력 층 LTG가 은닉 층이 되고, 도 5의 과정 (60)에 따라 새로운 출력을 NN에 출력으로서 추가한다.

<333> 지금껏, LTG를 상기 은닉 층에 추가하는 방법이 논의되었고, 이제 새로운 출력을 NN에 추가하는 방법을 조사할 것이다.

<334> **새로운 출력 추가:** 특히, 본 발명의 DR 훈련 알고리즘 (30)에 따라 NN에 출력 LTG를 할당하기 위한, 도 5의 과정 (60)을 참조. 출력 O_j를 선택한 후 [도 2의 블록 (32)], 앞서 설명한 것처럼, 모든 입력 소스 (source)를 단일 출력 LTG에 직접적으로 연결한다. 도 3을 참조하여 앞서 설명한 것처럼, 상기 LTG에 입력 벡터를 연속해서 적용하고 제약을 형성하여, 상기 단일 출력 LTG를 훈련시킨다. 도 10a (i)에, 현재 훈련되는 출력 O_j로 출력 층 (112)에 배열된, 단일 LTG, LTG A를 갖는 NN (110)의 개략도가 도시되어있다.

<335> 상기 입력 벡터가 형성한 각각의 제약으로 LTG A의 제약 세트에 있는 제약을 테스트한다 [블록 (42)].

사용된 테스트는 앞에서 제공하였다.

<336> 만일 새로운 제약이 기존의 제약 세트와 함께 해를 갖는다면, 그것을 상기 제약 제트에 추가한다. 하지만, 만일 해가 없다면 [블록 (42)에서], 도 10a(ii)에 도시된 것처럼, 다른 출력 층 (114)를 추가하고, 새로운 LTG, LTG C를 추가한다 [블록 (61)]. LTG C는 NN (110)의 새로운 출력 LTG, O_j가 된다. 도 10a(ii)에 도시된 것처럼, 입력 벡터를 학습할 수 없는 은닉 층 (112)[원래 출력 층 (112)]에 LTG, LTG A가 있었기 때문에, 새로운 LTG, LTG B를 은닉 층 (112)에 추가한다. 그때, LTG A가 학습할 수 없는 입력 벡터가 LTG B에 의해 학습될 수 있다. LTG A 및 LTG B의 출력이 출력 층 (114)[블록 (62)]에 있는 LTG C의 입력에 연결된다. 그때, LTG A 및 B가 NN (110)의 은닉 층 (112)을 형성한다.

<337> 다시, 만일 도 10b의 NN (110)의 LTG C가 일부 입력을 학습할 수 없다면, 새로운 은닉 층 (114)[이전에 출력 층 (114)]이 추가되고, 새로운 출력 층 (116)이 생성된다. 이러한 방법으로, 새로운 은닉 층이 생성되고, 출력 층이 추가된다. 도 10b를 참조하여, 새로운 은닉 층 LTG, LTG E를 은닉 층 (114)에 추가하고, 새로운 출력 LTG, LTG F를 새로운 출력 층 (116)에 추가한다.

<338> 요약하자면, 만일 출력 LTG가 상기 입력 벡터를 학습할 수 없다면, 다른 LTG를 현재 출력 층과 동일한 층에 추가하고, 모든 입력을 그것에 직접적으로 연결한다. 이러한 LTG는, 오래된 출력이 학습할 수 없는 입력을 학습한다. 추가 LTG를 다음 층에 추가한다. 이 LTG에 입력은 NN의 오래된 출력이고, 그 층에 새롭게 추가된 LTG이다.

<339> 이제, NN에 동적으로 할당된 LTG에 연결을 추가하는 방법이 설정되었는데, LTG를 어떻게 훈련시켜, NN이 계속해서 이전에 학습한 것을 재생산하는지를 조사하는 게 중요하다. 이것을 이제 논의할 것이다.

<340> **논리 학습:** 본 발명의 DR 훈련 알고리즘 (30)은 단일 패스 알고리즘이기 때문에, LTG를 NN에 추가할 때, NN이 이전에 학습된 입력 벡터에 대한 정확한 반응을 계속해서 생산하여야만 한다. 따라서, LTG의 추가로 인해, NN이 이전에 학습한 것을 잊지 않아야 한다. 이것은 다음과 같을 경우에 발생할 수 있다: (a) LTG를 은닉 층에 할당하는 경우; 또는 (b) 새로운 출력 층을 NN에 추가하는 경우. 이 경우, 새로운 LTG를 은닉 층에 할당한다.

<341> 이 문제를 피하기 위해: (a) 특정 논리 규칙에 따라, 상기 은닉 층에 새롭게 할당된 LTG를, 다른 LTG가 이 층에서 학습한 것으로 준비해야만 한다 [도 4의 블록 (55) 또는 (56)]; 및 (b) 새롭게 할당된 LTG가 할당된 층으로부터 직접적으로 입력을 받는 LTG의 층은, LTG가 새롭게 할당된 LTG의 원하는 작동을 기초로 학습하고 갱신된 것을 가질 필요가 있다 [블록 (54)]. 이 경우는 새로운 출력 층의 할당을 포함한다.

<342> 이제, 새롭게 할당된 LTG가 학습하는 것을 조사할 것이다. 우선, 도 10a(i)에 도시된 NN과 같은, 어떠한 은닉 층도 없는 NN을 조사할 것이다.

<343> LTG가 입력 벡터를 학습할 수 없을 때, 적어도 2가지 상황이 있다: (1) 학습되는 입력 벡터가 1을 출력할 필요가 있지만, 상기 LTG가 이전에 학습한 것을 기초로 이 입력 벡터에 대해 오직 0을 출력할 수 있을 때; 및 (2) 학습되는 입력 벡터가 0을 출력할 필요가 있지만, 상기 LTG가 이전에 학습한 것을 기초로 이 입력 벡터에 대해 오직 1을 출력할 수 있을 때.

<344> 앞서 논의된 것처럼, 이러한 상황에서, 도 10a(ii)에 도시된 것처럼, 새로운 출력을 NN에 할당한다.

<345> 그 다음 층이, LTG가 할당되는 은닉 층으로부터의 입력을 결합시킬 수 있는 가능한 방법에는 적어도 2가지가 있다: (1) 출력 LTG가 논리 (logical) OR로 은닉 층 LTG로부터 입력을 결합시키고 [블록 (64)]; 및 (2) 상기 출력 LTG가 논리 AND로 상기 은닉 층 LTG로부터의 입력 벡터를 결합시킨다 [블록 (63)].

<346> **OR 학습:** 특히, 도 5의 과정 (60)의 블록 (64) 참조. 우선, 오래된 LTG가 학습할 수 없는 입력 벡터에 대해 조사할 것이다. 만일 상기 벡터로 인해 NN이 1을 출력해야 하고, 상기 LTG가 이전에 학습한 것의 결과로서 오직 0을 출력할 수 있다면, 새로운 출력은 그것의 입력 사이에 OR을 형성할 필요가 있다

<347> 다시 도 10a(ii)를 참조하여, LTG A가 활성화될 때, NN (110)의 출력, LTG C가 활성화될 필요가 여전히 있지만, LTG A가 활성화될 필요가 있는 경우, 그렇게 될 수 없어, 그 결과 LTG B가 상기 입력에 있는 이러한 특징을 학습한다. 또한, LTG B는 NN (110)에 의해 이전에 학습된 입력 벡터를 학습할 필요가 있다. 이를 통해, LTG B가 상기 출력을 활성화시키지 않아야 할 때, 확실히 활성화되지 않게 한다. 이를 위해, LTG A의 제약 세트에 있는 모든 제약들을 LTG B의 제약 세트에 복사하지만, 모든 제약들은 <T로 학습된다. LTG B가, LTG A가 학

습할 수 없는 새로운 제약을 학습하고, 이 입력 벡터의 탐지에 의해 활성화될 것이다. 이로써, LTG C는, 그것의 2개의 입력 OR을 학습한 대로 활성화되고, 필요한 대로 1을 출력한다.

<348> **AND 학습:** 특히, 도 5의 과정 (60)의 블록 (63) 참조. 만일 출력이 0이 될 필요가 있고, LTG가 대신 1을 출력한다면, 새로운 출력이 LTG A 및 새롭게 할당된 LTG, LTG B로부터 AND 입력을 학습한다. 이 경우, $0<T$가 LTG A의 제약 세트에 있는 경우를 제외하고, 제약들을 LTG A의 제약 세트로부터 $\geq T$로 복사한다. 이 경우, 상기 제약을 되풀이해서 그대로 복사한다.

<349> LTG를 기존의 은닉 층에 더 할당할 경우, 앞서 설명된 대로, 그 층에 있는 이전 LTG (LTG B)로부터 그에 알맞게 상기 제약을 복사하고 변경한다. 하지만, 만일 LTG를 층 N에 추가한다면, 상기 LTG가 층 N+1에서 학습한 것은 변경이 필요하다.

<350> 그 다음 층이 학습하는 논리는, $Op_1...Op_N$이 논리 AND 또는 OR인 $(...(x_1\ Op_1\ x_2)Op_2\ x_3)Op_3\ x_4)...)$이고, $x_1...x_N$은, 새로운 LTG가 할당된 은닉 층으로부터 받은 입력이다. 만일 새롭게 할당된 LTG를 기존의 은닉 층에 할당하면, 이층으로부터 입력을 받는 LTG는 상기 논리를 기초로 그것의 제약이 갱신되도록 요구할 수 있다. 예를 들어, 만일 층이 기존의 논리 (x_1 AND x_2)를 갖는다면, 그것은 제약들 $\{w_{n1} + w_{n2} \geq T_n,\ w_{n1} < T_n,\ w_{n2} < T_n,\ 0 < T_n\}$를 가질 것이다. 만일 상기 논리가 ($x_1$ AND x_2) AND x_3이 된다면, 상기 제약 세트는 $\{w_{n1} + w_{n2} + w_{n3} \geq T_n,\ w_{n1} + w_{n3} < T_n,\ w_{n2} + w_{n3} < T_n,\ w_{n3} < T_n,\ w_{n1} + w_{n2} < T_n,\ w_{n1} < T_n,\ w_{n2} < T_n,\ 0 < T_n\}$이 된다.

<351> 입력 벡터가 학습될 수 있을 때 층에 있는 LTG에 의해 학습된 논리는, 그들이 NN에 추가되는 논리에 따른다. 만일 상기 LTG를 추가하여 AND를 형성한다면, 상기 LTG는 제약 $x_i.w \geq T$을 학습하고, 만일 상기 LTG를 추가하여 OR을 형성한다면, 상기 LTG는 $x_i.w < T$를 학습한다.

<352> **완전한 학습 및 일반화 설명**

<353> 이제, NN이 완전히 훈련되거나, 즉, NN이 그것이 학습한 것을 재생산할 수 있고, 또한 일반화할 수 있는 것을 설명할 것이다. 우선 LTG가, 그것이 학습한 입력을 회복시켜, 그것이 완전히 훈련되는 것을 설명한다.

<354> **LTG의 완전한 훈련:** 상기 LTG를 훈련시킬 때, 그 결과로 생긴 제약 세트를 LTG의 출력을 결정하는 데 사용할 수 있다. 상기 훈련된 LTG의 제약 세트에 입력을 적용하고, 맥컬럭-피트 LTG의 작동을 정의하는 방정식 1.1을 사용하여, 이것을 행한다. 이는 다음 예에 나와있다.

<355> 도 7에 도시된 대로, 다음의 제약 세트를 생산하기 위해 훈련된, 2-입력 LTG (82)를 고려: $\{w_1 + w_2 < T,\ w_2 > T\}$. 그런 후, 입력 벡터 [1 1]를 적용하면, $1.w_1 + 1.w_2 = w_1 + w_2 < T$이기 때문에, 상기 LTG는 0 출력을 생산할 것이다. 따라서, 가중치에 대한 숫자 값은, 상기 LTG가 완전히 훈련되는 데 필요하지 않다.

<356> 상기 논의는, LTG가 가중치 및 임계치에 대한 숫자 값을 찾지 않고 학습한 것을 재생산할 수 있다는 것을 증명한다. 또한 그것은, 상기 훈련된 NN에 의해 학습된 것이 100% 정확하게 회상될 수 있다는 것을 증명한다.

<357> 본 발명은 주로, 입력 벡터와 연관된 적당한 출력을 결정하기 위해 NN을 훈련시키는 것에 관한 것이지만, NN을 훈련시키기 위해 사용된 데이터 및 훈련 동안 문제를 일으킬 수 있는 2가지 문제에 대해서도 논의할 것이다.

<358> **일반화:** 본 발명의 바람직한 실시예는, 피드-포워드 NN이 간단한 데이터 세트로 훈련될 수 있어, 훈련 동안 NN에 의해 이전에 보이지 않은 데이터 패턴을 성공적으로 분류할 수 있기 때문에, 상기 피드-포워드 NN을 사용한다. 이것을 일반화라고 부른다.

<359> 데이터 공간에 대해 알려진 것이 거의 없는, 블랙 박스 (black box) NN 분류 시스템을 갖는 것이 바람직할 수도 있지만, NN을 훈련시킬 때 매우 중요한, 적어도 2개의 데이터의 양상 있고, 이들은 다음과 같이 열거된다: (1) 큰 노이지 (noisy) 데이터와 직면한 문제 중 하나는, 그것이 모순 (contradiction)을 가질 수 있다는 것인데, 예를 들어, 일부 입력 벡터 x_i가 있고, 만일 일 실시예에서 $x_i \rightarrow 0$이고, 다른 실시예에서, $x_i \rightarrow 1$이라면, NN은 이 벡터를 학습하는 데 어려움을 겪을 것이다. 이 문제는 모든 학습 알고리즘에 일반적이고; 및 (2) NN을 훈련시키기 위해 사용된 훈련 샘플을 확보하는 것이 상기 데이터 세트를 나타내는 것이다. 이제, 이것을 다음과 같이 더 자세히 설명할 것이다.

<360> 각각의 데이터 세트는 그것에 몇 개의 특징들을 갖는다. 훈련 동안 NN이 노출된 데이터 세트는, NN을 완전히 훈련시키는 데 필요한 모든 특징들을 나타낸다. 하지만, 훈련 세트가, 상기 데이터 세트가 크고 거의 이해되지 못할 때 완전한 데이터 세트에 있는 모든 특징들을 나타내는 것인지를 결정할 방법은 없다. 이 경우, 상기 데이터 세트를 '미지 (unknown)'라고 부른다.

<361> 훈련된 NN을 테스트하여, 만일 NN이 상기 데이터 세트에 있는 모든 특징들을 학습하였는지 결정하는 것은 불가능하다. 추가 분류된 입력 벡터로 NN을 테스트하는 것이 이를 달성하는 데 바람직한 방법이다. 피드-포워드 NN을 훈련시키기 위해 데이터 세트를 분류하는 것은 앞서 논의되었다. 하지만, 상기 데이터 세트가 잘 이해되지 않고 크다면, 상기 데이터 세트에 있는 다른 특징들은 더욱더 분명해지지 않을 수 있다.

<362> 이제, 연역 (deduction)이 항상 작용하지 않을 이유에 대해 논의할 것이다.

<363> 연역은, 상기 훈련 데이터 세트에 실종 (missing)된 특징들이 있을 때 실패 (fail)할 것이고, 이 문제는 '불충분한 훈련'이라고 불릴 수 있다.

<364> **LTG의 불충분한 훈련:** LTG를 훈련시키는 각각의 데이터 세트는 몇 개의 데이터 특징들을 가질 수 있다. 훈련 데이터는 그것 내의 나타난 이러한 특징들 몇 개를 가질 수 있으나; 미지의 데이터 세트 내에 존재하는 모든 특징들 또한 상기 훈련 세트에 나타난다는 것을 보장하지는 않는다.

<365> 따라서, 만일 상기 훈련 데이터 세트 내에 나타나지 않은 특징들이 있다면, 상기 LTG는 훈련 동안 상기 데이터 세트에 있는 모든 특징들에 노출되지 않는다. 따라서, 상기 LTG를 보이지 않는 입력 벡터로 테스트될 때, 그것은 잘못된 결과를 출력할 수 있다. 따라서, 상기 LTG를 불충분하게 훈련된 것으로 부를 수 있는 것이다.

<366> 예를 들어, 오직 2개의 입력 벡터, [0 0]→0 및 [0 1]→1로 훈련된, 도 7에 도시된 2-입력 LTG (82)를 고려한다.

<367> LTG (82)는 아직, 상기 입력 벡터에 있는 첫 번째 비트가 세트가 되는 어느 벡터에 노출되지 않기 때문에, 그것은 [1 0] 및 [1 1]을 정확히 분류할 수 없다. 이러한 LTG (82)가 상기 훈련 세트로부터 학습하는 제약 세트는 $\{0<T, w_2 \geq T\}$이고, 비록 w_2에 대한 일부 정보가 있지만, w_1과의 어떠한 관계도 설정되지 않았다. 예를 들어, $w_1 + w_2$ 또는 w_1이 T와 갖는 관계가 무엇인지 알려지지 않는다. 그 결과, 입력 벡터 [1 1] 및 [1 0]에 대한 출력을 연역하는 것은 불가능할 수 있다. 상기 입력 사이에 형성된 논리 관계 면에서, 이들은 x_2, x_1 OR x_2, 또는 x_1 XOR x_2일 수 있지만, 또 다른 정보 없이 그것이 무엇인지 말하는 것은 불가능하다.

<368> 본 발명의 DR 훈련 알고리즘 (30)에 따라, 만일 LTG가 불충분하게 훈련됐다면, 그것은 1을 출력할 것이 바람직하지만, 이것은 상기 데이터 세트에 따라 달라, 대신 0을 출력할 수 있다. 즉, 상기 LTG는, 그것이 상기 데이터 입력에 반응하는 방법을 학습할 때까지, 활성으로 남을 것이다. 하지만, 이것은 NN에 의해 학습되는 데이터 세트에 따라 변화될 것이다.

<369> 따라서, 상기 훈련 세트에 실종된 특징들이 있다면, LTG는 정확한 반응을 출력하는 데 어려움을 가질 수 있다는 것이 이해되어야만 한다. 이것은 또한, 역전파와 같은 다른 훈련 방법으로 훈련된 다른 뉴런이 경험하는 문제이다. 하지만, 역전파로 훈련된 퍼셉트론과 달리, 본 발명의 DR 훈련 알고리즘 (30)이, 상기 LTG가 데이터 패턴을 분류하는 방법을 아직 학습하지 못한 때를 확인하는 것은 가능하다. 즉, 본 발명의 DR 훈련 알고리즘 (30)으로 훈련된 LTG는, 그것이 입력 벡터를 정확히 분류하는 방법을 알지 못한 때를 나타낼 수 있다.

<370> 이제, LTG의 가장 유용한 특성 중 하나, 즉 충분한 훈련이 주어진다면, 보이지 않는 입력 벡터의 출력을 연역할 수 있는 상기 LTG의 능력에 대해 논의할 것이다.

<371> **보이지 않는 입력 벡터 연역:** NN을 훈련시킬 때, 그것을 입력 벡터의 세트로 훈련한 후, NN이 훈련 동안 노출되지 않은 많은 입력 벡터로 테스트한다. 이러한 입력 벡터는 '보이지 않는'으로 불리고, NN이 그들의 연관 출력을 정확히 결정할 수 있는지 결정한다.

<372> NN이, 훈련 동안 보이지 않았던 데이터 패턴의 분류를 결정할 수 있기 위해, NN은 이전에 학습한 것으로부터 보이지 않는 입력 벡터의 분류를 연역할 수 있어야 한다. 모든 가능한 입력 벡터 및 그들의 연관 출력에 상기 LTG를 노출시키는 것을 제외하고, 모든 특징들이 훈련 동안 학습된다는 보장이 없기 때문에, 상기 출력을 정확히 연역하는 것은 불가능할 수 있다. 종종, 모든 데이터 패턴 또는 입력 벡터가 이용가능한 것이 아니고, 만일 그들이 상기 입력 데이터 세트를 완전히 나열 (enumerate)하여 찾아지더라도, 그들을 분류하는 방법은 알

려지지 않을 수 있다. 만일 오직 입력 데이터 세트의 일부인 훈련 세트가, NN이 학습하기 위해 훈련되는 그 데이터 세트를 나타내는 것인지 결정하는 방법이 있을 것 같지 않다. 그 결과, 특정 데이터 세트로 훈련하여, 보이지 않는 데이터 패턴의 출력이 일부 경우 NN에 의해 정확히 결정될 수 있다는 것을 보여주는 것만이 가능하다.

<373> **명제 (Proposition) 3.1:** 만일 보이지 않는 입력 벡터에 대한 LTG의 임계치와의 관계가, 그것을 이전에 학습한 제약으로부터 연역될 수 있다면, 그것은 보이지 않는 입력 벡터에 대한 출력을 결정할 수 있을 것이다.

<374> 도 11에 도시된 LTG (120) 고려. 상기 LTG는 다음 입력 및 출력 벡터를 사용하여 훈련된다: [0 0 0]→0; [0 0 1]→1; [0 1 0]→1; 및 [1 1 1]→0.

<375> 그때, LTG (120)는 다음의 제약 세트를 가질 것이다: $\{0<T, w_3 \geq T, w_2 \geq T, w_1 + w_2 + w_3 < T\}$.

<376> 상기 입력 벡터 [1 0 0]은 훈련 동안 LTG (120)에 의해 보이지 않고, 이 벡터에 필요한 출력은 0이다. 만일 LTG (120)가 상기 임계치와의 관계를 연역할 수 없다면, 상기 출력이 0인 것을 결정할 수 없을 것이다.

<377> $0<T$, $w_2 \geq T$ 및 $w_3 \geq T$이기 때문에, T, w_2 및 w_3은 모두 양수이고, w_2 및 $w_3 \geq T$이다. 따라서, $w_2 + w_3$은 또한 $\geq T$이어야만 한다. 하지만, $w_1 + w_2 + w_3 < T$는, w_1이 작고 음수이고 따라서 $<T$인 것을 암시한다. 따라서, LTG (120)에 적용되고, LTG, $1.w_1 + 0.w_2 + 0.w_3 = w_1 < T$의 작동을 정의하는 방정식 1.1을 사용할 때, 입력 벡터 [1 0 0]가 연역된다. 따라서, 상기 LTG는 0을 출력할 것이다.

<378> 그러므로, LTG (120)는 정확한 출력을 연역할 수 있다. LTG (120)가 상기 정확한 출력을 유도할 수 있기 때문에, 그것이 충분히 훈련된다면, 출력을 연역할 수 있다는 것을 보여준다.

<379> 본 발명의 DR 훈련 알고리즘 (30)은 LTG를 사용하여 NN을 구성하는 것이 바람직함에 따라, 연역 원리를 본 발명에 따라 사용하여, NN의 보이지 않는 입력 벡터의 분류를 연역할 수 있는 것이 바람직하다. 대안적으로, 훈련 동안 보이지 않았던 데이터 패턴의 분류를 결정하는 다른 방법 또한 본 발명의 다른 양태에 따라 사용할 수 있다. 이제, 데이터 패턴의 분류를 결정하거나, 또는 제약 세트의 입력 벡터가 드러나는지 드러나지 않는지를 결정하는 이 대체 방법을 설명할 것이다.

<380> 이제, 입력 벡터가 명백히 학습됐는지와 상관없이, NN이 입력 벡터를 분류하는 방법을 학습했는지를 결정하기 위해 신규한 테스트가 제공될 것이다. 다음 테스트는, 상기 NN 출력이 입력 벡터에 대해 드러나는지를 보여준다.

<381> **LTG가 입력 벡터를 아는지 테스트:** 훈련된 LTG로부터 입력 벡터, 또는 패턴의 분류를 찾을 수 있는 것이 바람직하다. 입력 벡터가 훈련된 LTG에 적용될 때, 그것은 다음 중 하나를 할 것이다: (1) 활성화; (20) 활성화 실패; 또는 (3) 불충분한 훈련의 결과로, 입력 벡터를 분류하는 방법을 알 수 없음.

<382> 종래의 훈련 알고리즘은, LTG가 입력 벡터를 분류하는 방법을 알지 못할 때의 상황을 확인하도록 하는데 실패한다. 본 발명의 DR 훈련 알고리즘 (30)은, 상기 LTG가 분류 방법을 알지 못하는 입력 벡터의 확인을 허용한다.

<383> 이제, 본 발명의 또 다른 양태에 따라, 제약 세트의 입력 벡터가 드러나는지 드러나지 않는지 결정하는 방법 (130)의 바람직한 실시예를 도 12의 흐름도를 참조하여 설명할 것이다. 상기 제약 세트가, 본 발명의 DR 훈련 알고리즘 (30)에 따라 훈련된 NN 뉴런의 제약 세트인 것이 바람직하다. 입력 벡터가 드러나는지 드러나지 않는지 결정하는 방법 (130)은 NN에 한정되지 않는다는 것이 이해되어야만 한다. 또한, 입력 벡터를 분류하는 방법 (130)은, 예를 들어, DNA와 같은, 데이터의 문자열 (string) 분석과 같은 제약 시스템을 사용하는 다른 분야에 유용할 수 있다고 여겨진다. 마찬가지로, 입력 벡터를 분류하는 방법 (130)은 또한, CSP 및 운영 연구 애플리케이션 (operational research application)에 사용될 수 있다. 따라서, 본 발명의 이러한 양태는 독립적이고, 본 발명의 DR 훈련 알고리즘 (30)으로 사용하는 데 한정되지 않는다.

<384> 이제, 입력 벡터를 분류하는 방법 (130)을, 본 발명의 DR 훈련 알고리즘 (30)에 따라 훈련된 LTG의 출력을 결정하는 면에서 설명할 것이다. 이러한 설명은 단지, 본 발명에 따른 방법 (130)의 가능한 사용 중 하나인 예에 불과하다.

<385> LTG가 불충분하게 훈련됐는지, 즉, 입력 벡터, x_i를 분류하는 방법을 알지 못하는지를 결정하기 위해, 앞서 설명한 것처럼, 우선 블록 (131)에서, 제약 및 그것의 보수를 상기 입력 벡터로부터 구성한다. 형성된 제

약들은 다음과 같다: $x_i.w < T$ 및 그것의 보수 $x_i.w \geq T$, 또는 $x_i.w \geq T$ 및 그것의 보수 $x_i.w < T$.

<386> 이 입력 벡터와 연관된 출력은 아직 드러나지 않은 것으로 가정한다. 상기 제약 $x_i.w < T$ 또는 $x_i.w \geq T$를 훈련된 LTG 제약 세트에 추가한 후, 해가 있는지 결정하기 위해 블록 (132)에서 적당한 제약 만족 알고리즘을 사용하여 테스트한다 (수치 해가 찾아질 수 있지만, 특정한 해를 찾는 것은 중요하지 않고, 또는 동등하게 (equivalently) 제약들에 의해 정의된 체적의 교차 구역이 도 12에 도시된 대로 찾아질 수 있다). 만일 해가 없다면, 상기 LTG는, 블록 (133)으로 나타낸 0 또는 1을 출력해야만 하고, 상기 LTG는 충분히 훈련되고, 이 입력 벡터를 분류하는 방법을 안다. 즉, $x_i.w \geq T$ 또는 $x_i.w < T$에 대한 해가 있어야만 한다.

<387> 하지만, 블록 (132)에서, 만일 제약 $x_i.w < T$ 또는 $x_i.w \geq T$를 훈련된 LTG의 제약 세트에 추가했을 때, 해가 있었다면, 블록 (134)에서, 제약 $x_i.w < T$ 또는 $x_i.w \geq T$를 제거하고, 그것의 보수를 대신 추가한다. 만일 블록 (135)에서 점검을 수행할 때 해가 없다면, 상기 LTG는 이 입력 벡터를 분류하는 방법을 알고, 블록 (136)으로 나타낸 대로 0 또는 1을 출력할 것이다.

<388> 하지만, 만일 블록 (135)에서 점검을 수행할 경우, 상기 제약 및 그것의 보수를 블록 (134)에서 대안적으로 추가했을 때 상기 LTG가 해를 가졌다면, 블록 (137)에 나타낸 대로, 상기 입력 벡터가 불충분하게 훈련됨에 따라, 상기 입력 벡터가 분류되는 방법은 알려지지 않는다. 이러한 단계의 순서는 중요하지 않다는 것이 이해되어야만 한다.

<389> 적당한 제약 만족 방법 또는 알고리즘을 사용하여, 상기 제약이 학습될 수 있는지 테스트할 수 있다. 가중치 및 임계치 값에 대한 특정 수치 해를 찾는 것은 중요하지 않지만, 그들이 찾아질 수 있는지 결정하는 것은 중요하다. 이것은 상기 제약에 의해 정의된 체적에서 교차 구역을 찾는 것과 동일하게 언급될 수 있다.

<390> 상기 입력 벡터를 제약으로 변환할 때, 그것은 상기 LTG의 가중치-공간에서 평면 (plane)을 형성한다. 입력 벡터가 상기 LTG에 의해 학습될 때마다, 그것은, 상기 훈련 조건을 만족시키는 체적을 줄이며, 상기 가중치-공간을 양분 (bisect)하는 평면을 형성한다. 이것은 도 13 (a)에 도시되어 있는데, 여기서 밀폐된 오목 영역 (enclosed concave region)은, 지금까지 학습된 훈련 조건을 만족시키는 가중치-공간이다. 상기 영역을 양분하는 평면은, 상기 LTG에 제공되는 입력 벡터로부터 형성된다. 이러한 상황에서, 도 13 (b) 및 13 (c)에 각각 도시된 대로, 상기 LTG는 $x_i.w < T$ 또는 $x_i.w \geq T$를 학습할 수 있다. 이 경우, 상기 입력 벡터를 분류하는 방법은 알려지지 않는다. 도 13 (d)에서, 상기 LTG는 상기 평면 아래가 아닌 위의 영역만을 학습할 수 있으므로, 따라서 상기 훈련 조건을 만족시키는 가중치-공간의 체적과 교차하는 제약에 의해 상기 출력이 결정될 것이다.

<391> 도 13(e)에서, $x_i.w = T$에 의해 형성된 평면은 상기 볼록 (convex) 체적과 교차하지만, 도 13(f)에서는 그 평면으로 형성된 오직 하나의 제약만이 상기 LTG에 의해 학습될 수 있다. 이러한 모든 제약들의 교차 구역에 의해 형성된 영역에만 관심 (interest)이 주어지기 때문에, 상기 볼록 영역은 오목 (concave) 영역으로 줄어든다.

<392> 만일 상기 입력 벡터가 상기 훈련 세트에 있게 된다면, 그것은, 훈련 동안 구성되었던 제약 세트에 의해 정의된 체적의 면 중 하나를 형성할 것이고, 따라서, 상기 입력 벡터가 알려지게 될 것이다.

<393> 요약하자면, 상기 제약 및 그것의 보수 모두 상기 입력 벡터로부터 형성되고 [블록 (131) 및 (134)], 교차 구역의 존재에 대해 훈련된 LTG의 제약 세트로 훈련된다 [블록 (132) 및 (135)]. 만일 상기 제약들 중 어느 것도 해를 이끌어낼 수 없다면 [블록 (133) 및 (136)], 이 입력 벡터에 있는 특징들이 훈련 동안 학습되었다는 것을 암시한다. 하지만, 만일 상기 LTG가 이미 학습한 것과 함께 제약에 이용가능한 해가 있다면 [블록 (137)], 상기 훈련 세트로부터 실종된 특징들이 있다. 상기 특성은 다음과 같이 공식적으로 진술될 수 있다:

<394> **정리 (Theorem):** 상기 LTG가 학습한 제약 목록에 $x_i.w < T$ 또는 $x_i.w \geq T$ 및 그것의 보수를 교대로 추가한 [블록 (131) 및 (134)] 후, 교차 구역에 대해 테스트하여 [블록 (132) 및 (135)], 상기 벡터가 x_i가 학습되는지를 결정할 수 있다. 만일 두 가지 경우에 해가 있다면 [블록 (137)], 상기 제약은 학습되지 않는다. 하지만, 만일 오직 $x_i.w < T$ 또는 그것의 보수가 이전에 학습된 제약으로 해를 갖는다면, 이 벡터는 상기 LTG에 의해 학습된다 [블록 (133) 및 (136)].

<395> **증명 (Proof):** 도 13에서, 상기 LTG가 학습한 것에 의해 정의된 체적의 2개의 표상 (representation)이 주어진 가중치-공간의 표에 도시되어 있다. 오목 영역은 도 13(a) 내지 13(d)에 도시돼 있고, 볼록 영역은

도 13(e) 및 13(f)에 도시돼 있다. 상기 벡터에 의해 형성된 평면을 상기 가중치-공간, 즉 $x_i.w = T$에 적용한다. 그것은 도 13(d)에 있는 것처럼, 정의된 체적과 교차하지 않거나, 모든 다른 경우에 있는 것처럼, 교차할 것이다. 만일 그것이 교차하지 않는다면, 상기 입력 벡터는 학습된 것이다. 이 경우, 체적, 즉 $x_i.w < T$ 또는 $x_i.w \geq T$가, 상기 LTG가 이미 학습한 제약에 의해 형성된 체적과 교차하는 것에 따라, LTG를 활성화시키거나 활성화시키지 않을 것이다. 그렇지 않으면, $x_i.w$는 학습되지 않는다.

<396> 상기 평면이 도 13 (e) 및 13 (f)에서와 같은 볼록 영역과 교차하는 경우, 상기 영역은, 상기 LTG가 이전에 학습한 모든 제약에 공통적임에 틀림없기 때문에, 이들 중 오직 하나만이 학습될 수 있다 [주의: 관심이 오직 두 영역의 공통 영역, 즉 그들의 교차 구역에만 주어지기 때문에, 도 13 (e) 및 13 (f)에 있는 영역은 오목 영역으로 줄어든다]. 이제, 이렇게 되는 것을 증명하기 위해, 일 예가 주어질 것이다.

<397> 도 11에 주어진 3-입력 LTG (120)을 고려. 만일 LTG (120)가 다음의 입력 벡터, [0 1 0]→1 및 [0 1 1]→0으로 훈련된다면, 상기 LTG (120)가 학습한 제약 세트는 $\{w_2 \geq T, w_2 + w_3 < T\}$이다. 벡터 [1 0 0] 및 [0 0 1]에 대한 출력이 결정되도록 한다.

<398> 상기 입력 벡터 [1 0 0]에 대해, 평면 $w_1 = T$가, 상기 영역 $\{w_2 \geq T, w_2 + w_3 < T\}$과 교차하는 것으로 발견되어, 그 결과 $w_1 < T$ 및 $w_1 \geq T$가 LTG (120)에 의해 학습된 영역과 교차한다. 따라서, LTG (120)은, 상기 출력이 되어야만 하는 것을 알지 못한다. 앞에서, 상기 출력이 1이어야만 한다는 것이 진술되었지만, 이것은 필요시, 학습되는 데이터 세트에 따라 변경될 수 있다.

<399> 상기 입력 벡터 [0 0 1]에 대해, 평면 $w_3 = T$가, 상기 영역 $\{w_2 \geq T, w_2 + w_3 < T\}$과 교차하지 않는 것으로 발견되고, 그렇게 하기 위한 유일한 영역은 $w_3 < T$이다. 따라서, 상기 벡터 [0 0 1]에 대한 출력은 0이 될 것이라는 게 알려진다.

<400> 상기 입력 공간에 대해 그렇게 많이 알려지지 않지만, 본 발명에 따라 LTG를 훈련시키는 DR 훈련 알고리즘 (30)은 상기 가중치-공간에 관한 많은 정보를 제공한다.

<401> 상기 제약 세트에 각각의 제약을 추가하여, 이 LTG에 대한 모든 훈련 조건을 만족시키는 가중치-공간 영역을 감소시킨다.

<402> 이제, 입력 벡터의 출력을 결정하는 것은, 그것이 임계치로 형성할 수 있는 가능한 2개의 제약을, 상기 LTG가 학습한 제약과 비교하는 것이 바람직한 방법인 것으로 이해되어야만 한다. 그것이 상기 LTG를 활성화시키거나 활성화시키지 않거나, 또는 정확한 출력을 알지 못할 것이다.

<403> 지금껏 NN을 훈련시키고 테스트하여, 보이지 않는 입력을 연역하는 방법이 설명되었고, 이제, 본 발명의 DR 훈련 알고리즘 (30) 및 일반화를 증명하기 위해 완전히 작용하는 예가 주어질 것이다.

<404> **모듈로-8 문제가 주어진 DR 훈련 알고리즘의 사용 예**

<405> 본 발명의 DR 훈련 알고리즘 (30)의 상세한 설명은 다음의 논의에서 예시된다. 이 예에서, 모듈로-8 문제를 해결하는 NN (140)의 바람직한 실시예가 사용된다. 데이터 세트는 이진수의 3차원 입력 벡터를 포함하고, 출력은 순서대로 다음 이진수이다. 입력 벡터 [1 0 1]은 무작위로 선택되고, 테스트를 위해 보존된다. NN (140)을 훈련시키는 제약 세트를 만든 후, NN이 [1 1 0]인 입력 벡터 [1 0 1]에 대한 출력을 연역할 수 있다는 것을 볼 수 있다.

<406> 상기 데이터 세트가 표 1에 나와있고, 다음의 입력 및 연관된 출력을 갖는다.

표 1

입력 벡터			출력 벡터		
X_1	x_2	x_3	O_1	O_2	O_3
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

<407>

<408> **모듈로-8을 정의하는 입력 및 출력 벡터**

<409> **제약 세트 구성:** 상기 입력 벡터는 [x_1 x_2 x_3]으로 정의되고, 상기 출력 벡터는 [0_1 0_2 0_3]으로 정의된다. 우선, 상기 출력 벡터에 있는 첫 번째 출력을 선택하여 처음에 훈련시킨다 [도 2의 블록 (32)]. LTG는 그들의 임계치가 갖는 첨자 (subscript)로 나타낼 것이다. 예를 들어, LTG_{11}은 임계치 T_{11}를 갖는다. 우선, 이 경우 0, [0 0 0] 벡터가 학습되도록 이용가능한지 결정한다. 그래서, 상기 벡터 $\{0<T_{11}\}$를 제약 세트에 추가한다. 상기 입력 벡터를 정렬하여 [블록 (33)], 그 결과 현재 훈련되는 출력 벡터의 위치에서 1을 출력하는 벡터들을 먼저 학습한다. 도 14a 참조.

<410> 도 3의 과정 (40)에 따라, 출력 0_1에 대한 훈련으로 인해, LTG_{11}이 다음과 같이 정의된다: $\{0 < T_{11}, w_{113} < T_{11}, w_{112} < T_{11}, w_{112} + w_{113} \geq T_{11}, w_{111} \geq T_{11}, w_{111} + w_{112} \geq T_{11}\}$.

<411> 이것은, 블록 (42)에서 점검할 때 해를 갖는다. 하지만, 다음 제약의 추가는 해를 갖지 않는다: $w_{111} + w_{112} + w_{113} < T_{11}$.

<412> 따라서, LTG_{11}은, 입력 벡터 [1 1 1]에 대해 0 대신 1을 출력할 것이다. 새로운 LTG, LTG_{12}를 도 4의 과정 (50)에 따라 NN (140)에 추가하여, 상기 입력 벡터 [1 1 1]를 학습한다. NN (140)의 새로운 위상 (topology)에 대해서는 도 14b 참조. 제약 $\{w_{121} + w_{122} + w_{123} < T_{12}\}$을 LTG_{12}의 제약 세트에 추가한다 [블록 (55)]. LTG_{11}이 학습한 정보를 되풀이해서 복사하고 변경하여, LTG_{12}의 제약 세트가 다음처럼 된다: $\{w_{121} + w_{122} + w_{123} < T_{12}, 0 \geq T_{12}, w_{123} \geq T_{12}, w_{122} \geq T_{12}, w_{122} + w_{123} \geq T_{12}, w_{121} \geq T_{12}, w_{121} + w_{122} \geq T_{12}\}$.

<413> LTG_{11}이 현재 출력이기 때문에 [블록 (48) 점검], 새로운 출력 LTG, LTG_{21}을 도 5의 과정 (60)에 따라 NN (140)에 추가한다. NN (140)의 새로운 위상에 대해서는 도 14c 참조. 상기 출력이 0 대신 1이었기 때문에, 이것은, 상기 새로운 출력 LTG_{21}이 LTG_{11}과 LTG_{12} 사이의 AND를 형성할 것을 의미한다 [블록 (63) 참조].

<414> LTG_{21}이 그것의 입력 사이에 AND를 형성하기 때문에, 그것의 제약 세트는 다음처럼 된다: $\{0 < T_{21}, w_{211} < T_{21}, w_{211} + w_{212} \geq T_{21}, w_{212} < T_{21}\}$.

<415> 첫 번째 출력, 0_1을 학습하는 데 필요한 3개의 LTG들에 대한 제약 세트는 다음과 같다: **LTG_{11}**: $\{0 < T_{11}, w_{113} < T_{11}, w_{112} < T_{11}, w_{112} + w_{113} \geq T_{11}, w_{111} \geq T_{11}, w_{111} + w_{112} \geq T_{11}, w_{111} + w_{112} + w_{113} \geq T_{11}\}$; **$LTG_{12}$**: $\{w_{121} + w_{122} + w_{123} < T_{12}, 0 \geq T_{12}, w_{123} \geq T_{12}, w_{122} \geq T_{12}, w_{122} + w_{123} \geq T_{12}, w_{121} \geq T_{12}, w_{121} + w_{122} \geq T_{12}\}$; 및

LTG$_{21}$: {$w_{211} < T_{21}$, $w_{211} + w_{212} \geq T_{21}$, $w_{212} < T_{21}$, $0 < T_{21}$}.

<416> 이제, 출력, 0_1에 대한 출력이 훈련되었고, 상기 과정을 계속하여, 출력 0_2를 훈련시킨다 [블록 (35)에서 점검한 후, 블록 (32)로 되돌아감]. 상기 데이터를 표 2에 나열된 대로 정렬한다 [블록 (33)]. 입력 벡터 [1 0 1]가 테스트 목적을 위해 남겨졌다는 점이 중요하다. 도 14d는 기존의 NN (140)과 함께, 0_2에 대한 초기 출력 LTG, LTG$_{13}$의 개략도를 제공한다.

표 2

입력			출력
x_1	x_2	x_3	O_2
0	0	0	0
0	0	1	1
0	1	0	1
1	0	1	1
0	1	1	0
1	0	0	0
1	1	1	0

<417>

<418> **모듈로-8 데이터 세트에 대한 두 번째 출력**

<419> LTG$_{13}$은 제약 {$0 < T_{13}$, $w_{133} \geq T_{13}$, $w_{132} \geq T_{13}$, $w_{131} + w_{123} \geq T_{13}$}을 학습한다 [블록 (34)].

<420> 하지만, 입력 벡터 [0 1 1]에 대한 제약을 상기 제약 세트에 추가함으로써, $w_{133} + w_{132} < T_{13}$은, 도 3의 과정 (40)의 블록 (42)에서 점검할 때, 어떠한 해도 갖지 않는다. 새로운 LTG, LTG$_{14}$를 NN (140)에 할당해, 도 4에 도시된 과정 (50)에 따라 LTG$_{13}$으로 은닉 층 (142)를 형성한다. LTG$_{14}$는 입력 벡터 [0 1 1]에 대해 제약 $w_{143} + w_{142} < T_{14}$를 학습할 것이다 [블록 (55)]. NN (140)의 새로운 위상에 대해서는 도 14e 참조.

<421> LTG$_{13}$이 입력 벡터 [0 1 1]에 대해 필요한 0 대신 1을 출력하기 때문에, 출력 LTG가 그것의 입력 사이에 AND를 형성하여야만 한다 [블록 (54)].

<422> 따라서, 새로운 LTG$_{14}$를 추가해 이 조건을 학습하고, LTG$_{14}$가 상기 입력 벡터 [0 1 1]을 학습한다. **LTG$_{14}$**: {$0 < T_{14}$, $w_{143} + w_{143} < T_{14}$, $w_{143} \geq T_{14}$, $w_{142} \geq T_{14}$}.

<423> 다시, 도 5의 과정 (60)에 따라, 새로운 출력 LTG, LTG$_{22}$를 출력 0_2에 대해 추가하고, AND를 사용하여 그것의 입력을 결합시키는 것을 학습하여, 그 결과 다음의 제약 세트를 생산한다: {$0 < T_{22}$, $w_{223} < T_{22}$, $w_{223} + w_{224} \geq T_{22}$, $w_{224} < T_{14}$}. NN (140)의 새로운 위상의 개략도에 대해서는 도 14f 참조.

<424> LTG$_{13}$은 다음 벡터 [1 0 0]을 학습하고, 그것의 제약 세트는 다음과 같이 된다: **LTG$_{13}$**: {$0 < T_{13}$, $w_{133} \geq T_{13}$, $w_{132} \geq T_{13}$, $w_{131} < T_{13}$, $w_{131} + w_{132} \geq T_{13}$}. LTG$_{14}$에 대한 제약 세트는 다음과 같이 된다: **LTG$_{14}$**: {$w_{143} + w_{143} < T_{14}$, $w_{143} \geq T_{14}$, $w_{142} \geq T_{14}$, $0 \geq T_{14}$, $w_{141} \geq T_{14}$}.

<425> 마지막 벡터 [1 1 1]이 형성한 제약은 LTG$_{13}$에 의해서 학습될 수 없지만, LTG$_{14}$에 의해 학습될 수 있고, 따라서 모든 3개 LTG에 대한 마지막 제약 세트는 다음과 같다: **LTG$_{13}$**: {$0 < T_{13}$, $w_{133} \geq T_{13}$, $w_{132} \geq T_{13}$, $w_{131} <$

T_{13}, $w_{131} + w_{132} \geq T_{13}$, $w_{131} + w_{132} + w_{133} \geq T_{13}$, $w_{133} + w_{132} \geq T_{13}$}; **$LTG_{14}$**: {$0 \geq T_{14}$, $w_{143} + w_{142} < T_{14}$, $w_{143} + w_{142} + w_{141} < T_{14}$, $w_{143} \geq T_{14}$, $w_{142} \geq T_{14}$, $w_{141} \geq T_{14}$, $w_{141} + w_{142} \geq T_{14}$}; 및 **$LTG_{22}$**: {$0 < T_{22}$, $w_{223} < T_{22}$, $w_{223} + w_{224} \geq T_{22}$, $w_{224} \geq T_{22}$}.

<426> 이제, 두 번째 출력이 훈련되었고, 마지막 출력, 0_3을 훈련시켜야만 한다 [블록 (35)에서 점검한 후, 블록 (32)로 되돌아감]. 처음에, 블록 (34)[도 3의 과정 (40)을 사용]에서, LTG_{15}가 다음의 제약 세트를 학습하는 것이 발견된다: **LTG_{15}**: {$0 \geq T_{15}$, $w_{153} \geq T_{15}$, $w_{152} \geq T_{15}$, $w_{153} + w_{152} < T_{15}$, $w_{151} \geq T_{15}$, $w_{151} + w_{152} + w_{153} < T_{15}$, $w_{152} + w_{151} \geq T_{15}$}. 도 14g는 기존의 NN (140)과 함께, 0_3에 대한 처음 출력 LTG, LTG_{15}의 개략도를 도시한다.

<427> LTG_{15}가 완전한 훈련 세트에 노출되고, 블록 (42)에서 점검할 때 해를 갖고 있어, 어떠한 새로운 LTG도 NN (140)에 추가할 필요가 없고, 따라서 도 14g는, 모듈로-8 데이터 세트를 학습한, 최종적으로 완전히 훈련된 NN (140)의 개략도가 된다.

<428> 이제, 도 12에 도시된 대로, 입력 벡터가 드러나는지 드러나지 않는지 결정하는 방법 (130)에 따라 보이지 않는 입력을 연역하는 방법을 조사할 것이다.

<429> **테스트 입력 벡터에 대한 출력 연역:** 이제, 보이지 않는 입력 벡터가 [1 0 1]이고 그것의 연관 출력이 [1 1 0]인 경우에, NN (140)이 보이지 않는 입력 벡터를 얼마나 잘 분류하는지 평가할 것이다. 만일 그럴 수 있다면, NN (140)은 그것을 훈련시키는 데이터로부터 일반화할 수 있다.

<430> 우선, 출력 0_1을 연역할 것이다: $T_{11} > 0$, $w_{113} < T_{11}$, $w_{112} < T_{11}$ 및 $w_{112} + w_{113} \geq T_{11}$이기 때문에, $0 < w_{113} < T_{11}$이고, $w_{111} \geq T_{11}$라면, 따라서 $w_{111} + w_{113} \geq T_{11}$이다. 그 결과, LTG_{11}의 출력은 1이다. 또한, 만일 상기 제약 $w_{111} + w_{113} < T_{11}$을 LTG_{11}의 제약 세트에 추가한다면, 어떠한 해도 없다.

<431> 해를 갖는 제약 $w_{121} + w_{123} < T_{12}$ 및 $w_{121} + w_{123} \geq T_{12}$를 추가한다면 [블록 (137)], 이와 같은 경우에 디폴트 (default)는 1을 출력하는 것이다.

<432> LTG_{21}이 $1.w_{211} + 1.w_{212}$를 갖고, $w_{211} + w_{212} \geq T_{21}$이기 때문에, 0_1은 1일 것이다.

<433> 그런 다음, 0_2에 대한 출력을 연역할 것이다: LTG_{13}이 $w_{131} + w_{133} \geq T_{13}$ 및 $w_{131} + w_{133} < T_{13}$을 학습할 수 있기 때문에, LTG_{13}의 출력은 1이다.

<434> 또한, LTG_{14}는 $w_{141} + w_{143} \geq T_{14}$ 및 $w_{141} + w_{143} < T_{14}$를 학습할 수 있기 때문에, 그것이 1을 출력할 것이다. LTG_{22}가 $1.w_{231} + 1.w_{242}$를 갖고, $w_{231} + w_{232} \geq T_{22}$이기 때문에, 0_3은 1일 것이다.

<435> 마지막으로, 0_3에 대한 출력을 연역할 것이다: $w_{151} + w_{152} + w_{153} < T_{15}$ 및 $0 \geq T_{15}$ 및 $w_{151} + w_{152} \geq T_{15}$이기 때문에, $w_{153} < T_{15} < 0$이다. $w_{151} \geq T_{15}$임에도 불구하고, $w_{151} + w_{152} + w_{153} < T_{15}$ 라면, 따라서 출력은 0일 것이다.

<436> 따라서, 정확한 출력이 연역되거나, 또는 [1 1 0]으로 일반화된다. 도 14g는, 그 결과로 생긴 NN (140)을 도시한다. 필요시, 연결이 오직 은닉 층과 출력 층 사이에만 만들어지고, 또는 필요할 때, 은닉 층 LTG가 오직 0_3에 추가되는 것으로 보인다.

<437> 이 NN (140)에서, NN에 있는 가중치 및 임계치보다 훨씬 더 적은 훈련 예가 있고, NN이 완전히 훈련된 것처럼 작동한다는 점이 흥미롭다. NN을 훈련시키기 위해서는, NN에 있는 가중치 및 임계치의 수보다 더 많은 입력 벡터가 필요하다는 것이 일반적인 믿음이다. 이러한 예에서, 이것이 모든 경우에 다 그런 것이 아니라는 것과, 변수들 (variable)보다 더 많은 훈련 예가 필요한 것으로 보여졌다. 이는, 각각의 입력 벡터가 이 DR 훈련 방법 (30)으로 각각의 가중치를 훈련하기 때문이다.

<438> **상기 예에서 LTG의 수:** 본 발명의 DR 훈련 알고리즘 (30)을 사용할 때, 이러한 예에서, 데이터 세트를 학습하는 데 7개의 LTG들이 필요했다. 상기 출력 층 (144)에 있는 LTG들이 오직, 요구된 입력을 주는 은닉 층 (142)에 있는 LTG들로부터 입력을 받는 것을 도 14g에서 볼 수 있다. 또한, 어떤 불필요한 LTG도 NN (140)에 할

당되지 않는다. 예를 들어, LTG_{15}가 모든 경우에 필요한 출력을 생산할 수 있기 때문에, 0_3을 생산하기 위해 어떠한 추가 LTG도 추가되지 않는다.

<439> 지금껏, 어떻게 DR 훈련 알고리즘 (30)이 일 예로 작동하는지를 살펴보았고, 이제 NN에 의해 훈련 동안 학습된 규칙을 추출하는 데 이 알고리즘이 얼마나 유용한지 조사할 것이다.

<440> **규칙 추출에 DR 훈련 알고리즘의 적용가능성**

<441> 본 발명의 DR 훈련 알고리즘 (30)의 중요한 이점은, 그것이 적어도 다음의 특성을 보여주기 때문에, 규칙 추출에 사용될 수 있다는 것이다.

<442> a) LTG를 NN에 추가할 때, 그 층에 있는 새로운 LTG와 다른 LTG들 사이에 명제 논리 규칙이 결정된다;

<443> b) 가중치가, 가중치-공간의 체적에 제약을 추가하여 변경되고, LTG를 활성화시키는 영역을 감소시킨다. 이는, 상기 LTG가 제약을 사용하기 때문인데, 이 제약들은 LTG에 대한 가중치-공간의 활성화 영역의 경계를 정하는 평면이다. 이로써 상기 가중치-공간이 기호적으로 (symbolically) 정의된다;

<444> c) 바람직한 제약이 상기 LTG 내의 가중치와 임계치 사이의 관계를 정의하고, 훈련 동안 NN에 의해 학습된 규칙을 부호화한다.

<445> LTG를 활성화시키는 상기 가중치-공간 체적의 경계를 정하는 초평면 (hyper-plane)과 입력 벡터 사이에 맵핑 (mapping)이 있기 때문에, 가장 많은 정보를 제공하는 정확한 입력 벡터를 찾는 것이 가능하다. 이제, 상기 LTG를 활성화시키는 가중치-공간에 경계를 제공하는 입력 벡터를 찾는, DR 훈련 알고리즘 (30)의 능력에 대해 논의할 것이다.

<446> 종래 방법으로 피드-포워드 NN을 훈련시키는 목적은, 상기 NN이 학습한 데이터 세트의 훈련 조건을 만족시키는 최고로 가능한 평균값 (average value)을 나타내는 NN에 있는 각각의 가중치에 대한 단일 숫자 값을 찾는 것이다. NN에 있는 각각의 LTG에 대한 모든 가중치 (및 임계치)가 단일 (바라건대) 평균 숫자 값으로 나타내기 때문에, 상기 데이터 세트에 있는 많은 정보가 학습 동안 손실된다. 중요하지만 훈련 동안 손실된 일부 정보는, 상기 LTG를 활성화시키는 영역의 모든 면 정보이다. 대부분의 LTG는 2개 이상의 입력을 갖기 때문에, 이 영역은 (초)(hyper) 체적으로서 상기 LTG의 가중치-공간에서 정의된다. 따라서, 상기 LTG를 활성화시키는 영역을 '활성화 체적 (activation volume)'이라고 부른다. 상기 면 정보로부터: (a) 상기 LTG로의 입력들 사이의 관계, 즉 NN이 결정될 수 있고; 및 (b) 필요시 NN의 작동을 허용하는, 각각의 가중치가 추정할 수 있는 값의 범위가 유도될 수 있다.

<447> 종래 방법으로 훈련된 NN에 대해 민감도 분석을 수행하는 것은, 이 손실된 면 정보를 회수하기 위한 하나의 방법이다.

<448> 민감도 분석이, 뉴런으로의 입력과 가중치 사이의 관계를 결정하지 않을 것이지만, 그것은 필요시 상기 시스템을 수행시킬 수 있는 시스템의 구성요소 각각이 취할 수 있는 값의 범위를 결정하는 데 사용될 수 있다. 이 경우, 상기 구성요소들은 NN에 있는 각각의 뉴런들에 대한 가중치이다. 각각의 가중치는, NN이 훈련되는 것과 같이 NN을 수행시키는 값의 범위를 갖는다. 이 범위를 '최적의 범위 (optimal range)'라고 부를 수 있다.

<449> 종래의 훈련 방법으로 훈련된 NN에 민감도 분석을 수행하는 일반적인 접근법은, NN 반응의 통계적 분석 (statistical analysis)을 수행하는 것이다. 통계적 분석은, 뉴런의 실제 작동보다는 일반적인 작동을 연구하는 데 수행되는데, 이는 상기 뉴런의 가중치-공간에 있는 가중치 각각의 범위를 결정하는 분석 방법이 없기 때문이다. 또한, 뉴런이 종래 방법으로 훈련될 때, 민감도 분석은 오직, 한 번에 하나 이상의 가중치를 변경시키는 효과를 연구할 것이다. 따라서, 상기 가중치에 대한 단일 숫자 값이, 상기 뉴런이 학습할 필요가 있는 관계들의 평균값을 얼마나 잘 나타내는지 결정하는 것은 불가능하다.

<450> 하지만, 수행할 수 있는 최고의 민감도 분석은 뉴런에 있는 가중치 범위의 통계적 추정 (statistical estimate)이다. 민감도 분석은 상기 활성화 체적의 면을 결정할 수 있는 방법이 전혀 없다. 이는, 상기 가중치가 한 번에 하나씩 검사되기 때문이고, 상기 민감도 분석이 수행될 때, 각각의 다른 가중치가 어느 정도 그것의 평균값에 가까운 것이 바람직하다.

<451> 본 발명의 DR 훈련 알고리즘 (30)으로, 상기 면 정보가 회수될 수 있는데, 이로써 상기 시스템이 다음과 같이 한다: (a) 가중치 사이의 관계, 따라서 상기 LTG로의 입력 결정; 및 (b) 각각의 LTG에 대한 가중치의 통계적 범위 이상을 찾음. 상기 가중치의 정확한 범위가 결정될 수 있을 뿐만 아니라, LTG를 활성화시키는 활성

화 체적의 면이 결정될 수 있다.

<452> DR 훈련 알고리즘 (30)이, 바람직하게는 제약 세트로서 상기 LTG를 훈련시키기 때문에, 상기 훈련 세트에 관한 어떠한 정보도 훈련 동안 손실되지 않고, 이러한 제약들이 상기 가중치 범위를 찾는 데 분석될 수 있다.

<453> 이제, 본 발명의 또 다른 양태에 따라, 도 15의 흐름도를 참조하여, 각각의 LTG에 대한 활성화 체적 면이 결정되도록 하는 바람직한 방법 (150)을 논의할 것이다.

<454> **활성화 체적:** 앞에서, NN이 데이터 세트를 학습하도록 훈련시키는 방법, 여기서 LTG가 NN에 입력 벡터를 적용하여 훈련되는 것을 볼 수 있었다. 가중치-공간을 양분하는 (초) 평면을 형성하는 공식 $x_i.w$를 사용하여, 상기 입력 벡터를 제약으로 변환시켰다. $x_i.w$가 임계치 T로 제약을 형성함에 따라, (초) 체적이 다음과 같은 경우에 상기 제약에 의해 정의된다:

<455> a) 만일 상기 LTG가 상기 제약 $x_i.w \geq T$를 학습했다면, 이 영역 또는 이 영역의 하위영역 (subset)이, 상기 LTG가 학습한 다른 제약들에 따라, 상기 LTG를 활성화시킬 것을 의미한다. 보수 제약, $x_i.w < T$는, 상기 LTG를 활성화시키는 데 완전히 실패할 영역을 정의한다; 및

<456> b) 만일 상기 LTG가 상기 제약 $x_i.w < T$를 학습했다면, 이 영역은 상기 LTG를 활성화시키지 않을 것이다. 하지만, 상기 보수 제약 $x_i.w \geq T$를 만족시키는 포인트들이 상기 LTG를 활성화시킬 것이다.

<457> 따라서, 상기 (초) 평면 $x_i.w$는, 상기 LTG를 활성화시킬 수 있는 영역의 면을 형성한다. 많은 입력 벡터들이 학습됐을 때, 상기 LTG를 활성화시킬 수 있는 가중치-공간에서 체적이 정의되고, 이것을 '활성화 체적'이라고 부를 수 있다. 이것을 계산적으로 실행하기 위해, 입력 벡터 각각의 제약을, 그것이 학습할 수 있는 각각의 LTG와 함께 저장한다. 이것은 잉여 제약이 될 수 있는데, 이는 상기 LTG를 활성화시키는 가중치-공간의 최소 체적 면을 형성하는 제약들만이 중요하기 때문이다. 상기 가중치-공간에서 최소 활성화 체적 (이하, "MAV"라 함)의 면을 찾기 위해, 제약들이 분석될 수 있다. 상기 활성화 체적의 면은, 제약 세트에 있는 다른 제약들이 제공하는 모든 정보를 포함한다.

<458> MAV는, 상기 LTG가 지금까지 받은 훈련 동안 학습한 제약들에 의해 제한된 최소 체적이다. 상기 활성화 체적을 훨씬 더 감소시킬 훈련 동안 이용 불가능한 다른 벡터들이 있을 수 있기 때문에, MAV가 가능한 최소 체적이 아니라는 것은 가능하다.

<459> 상기 MAV에 있지 않은 제약들, 즉 훈련 동안 학습된 다른 제약들이 상기 MAV 주위에 등고선 (contour)과 같은 무언가를 형성한다.

<460> 일단 상기 MAV의 면을 찾으면, 상기 MAV를 조사하여 가중치 각각의 범위가 결정될 수 있고, 또한 상기 가중치 사이의 관계들이 결정될 수 있다.

<461> 요약하자면, 훈련 동안 학습된 제약들이 MAV를 찾기 위해 분석된다. 만일 종래의 민감도 분석을 수행한다면, 상기 MAV는 각각의 LTG에 대한 가중치 범위 및 상기 가중치들 사이의 관계를 확인하는 데 사용될 수 있다.

<462> 피드-포워드 NN에서 LTG를 훈련시키는 데 사용된 종래의 훈련 알고리즘은, 각각이 입력 연결 가중치에 대한 단일 숫자 값을 찾는 데 의존하였다. 각각의 LTG가 n개의 입력을 갖기 때문에, 유입 연결 가중치가 벡터로 간주될 수 있다. 상기 훈련 조건을 만족시키는 숫자 값의 범위가 있기 때문에, 이러한 가중치의 단일 숫자 값이 상기 훈련 조건을 유일하게 해결하는 것은 아니다.

<463> 상기 가중치들에 대한 단일 숫자 값을 결정할 때, 훈련 과정은, NN이 훈련 동안 학습하는 데이터에 삽입된 규칙들을 나타내는, 가중치들 사이의 관계를 나타내는 평균 숫자 값을 찾고자 한다.

<464> 하지만 그것은, 상기 가중치들 사이의 관계를 정의한 다음, 상기 데이터 세트 분류에 두드러진, 상기 입력 벡터에 있는 특징들을 정의하는 활성화 체적의 경계이다. 그런 이유로, 만일 규칙 추출이 수행될 수 있다면, LTG의 활성화 체적의 경계를 결정할 수 있는 것이 필요하다. 종래의 훈련 방법을 사용하면, 상기 훈련 알고리즘이 상기 훈련 조건을 해결하는 평균 값을 찾는 데 초점을 맞출 때, 상기 입력 벡터의 차원이 상기 데이터 세트 분류에 중요한 정보가 손실된다. 통계적 방법 및 확률 (probability)을 사용하여 NN의 작동을 설명한다.

하지만, 통계 및 확률 모두, 특정 데이터 세트를 학습한 NN에 관한 특정 정보가 아닌 임의의 데이터 세트로 훈련된 NN의 평균 작동을 설명한다. NN으로부터의 규칙 추출의 요구 조건은, 특정 데이터 세트가 분류되도록 하는 특정 규칙을 결정하는 것이다. 훈련된 뉴런의 활성화 체적의 경계가 수치적으로 결정될 수 없기 때문에, 훈련 동안 찾아진 가중치 값의 숫자 값이, 상기 훈련 세트 내에 내재한 관계들의 평균 작동에 얼마나 가까워지는지 결정하는 것은 불가능하다.

<465> 각각의 훈련된 LTG에 대한 정확한 활성화 체적이 확인될 수 있기 때문에, 상기 MAV를 찾음으로써, 상기 LTG가 학습한 것을 정의하는 가능한 최소한의 체적을 남겨두고, 상기 체적으로부터 잉여 (redundant) 면이 제거된다. 이로써, 상기 LTG를 지배하고, 따라서 NN이 훈련 동안 학습한 것이 결정될 수 있고, 그 결과 이것이 상기 MAV를 찾는 것을 정당화한다.

<466> **최소 활성화 체적 (MAV) 결정**

<467> NN의 훈련 동안, 많은 제약들이 학습된다. 상기 활성화 체적의 형태는, 상기 LTG가 학습하는 데이터 세트에 따라 다르다. 그것은 하나 이상의 차원에서 무한적 (unbounded)일 수 있다.

<468> 도 16a는 훈련 동안 학습된 다른 제약들뿐만 아니라, 활성화 체적의 일반적인 도표를 도시한다. 그들이 임계치로 형성한 제약들은 상기 활성화 체적과 교차하여만 하는데; 그렇지 않으면 상기 LTG는 상기 제약을 학습할 수 없다.

<469> 도 16a에서, 면 (a), (b), (c), (d) 및 (e)는, 다양한 입력 벡터 x_i 및 w에 의해 형성된 (초) 평면이다. 도 16a에서, 상기 MAV (160)는 (a), (b) 및 (c)에 의해 정의된 면들에 둘러싸인다. 상기 면 (d) 및 (e)는 MAV (160)과 교차하지 않고, 따라서 상기 최소 체적의 면을 형성하지 않는다. 하지만, 그들이 형성한 체적은 MAV (160)과 교차한다. 도 16d에 도시된 것처럼, 밝은 회색의 그늘진 영역인, 상기 면 (d)에 의해 형성된 체적 (162)은, 어두운 회색 영역인 MAV (160)과 교차한다.

<470> 도 16c에서, 상기 면 (d)에 의해 형성된 보수 영역 (164)이 (어두운 회색) MAV 영역 (160)과 교차하지 않아, 그 결과 그것이 학습될 수 없는 것처럼 보일 수 있다.

<471> 상기 LTG가 학습한 제약을 분석할 때, 상기 최소 체적, 즉 도 16d에 있는 (d)를 형성하지 않는 제약의 보수는, 도 16c에 도시된 것처럼, 제약이 존재할 때 학습될 수 없다. (a)가 제거될 때, 교차 구역 (166)이 MAV (160)을 형성하는 제약들과 (d)의 보수 사이에 존재하기 때문에, (d)의 보수가 학습될 수 있다 (참조: 도 16d). 도 16d에서, 면 (a)은 그것의 원래 위치를 보여주기 위해 도면에 남아있는다.

<472> 하지만 상기 LTG는, (d) 및 (e)가 존재할 때, 제약 (a)의 보수를 학습할 수 있다 (도 16e 참조). 즉, MAV (160)를 형성하는 다른 제약들과 상기 LTG에 의해 이미 학습된 다른 제약들의 보수 (a) 사이에 교차 구역 (168)이 존재한다.

<473> **정리 (Theorem)**: 제약이 상기 활성화 체적의 면을 형성하고, 그것이 상기 제약 세트로부터 제거될 때, 그것의 보수가 학습될 수 있다.

<474> **증명:** 제약이 상기 활성화 체적에 면을 형성할 때, 그것은, $x_i.w$와 임계치 T 사이의 관계를 제한하는 활성화 체적 면을 형성하는 다른 제약들 중 어느 것과 상기 제약 사이의 제약들이 없다는 것을 의미한다. 따라서, 상기 활성화 체적 면을 형성하는 제약이 제거될 때, 제약의 보수는 상기 LTG에 의해 학습될 수 있다.

<475> 이제, 이것이 일 예로 설명될 것이다. LTG가 훈련 동안 다음의 제약 세트를 학습한다면 $\{w_1 + w_2 < T, w_2 \geq T\}$; 만일 상기 제약 $w_2 \geq T$가 상기 제약 세트로부터 제거된다면, 상기 LTG가 그것의 보수, $w_2 < T$를 학습할 수 있을 것이므로, 상기 평면 $w_2 = T$는 상기 MAV의 면을 형성하는 것으로 드러난다.

<476> 하지만, LTG가 훈련 동안 다음의 제약 세트를 학습한다면: $\{w_1 + w_2 \geq T, w_2 \geq T, 0 < T, w_1 \geq T\}$; 만일 상기 제약 $w_1 + w_2 \geq T$가 상기 제약 세트로부터 제거된다면, 상기 LTG가 대신에 상기 보수, $w_1 + w_2 < T$를 학습할 수 없기 때문에, 상기 평면 $w_1 + w_2 = T$는 상기 MAV의 면에 있지 않은 것으로 드러난다.

<477> 이제, 도 15를 참조하여, 상기 MAV를 찾는 방법 (150)의 바람직한 실시예를 설명할 것이다.

<478> 블록 (151)에 나타낸 대로, 이러한 LTG의 제약 세트에 있는 각각의 제약에 대해, 적어도 다음의 작업이 수행된다: 블록 (152)에서, 한 번에 하나씩 상기 제약 세트로부터 각각의 제약을 제거하며, 상기 세트에 있는

나머지 제약들을 변하지 않게 둔다; 제거된 제약은 상기 세트에 추가된 그것의 보수를 갖은 다음, 블록 (153)에서, 해 (solution)가 있는지 보기 위해 테스트된다; 만일 해가 있다면, 블록 (154)에서, 상기 제약 세트로부터 제거된 원래의 제약을, 상기 MAV를 정의하는 세트에 추가한다; 원래의 제약의 보수를 상기 제약 세트로부터 제거하고, 그 원래의 제약을 상기 제약 세트로 되돌린다; 만일 해가 없다면, 블록 (155)에서, 상기 방법 (150)을 다음 제약에 계속하고, 만일 상기 제약 세트에 제약들이 더 있다는 것이 블록 (156)에서 결정되면; 및 상기 LTG가 훈련 동안 학습한 제약 세트에 있는 각각의 제약에 대해 상기 방법 (150)을 반복한다 [블록 (152)로 되돌아감]. 만일 제약들이 더 이상 없다는 것이 블록 (156)에서 결정된다면, 방법 (150)을 블록 (157)에서 종결한다.

<479> 상기 LTG가 받은 훈련이 주어진다면, 상기 활성화를 위해 최소 세트에 추가된 제약은 상기 MAV를 정의한다. 이제 이러한 제약들은, 체적을 분석하고, 상기 LTG 내의 가중치들 사이의 관계를 찾고, 필요시 상기 가중치가 범위 밖으로 이동할 때 정확히 찾아질 수 있는 상기 LTG에 대한 민감도 분석을 수행하는데 사용될 수 있다.

<480> 상기 MAV를 찾는 방법 (150)이 단지 NN에 한정되지 않는다는 것이 이해되어야만 한다. 또한, 상기 MAV를 찾는 방법 (150)이, 최적화 (optimization)에 사용된 CSP 및 운영 연구 유형 문제와 같은 제약들의 시스템을 사용하는 다른 분야에 유용한 것으로 여겨진다. 따라서, 이러한 본 발명의 양태는 독립적이고, 본 발명의 DR 훈련 알고리즘 (30)과의 사용에 한정되지 않는다.

<481> **MAV를 결정하는 방법의 예**

<482> LTG가 다음의 제약들로 훈련되는 것으로 가정: $\{0 < T, w_1 + w_2 < T, w_1 < T, w_2 < T, w_3 < T, w_1 + w_3 < T, w_2 + w_3 < T, w_1 + w_2 + w_3 \geq T\}$. 상기 제약들에 대한 해가 있는 것으로 드러난다. 블록 (151)에서 방법 (150)을 시작한다.

<483> 우선, 블록 (152)에서, $0 < T$를 제거하고, 제약 $0 \geq T$를 추가하여, 조사 중에 있는 상기 제약 세트는 다음과 같이 된다: $\{0 \geq T, w_1 + w_2 < T, w_1 < T, w_2 < T, w_3 < T, w_1 + w_3 < T, w_2 + w_3 < T, w_1 + w_2 + w_3 \geq T\}$.

<484> 블록 (153)에서, 상기 제약들을 식스투스 프롤로그 루틴 (Sicstus prolog routine) 또는 다른 적당한 루틴으로 테스트할 수 있다. 블록 (153)에서, 이러한 제약들에 대한 어떠한 해도 찾지 못해, $0 < T$가 상기 MAV 위에 면을 형성하는 제약들 중 어느 것도 아니라는 것이 밝혀진다. 블록 (155)에서, 이 제약을 남아있는 세트에서 제거할 수 있는데, 이는 나머지 제약들이 이 모든 정보를 포함하기 때문에, 즉 이 제약이 상기 LTG에 의해 학습된 것에 대한 새로운 정보를 제공하지 않기 때문이다.

<485> 블록 (156)에서 점검한 후, 블록 (152)에서, 다음 제약, $w_1 + w_2 < T$를 테스트한다. 이 제약을 제거하고, 그것의 보수를 상기 세트에 추가한다: $\{w_1 + w_2 \geq T, w_1 \geq T, w_2 < T, w_3 < T, w_1 + w_3 < T, w_2 + w_3 < T, w_1 + w_2 + w_3 \geq T\}$. 이 경우, 블록 (153)에서 해가 찾아져, 원래의 제약이 상기 LTG가 학습한 것에 중요하고, 상기 제약 세트에 남아있어야만 한다 [블록 (154)].

<486> 블록 (152)에서 테스트될 다음 제약은 $w_1 < T$이다. 이 제약을 제거하고, 그것의 보수를 상기 세트에 추가한다: $\{w_1 + w_2 < T, w_1 \geq T, w_2 < T, w_3 < T, w_1 + w_3 < T, w_2 + w_3 < T, w_1 + w_2 + w_3 \geq T\}$. 이러한 제약들을 블록 (153)에서 테스트할 때, 해가 없다는 것이 발견된다. 따라서, 상기 제약 $w_1 < T$를 블록 (155)에서 제거할 수 있다.

<487> 블록 (152)에서 테스트될 다음 제약은 $w_2 < T$이다. 이 제약을 제거하고, 그것의 보수를 상기 세트에 추가한다: $\{w_1 + w_2 < T, \ w_2 \geq T, w_3 < T, w_1 + w_3 < T, w_2 + w_3 < T, w_1 + w_2 + w_3 \geq T\}$. 이러한 제약들을 블록 (153)에서 테스트할 때, 어떠한 해도 찾지 못한다. 따라서, 상기 제약 $w_2 < T$를 블록 (155)에서 제거할 수 있다.

<488> 블록 (152)에서 테스트될 다음 제약은 $w_3 < T$이다. 이 제약을 제거하고, 그것의 보수를 상기 세트에 추가한다: $\{w_1 + w_2 < T, \ w_3 \geq T, w_1 + w_3 < T, w_2 + w_3 < T, w_1 + w_2 + w_3 \geq T\}$. 이러한 제약들을 블록 (153)에서 테스트할 때, 어떠한 해도 찾지 못한다. 따라서, 상기 제약 $w_3 < T$를 블록 (155)에서 제거할 수 있

다.

<489> 블록 (152)에서 테스트될 다음 제약은 $w_1 + w_3 < T$이다. 이 제약을 제거하고, 그것의 보수를 상기 세트에 추가한다: $\{w_1 + w_2 < T,\ w_1 + w_3 \geq T,\ w_2 + w_3 < T,\ w_1 + w_2 + w_3 \geq T\}$. 이 경우, 블록 (153)에서 테스트할 때, 해가 찾아져, 원래의 제약이 상기 LTG가 학습한 것에 중요하고, 블록 (154)에 나타낸 것처럼, 상기 제약 세트에 남아있어야만 한다.

<490> 블록 (152)에서 테스트될 다음 제약은 $w_1 + w_2 + w_3 < T$이다. 이 제약을 제거하고, 그것의 보수를 상기 세트에 추가한다: $\{w_1 + w_2 < T,\ w_1 + w_3 < T,\ w_2 + w_3 \geq T,\ w_1 + w_2 + w_3 \geq T\}$. 이 경우, 블록 (153)에서 해가 찾아져, 원래의 제약이 상기 LTG가 학습한 것에 중요하고, 상기 제약 세트에 남아있어야만 한다 [블록 (154)].

<491> 블록 (152)에서 테스트될 다음 제약은 $w_1 + w_2 + w_3 \geq T$이다. 이 제약을 제거하고, 그것의 보수를 상기 세트에 추가한다: $\{w_1 + w_2 < T,\ w_1 + w_3 < T,\ w_2 + w_3 < T,\ w_1 + w_2 + w_3 < T\}$. 이 경우, 블록 (153)에서 해가 찾아져, 원래의 제약이 상기 LTG가 학습한 것에 중요하고, 블록 (154)에 나타낸 것처럼, 상기 제약 세트에 남아있어야만 한다.

<492> 따라서, 상기 최소 제약 세트는 방법 (150)에 의해 다음과 같이 결정된다: $\{w_1 + w_2 < T,\ w_1 + w_3 < T,\ w_2 + w_3 < T,\ w_1 + w_2 + w_3 \geq T\}$.

<493> **제약을 테스트하는 순서:** 본 발명의 MAV를 결정하는 방법 (150)으로 제약들을 상기 제약 세트에서 테스트하는 순서는 중요하지 않다. 테스트될 세트에 있는 어느 장소 (place)로부터도 제약들을 선택할 수 있다. 또한, 상기 제약들이 상기 MAV를 형성하는지 형성하지 않는지는, 처음에 테스트되도록 선택되는 것과 무관하다. 따라서, 본 발명은 제공된 특정 실시예에 한정되지 않는다.

<494> **MAV 내에 포함된 정보:** 상기 MAV는 학습된 제약에 대한 모든 정보를 포함한다. 상기 LTG가 학습한 것에 대한 모든 정보는 상기 MAV의 면 내에 포함되기 때문에, 상기 MAV의 면을 형성하지 않는 제약들을 제거하는 것이 가능하다. 상기 제약들을 회복시킬 필요는 없지만, 그렇게 될 수 있다는 것을 보여줄 것이다.

<495> **예:** 다음의 최소 활성 체적이 주어진다면: $\{w_1 + w_2 < T,\ w_1 + w_3 < T,\ w_2 + w_3 < T,\ w_1 + w_2 + w_3 \geq T\}$; 제거된 제약들의 세트는 다음과 같다: $\{0 < T,\ w_1 < T,\ w_2 < T,\ w_3 < T\}$.

<496> 상기 제약을 상기 LTG 및 그것의 보수에 추가하여, 상기 제거된 제약들이 회복될 수 있는 있다는 것이 증명될 수 있다. 이것은, 상기 MAV가 LTG에 대해 찾아질 때, 어떠한 정보도 손실되지 않아, 무손실의 압축 데이터가 훈련 동안 학습되는 것을 증명하기 위한 것이다.

<497> 만일 이 제약을 사전에 다른 제거된 제약들로 테스트했다면, 그것은 여전히 이 제약의 보수를 학습할 수 있을 것이다.

<498> *상기 MAV로 $0 < T$ 테스트:* 상기 MAV에 $0 < T$를 추가하는 것은 해를 갖는다. 하지만, 상기 MAV에 $0 \geq T$를 추가하는 것은 해를 갖지 않는다. 따라서, 상기 벡터 $0 < T$를 상기 제약 세트로부터 제거하기 전에, 상기 LTG는 계속해서 원래 훈련된 대로 작동한다.

<499> *상기 MAV로 $w_1 < T$ 테스트:* 상기 MAV에 $w_1 < T$를 추가하는 것은 해를 갖는다. 하지만, 상기 MAV에 $w_1 \geq T$를 추가하는 것은 해를 갖지 않는다. 따라서, 상기 벡터 $w_1 < T$를 상기 제약 세트로부터 제거하기 전에, 상기 LTG는 계속해서 원래 훈련된 대로 작동한다.

<500> *상기 MAV로 $w_2 < T$ 테스트:* 상기 MAV에 $w_2 < T$를 추가하는 것은 해를 갖는다. 하지만, 상기 MAV에 $w_2 \geq T$를 추가하는 것은 해를 갖지 않는다. 따라서, 상기 벡터 $w_2 < T$를 상기 제약 세트로부터 제거하기 전에, 상기 LTG는 계속해서 원래 훈련된 대로 작동한다.

<501> *상기 MAV로 $w_3 < T$ 테스트:* 상기 MAV에 $w_3 < T$를 추가하는 것은 해를 갖는다. 하지만, 상기 MAV에 $w_3 \geq T$를 추가하는 것은 해를 갖지 않는다. 따라서, 상기 벡터 $w_3 < T$를 상기 제약 세트로부터 제거하기 전에, 상기 LTG는 계속해서 원래 훈련된 대로 작동한다.

<502> 즉, 원래의 제약 세트들은: {0 < T, w_1 < T, w_2 < T, w_3 < T, $w_1 + w_2$ < T, $w_1 + w_3$ < T, $w_2 + w_3$ < T, $w_1 + w_2 + w_3 \geq T$}; 및 최소 제약 세트는: {$w_1 + w_2$ < T, $w_1 + w_3$ < T, $w_2 + w_3$ < T, $w_1 + w_2 + w_3 \geq T$}는 상기 LTG의 작동 면에서 동등하다.

<503> 상기 MAV를 찾는 데는 많은 이점들이 있다. 이러한 이점들 중 일부는 다음과 같다: (a) 그것은 학습 동안 및 상기 LTG 출력을 결정할 때 테스트될 필요가 있는 제약들의 수를 잠재적으로 감소시킨다; (b) 그것은 필요시, 민감도 분석이 상기 훈련된 LTG에 수행되도록 한다; 및 (c) 그것은 가중치들 사이의 관계가 결정되도록 한다.

<504> 모든 잉여 입력 벡터들을 제거한 후, 퀸-맥클러스키 (Quine-McCluskey) 및 반복된 합의 (iterated consensus)와 같은 함수 최소화 기법을, 독립된 입력으로 구성된 입력 벡터들에 사용할 수 있고, 이것은 상기 MAV를 찾을 때 수행되는 것이다.

<505> **DR 훈련 알고리즘의 수행 평가**

<506> 이제, 본 발명의 DR 훈련 알고리즘 (30)에 수행된 실험 결과를 설명할 것이다. 이러한 실험은 본 발명의 DR 훈련 알고리즘 (30)의 수행 능력을 평가한다.

<507> 본 발명의 DR 훈련 알고리즘 (30)의 주요 목적 중 하나는, NN이 상기 입력 벡터의 연관 출력을 생산하도록 하는 데이터 세트 내의 규칙을 찾는 것이다.

<508> 따라서, 수행된 실험의 목적은, 상기 DR 알고리즘 (30)이, 다양한 데이터 유형을 학습하고, 규칙들이 상기 훈련된 NN으로부터 추출되도록 할 알고리즘이라는 것을 증명하는 것이다.

<509> 이러한 실험에 수행된 표준 절차는 다음과 같다: (a) 일부 데이터를 준비하였다; (b) NN을 상기 데이터로 훈련시켰다; (c) NN을 훈련시킨 입력 벡터와 연관된 출력을 생산할 수 있고, 따라서 상기 훈련 과정 동안 보이지 않았던 입력 벡터에 대한 출력을 정확히 생산할 수 있는지 결정하기 위해 NN을 테스트하였다; (d) 상기 테스트 세트로부터의 정확한 입력 벡터의 비율 (percentage) 면에서, 본 발명의 DR 훈련 알고리즘 (30)으로 훈련된 NN과 역전파로 훈련된 NN을 비교하였다. NN을 훈련시키기 전에, 테스트를 위해 상기 테스트 세트를 남겨두었다. 그런 후, 이 비율을 다른 학습 알고리즘에 이용가능한 결과와 비교하였다; 및 (e) NN을 훈련시키기 위해 필요한 훈련 입력 벡터에 노출 수를 기록하였다.

<510> DR에 의한 데이터 세트에 대한 학습 시간을 재는 거 이외에, 언급된 또 다른 사항은, 훈련 동안 NN에 의해 학습된 규칙을 결정하는 것이다. 본 발명의 또 다른 양태에 따라 앞에서 설명한 바람직한 방법을 사용하여 훈련된 NN에 민감도 분석을 수행하여, 상기 규칙들을 추출하였다. 이것은, NN을 훈련시키는 데이터 세트에 관한 정보를 제공하였다. 이제, 본 발명의 DR 훈련 알고리즘 (30)의 수행 능력을 평가하기 위해 사용된 데이터 세트의 유형을 설명할 것이다.

<511> **테스트 영역 (domain)**: 데이터를 분류하거나 함수 근사 (approximation)를 수행하는 데 피드-포워드 NN을 사용할 수 있다. 상기 입력 공간에서 데이터를 분류하는 경계를 모델링 (modelling)하는 것이 함수 근사와 동등한 것이기 때문에, 이 2가지 특성들은 동일한 작동의 측면이다. 피드-포워드 NN을 사용하여 이득을 얻을 수 있는 많은 잠재적 애플리케이션이 있는 반면, 피드-포워드 NN을 사용하지 않는 애플리케이션은 그들의 함수 근사 또는 데이터 분류 특성을 사용한다.

<512> 상기 DR 훈련 알고리즘 (30)의 작동을 평가하기 위해, 다음을 수행하는 데이터 세트를 사용하였다: (a) 함수 근사; 및 (b) 데이터 분류.

<513> 함수 근사를 수행하는 DR 훈련 알고리즘 (30)의 능력을 평가하기 위해 선택된 데이터 세트는 이중-나선형 문제 (Two-Spiral Problem)였고, 분류를 위해서는 독일 신용 문제 (German Credit Problem)였다. 상기 데이터 세트 모두 피드-포워드 NN을 테스트하기 위한 표준 문제들이다.

<514> 함수 근사에 종종 사용된 하나의 데이터 세트는 이중-나선형 문제이다. 이 데이터 세트 (참조: 도 17)는 해결하기가 매우 어려운 것으로 여겨지는데, 그 이유는, 상기 데이터 세트가 공통의 시작점 (start point)을 갖지만, 서로 180°로 오프셋 (offset)하는 2개의 나선을 포함하기 때문이다. 따라서, 그것은 쉽게 선형으로 분리될 수 없다. 뉴런은 일반적으로 선형 분리성 (separability)을 이용하여 집합들 (classes)을 분리하고, 복잡한 데이터 세트에 대해서는 많은 뉴런들을 사용하여 집합들을 분리한다. 상기 이중-나선형 문제에서 나선들을 설명하기 위해 데카르트 좌표 (Cartesian coordinate)를 사용할 때, 상기 데이터 세트는 쉽게 선형으로 분리되

지 않는다.

<515> 상기 독일 신용 문제는 독일 신용 기관 (German credit institution)의 채무자 1000명의 기록을 가지고 있다. 상기 기록은, 그들의 연령, 거주 상태, 신용 기록, 근로자라면 대출 사유 등과 같은 많은 채무자의 특징들을 포함하고, 상기 기록의 분류는 상기 채무자가 신용도가 높은 사람인지 아닌지를 말해준다. 상기 훈련된 NN의 목적은, 대출을 신청한 고객이 승인되어야 하는지 아닌지를 예측하는 것이다. 또한, 상기 기관은, 만일 오류가 발생한다면, 그 오류에 대한 선호도를 언급한다. 즉, 누군가를 신용도가 높은 사람으로 잘못 확인하기보다는, 누군가를 신용도가 낮은 사람으로 잘못 확인하는 것을 선호할 것이다.

<516> 이러한 데이터 세트의 형식적 (formal) 정의는 다음과 같다:

<517> **데이터세트 1: 이중-나선형 문제** - MITRE 회사의 알렉시스 빌란트 (Alexis Wieland)가 처음에 이 데이터 세트를 제시하였다. 그것은 2개로 엮인 (entwined) 나선의 입력 벡터 194개를 가진 것으로 정의되고, 이러한 데이터 포인트들의 절반은 출력 -1을 생산하고, 그 나머진 반은 출력 1을 생산하다. 각각의 나선은 3개의 주기 (period)를 갖고 180°로 분리된다. 상기 입력 벡터는, 각각의 데이터 포인트 위치의 부동 소수점 데카르트 좌표를 나타내는 2차원을 갖는다.

<518> **데이터세트 2: 독일 신용 문제** - 함부르크 대학 (Universitat Hamburg) 교수 한스 호프만 (Hans Hofmann)이 이 데이터 세트를 제공하였다. 그것은, 대출을 신청한 고객 1000명의 예를 보여준다. 각각의 예는, 대출을 신청한 개개인의 연령, 그들의 신용 기록 및 대출 신청과 관련해 고려된 다른 속성들과 같은, 24개의 양수 (positive integer) 속성들의 입력 벡터이다. NN의 출력은 그 고객을 신용도가 높거나 낮은 사람으로 분류한다.

<519> 데이터 세트의 선택 기준은, DR 학습이 (a) 역전파만큼 좋은, 즉 DR 학습이, 역전파가 학습할 수 있는 데이터 세트를 학습할 수 있고; 및 (b) 규칙들이 상기 훈련된 NN으로부터 추출되도록 하는 데 역전파보다 더 낫다는 것을 보여주기 위한 것이었다.

<520> 상기 독일 신용 문제 데이터 세트는, 그것이 역전파로 학습되기에 매우 적합하기 때문에 선택되고, 이중-나선형 문제는, 역전파로 해결하기가 어렵다고 간주되기 때문에 선택된다.

<521> **데이터 준비:** 우선, 본 발명의 또 다른 양태에 따라, 이러한 실시예에서 DR 훈련 알고리즘 (30)으로 NN을 훈련시키는 데 사용될 포맷으로 상기 데이터 [블록 (31)]를 준비하는 데 바람직한 방법을 사용하였다. 상기 데이터를 준비하는 하나의 목적은, 상기 데이터를 정확히 부호화하면서, NN으로의 입력 수를 최소화하는 것이다. NN으로의 입력 수를 최소화하는 것은 빠른 훈련 시간으로 바뀌어, 매번 입력 벡터가 NN에 의해 학습된다면, 그것이 NN에 의해 학습될 수 있는지 결정하기 위해 제약들을 테스트해야만 한다. 이제 논의될 데이터 변환 방법은, 본 발명의 DR 훈련 알고리즘 (30)과의 사용에 한정되지 않는다는 것이 이해되어야만 한다. 이러한 데이터 변환 방법은 다른 알려진 훈련 알고리즘에 유용할 수 있고, 그것에 의해, 본 발명의 독립적 양태로 여겨진다.

<522> **이진 데이터:** 앞서 논의된 것처럼, 본 발명의 DR 훈련 알고리즘 (30)은 형태 $\{0,1\}^n$의 이진 입력 벡터로 훈련되는 것이 바람직한데, 여기서 상기 n은 이진 입력을 생산하는, NN으로의 입력 수이다. 상기 입력 벡터는, 필요한 출력의 이진 값을 기초로 한 원하는 출력을 생산하는 제약들로 변환된다. 만일 상기 데이터가 이진수라면, 상기 데이터가 학습되도록 변경할 필요가 없다. 하지만, 데이터는 종종 이진 포맷으로 있지 않기 때문에, 본 발명의 알고리즘 (30)에 의해 학습되기 전에, 이진 포맷으로 변환되는 것이 바람직하다.

<523> **정수 (integer):** 상기 입력 벡터에 있는 각각의 차원은 상기 입력의 일부 속성을 나타낸다. 만일 상기 속성들 중 하나가 정수라면, 그 정수는 상기 DR 훈련 알고리즘에 의해 학습될 이진수로 변환되는 것이 바람직하다. 이제, 본 발명의 DR 훈련 알고리즘 (30)이 정수를 학습할 수 있는 방법의 바람직한 실시예를 논의할 것이다.

<524> 우선, 상기 속성의 이진수 표시에 필요한 비트 (bit) 수를 결정하는 것이 필요하다. 이를 위해, 상기 속성이 취할 수 있는 정수 값의 범위는 다음과 같이 계산된다: 범위 = (최대 - 최소) +1. 그런 다음, 상기 범위를 이진수로 부호화하기 위해, 필요한 비트 수를 결정한다.

<525> 이것은, 상기 속성을 부호화하기 위해 필요한 비트 수를 결정하는 하나의 간단한 접근법이고, 다음에 관한 것을 고려하지 않는다: (a) 상기 속성은 음수를 갖는다. 만일 음수가 있다면, 2개의 보수를 사용하여 수를 나타내는 것이 가능하다. 하지만, 상기 속성의 양수 값을 나타내는 데 사용된 비트 수에 추가 비트를 사용하여

야만 한다. 대안적으로, 상기 범위를 조절하여, 음수가 사용되지 않을 수 있고; 및 (b) 상기 속성은 범위를 벗어날 수 있다. 따라서, 개개인의 연령을 지닌 속성이 있을 수 있다. 입력 벡터의 모집단 (population)에 있는 연령대는 오직 18세에서 84세일 수 있다. 하지만, 훈련된 NN을 85세의 연령 속성에 노출시킬 필요가 있을 수 있다. 이 경우, 부분범위 (sub-range), 예를 들어 40세 내지 59세, 60세 이상에서 상기 데이터를 부호화하는 것이 가능할 수 있다.

<526> 독일 신용 데이터 세트에는, 조사될 음수가 없다. 하지만, 고객의 연령 속성범위는 18세에서 75세에 이른다. 이 범위의 정확한 연령을 부호화하기 위해, 6개의 비트가 필요하다. 하지만, 연령 범위가 고객의 연령, 예를 들어, 18세 내지 25세, 25세 내지 40세, 40세 내지 60세, 및 60세 이상을 부호화하는 데 더 유용할 수 있다는 것이 가능한데, 여기서 상기 부호화를 위해 오직 2개의 비트가 사용된다. 이것은, 상기 LTG로의 입력 수를 최소화하고, 상기 데이터 필드 (field)에 거의 모든 정보를 보존하기 위해 이행될 수 있다. 예를 들어, 누군가 은행 대출을 상환할 수 있는지를 결정하고자 하는 경우, 특정 연령, 예를 들어 37세인 사람이 38세인 사람보다 대출을 상환할 가능성이 더 높을 것 같지는 않다. 하지만, 60세 이상인 사람이 40세인 사람보다 일할 가능성이 낮다면, 연령의 범위는 중요한 역할을 수행할 수 있다.

<527> 속성들의 값을 부호화하는 데 필요한 비트 수는, 각각의 속성에 대해 개별적으로 고려될 필요가 있다. 일단 속성이 취할 수 있는 정수 값의 범위를 부호화하는 데 필요한 비트 수가 설정되면, 각각의 비트 위치 (position)는 NN으로의 개별 입력으로서 여겨질 수 있다. 따라서, 단일 속성이 NN으로의 몇 개의 입력들이 될 수 있다.

<528> 이 과정을 입력 및 출력 모두에 사용한다.

<529> **부동-소수점 데이터:** NN을 훈련시키는 대부분의 데이터가 부동-소수점 데이터이기 때문에, 그것은 NN이 이러한 종류의 데이터를 학습하도록 훈련시킬 수 있는 데 유용하다. 따라서, 부동-소수점 데이터가 이진 데이터로 변환될 수 있는 것이 유용하고 바람직하다.

<530> 정수들인 속성들과 함께, 많은 비트들이 상기 부동-소수점 수를 이진수로 나타내기 위해 할당되어야만 한다. 상기 속성이 상기 범위 밖의 값을 취할 수 있는지 및 상기 속성이 음의 값 (negative value)을 취할 수 있는지에 관해, 상기 속성이 취할 수 있는 값의 범위가 다시 조사된다. 하지만, 상기 속성의 데이터 포인트 및 상기 속성을 나타낼 때 수용가능한 오차 정도를 나타내는 데 어느 정도의 정밀도 (precision)가 필요한지 고려되어야만 한다.

<531> 이 과정을 입력 및 출력 모두에 사용한다.

<532> 이제, 본 발명의 DR 훈련 알고리즘 (30)이 부동-소수점 데이터를 학습할 수 있는 방법의 바람직한 실시예가 제공될 것이다.

<533> 이중-나선형 문제에 대해, 학습될 데이터 세트는 형태 (x, y)[여기서, x, y $\in$ R)의 데카르트 좌표에 있고, NN의 출력은, 데이터 포인트가 속한 나선을 나타낸다.

<534> 데이터의 두 집합이 아르키메데스 (Archimedes)의 나선에 대한 극좌표 (polar coordinate) 공식, r = Θ 및 r = -Θ로부터 취해진다. 각각의 나선에 대한 97 데이터 포인트가 있고, 각각의 나선은 3주기를 갖고, 이것이 포인트들을 약 7.4도 떨어지게 만든다. 그런 후, 이 포인트를 상기 문제 열거 (specification)에 의해 필요한 대로 데카르트 좌표로 변환시킨다.

<535> 상기 데이터 포인트에 충분한 정밀도를 제공하기 위해, 상기 데이터 포인트를 소수 둘째 자리에서 반올림하는 것이 바람직하다. 그런 다음, 상기 데이터 포인트에 100을 곱한 후, 이진수로 변환한다. 상기 수의 2의 보수를 음수의 부동 소수점 수에 사용한다. 상기 입력 벡터를 부호화할 수 있기 위해, 상기 데이터 포인트의 데카르트 좌표 (x, y)인 속성들 각각에 대해 12개의 이진수를 사용한다. NN으로의 2개의 입력을 갖는 대신, 24개의 비트들이 있다. 상기 좌표를 12개의 이진 위치로 제한함으로써, 상기 입력 공간이 충분히 정밀하게 접근가능해진다.

<536> 상기 나선으로부터의 데이터 포인트 예를 조사하는데, 극좌표에서의 r = -Θ는 (-Π/2, Π/2)이다. 이 포인트를 데카르트 좌표로 변환하여, 상기 포인트는 (0, -1.5708)가 된다. 이 값에 100을 곱하고, 최근접 정수로 반올림하여, (0, -157)이 된다. 그런 다음 마지막으로, 이 값을 12개의 이진 수 (000000000000, 111101100011)로 변환시킨다. NN에 적용된 입력 벡터는 000000000000111101100011이 된다.

<537> 이제, 이진 및 부동-소수점 데이터가 설명되었고, 기호 데이터 (symbolic data)에 대해 논의할 것이다.

<538> **기호 데이터:** 기호 데이터는 비수치 데이터로, 계산 목적의 이진수도 부동 소수점도 아니다. 그것은, 상기 데이터가 갖는 일부 비수량적 (non-quantifiable) 속성들이라 불릴 수 있다. 이제, 본 발명의 DR 훈련 알고리즘 (30)이 기호 데이터를 학습할 수 있는 바람직한 실시예를 제공한다.

<539> 성 (gender)과 같은 속성에 대해, 이 속성은 2개의 가능한 값을 가짐으로써, 단일 이진 입력이, 상기 데이터를 예를 들어 암 (female)은 1이고 수 (male)는 0으로 부호화하기 위해 할당될 수 있다. 다른 기호 속성들 또한 주어진 이진 값이 될 수 있다. 색 (colour)의 기호 속성의 조금 더 복잡한 예는 3개의 값을 가질 수 있다: 녹색, 남색 및 노란색. 이 속성의 값을 부호화하는 데 2개의 비트를 할당하는 것이 가능하다. 예를 들어, 2개의 비트가 이 필드를 부호화하는 데 할당될 수 있고, 이진 값이 임의적으로, 즉 01 - 녹색, 10 - 남색 및 11 - 노란색으로 할당될 수 있다. 대신 01 - 남색, 00 - 녹색 및 10 - 노란색 등처럼, 사용될 수 있었던 가능한 많은 다른 결합이 있을 수 있기 때문에, 이것이 많은 가능한 부호화 기법 중 유일한 것으로 이해되어야만 한다. 2개 대신 3개의 비트와 같은 다른 부호화 방법이 사용될 수 있다. 또한 비트 수는, 몇 개의 값들이 상기 속성에서 색에 대해 테스트될 수 있는지에 따라 다를 수 있다.

<540> 데이터 부호화 방법 선택할 때 주요 고려사항은, 분류되는 데이터 및 상기 데이터로부터 필요한 분류에 따라 다르다.

<541> 예를 들어, 만일 NN이 학습하는 데이터 속성 중 하나가 신용 기록이고 그것의 값이 낮음 (poor), 보통 (average), 높음 (good) 및 매우 높음 (excellent)이라면, 상기 훈련된 NN은 신용도가 낮은 고객으로부터 신용도가 높은 고객을 분리하는 것이다. 높은 신용도 기록을 가진 고객들과 대출을 상환하는 그들의 잠재적 능력 사이에 상관관계가 있다는 것이 추정될 수 있다.

<542> 상기 속성들에 할당될 수 있는 4개의 기호 값이 있기 때문에, 모든 4개의 값을 부호화하기 위해 오직 2개의 비트를 가질 필요가 있다. 상기 데이터에 필요한 출력에 관해 상기 값을 할당하는 데 케어 (care)가 선택되어야만 한다. 예를 들어, '낮음'은 01, '보통'은 10, '높음'은 11, 및 '매우 높음'은 00으로 부호화된다면, 만일 '높음' 및 '매우 높음'이 '낮음' 및 '보통'의 다른 값으로부터 분리된다면, 상기 데이터가 상기 입력 공간에서 XOR을 형성하고, 따라서 선형으로 분리될 수 없다. 비록 본 발명의 DR 훈련 알고리즘 (30)이 이것을 학습할 수 있더라도, 그것이 XOR과 동등하기 때문에, 상기 데이터를 학습하는 데 추가 LTG가 필요하다. 이것을 '기호 충돌 (symbolic conflict)'의 부호화라고 부를 수 있다. 상기 값이 상기 입력 공간에서 선형으로 분리될 수 있을 정도 다르게 부호화하여, 기호 충돌을 피할 수 있다. 예를 들어, '높음'은 11, '매우 높음'은 0, '낮음'은 00 및 '보통'은 01로 부호화하는 것이 이 문제를 피한다. 이 방법 '높음'/'매우 높음'은 '낮음'/'보통'으로부터 선형으로 분리된다. 데이터를 부호화하는 최고의 방법은 부호화되는 데이터에 따라 다르며, 그 결과 각각의 속성은 개별적으로 고려되어야만 한다.

<543> 하지만, 이것은 단순화 (simplification)인데, 이는 어느 고객이 신용도가 높은지를 예측하는 데 영향을 주는 신용 기록 이외의 추가 속성 관계들이 있다는 것이 추측되기 때문이다.

<544> 상기 출력과 속성 값 사이의 분명한 연관이 없을 수 있지만, 기호 충돌을 피하는 것이 항상 가능할 수는 없다. 본 발명의 DR 훈련 알고리즘 (30)은, NN에 필요한 대로 LTG를 추가할 수 있기 때문에, 기호 충돌을 학습할 수 있다.

<545> 지금껏, 본 발명의 DR 훈련 알고리즘 (30)에 의해 학습될 비-이진 데이터를 준비하는 방법이 조사되었고, 이제 그것의 실험 절차를 조사할 것이다.

<546> **실험 절차:** 상기 설명된 영역 (domain) 각각에 대해 동일한 실험을 수행하였다. NN을 데이터 세트로 훈련시키고, 상기 데이터 세트에 노출 수를 기록하였다. NN이 훈련 후에 보이지 않는 데이터를 얼마나 잘 일반화할 수 있는지 면에서, NN의 수행 능력을 평가하고, 그 결과를 공개된 동일한 데이터로 훈련된 NN의 결과와 비교하였다.

<547> 일단 훈련이 완료되면, NN을 테스트하여, 확실히 NN이 상기 훈련 세트에 대해 정확한 출력을 재생산하고 테스트 세트의 일부분을 분류할 수 있게 하였다.

<548> 각각의 NN에서 테스트된 데이터 세트는 앞에서 설명하였다. 상기 정의된 것과 동일하거나 유사한 바람직한 방법을 사용하여, 상기 데이터 세트를 이진수로 변환시켰다.

<549> **실험 결과**

<550> 이제, 2개의 데이터 세트에 대한 실험 결과를 조사할 것이다. 상기 훈련 및 테스트 단계로부터 중요한 2개의 결과 세트가 있다. 상기 훈련 단계에서, 몇 개의 훈련 패스 (passes)가 상기 데이터 세트를 학습하는 데 필요했는지가 중요하다. 상기 테스트 단계에서, 몇 개의 보이지 않는 입력 벡터가 성공적으로 분류되었는지가 중요하다.

<551> **이중-나선형 문제**

<552> **알려진 알로리즘의 공개된 결과:** 웨일랜드 (Weiland)는, 150,000-200,000 에포크 (epoch)의 변경된 역전파 버전으로 NN을 훈련시켰다. 하지만 표준 역전파로는 결코 해를 찾지 못했다. 하지만, 랑 및 위트브록 (Lang and Witbrock)은, 표준 역전파를 사용하여, 20,000 에포크의 데이터 세트를 학습하는, 2-5-5-5-1 구조, 2개의 입력, 5개의 은닉 유니트 각각의 3개의 은닉 층, 및 1개의 출력 유니트로 NN을 훈련시켰다. 하지만, 그들의 NN은, '단축 (shortcut)' 연결을 사용하여 모든 이전 층에 있는 모든 유니트로부터 직접적으로 입력을 받는 각각의 은닉 층 유니트를 가졌다.

<553> **본 발명의 DR 훈련 알고리즘의 실험 결과:** NN을 훈련시킨 데이터 세트 (170)는 도 17에 도시된다. 각각의 나선은 97개의 데이터 포인트를 갖는다. 그 결과로 얻은 훈련된 NN은 24개의 입력, 5개의 은닉 LTG들 및 1개의 출력 LTG를 갖는다. 상기 LTG를 모든 경우에 AND를 사용하여 함께 연결한다. 상기 NN은 단일 에포크로 학습하였다.

<554> **검토:** 생산된 NN (180)의 개략도는 도 18에 도시된다. 본 발명의 DR 훈련 알고리즘 (30)에 의해 단일 패스로 훈련된 NN (180) 및 미지의 입력 벡터에 대한 디폴트는 1을 출력한다. 그 결과로 얻은 상기 DR 학습의 NN (180)은 역전파에 필요한 것보다 간단한 표준 구조를 갖는다. 도 18에 도시된 것처럼, 출력 층 (184)에 있는 LTG T_{21}에 의해, 은닉 층 (182)에 있는 LTG들 (LTG_{11}, LTG_{12}, LTG_{13}, LTG_{14}, 및 LTG_{15})은 함께 모든 경우에서 AND로 연결되었다. NN (180)을 학습시킨 데이터 세트 (170)(도 17)는 각각 3개의 주기를 갖는 2개의 나선을 갖는다. 각각의 나선은 97개의 데이터 포인트들을 갖는다. NN (180)은 훈련 벡터를 100% 정확하게 회상할 수 있다.

<555> NN (180)을 각각의 나선에 대해 80개의 입력 벡터로 테스트하고, 어떠한 데이터 포인트도 상기 훈련 세트로부터 나오지 않았다. 1을 출력하는 나선으로부터의 입력 벡터들에서, 21/80 입력 벡터들이 부정확하게 분류되었다. 이것은 26%의 오차율을 낳는다.

<556> 0을 출력하기 위해 훈련된 나선으로부터의 80개의 입력 벡터들 중, 35/80 입력 벡터들이 부정확하게 분류되었다. 이것은 43%의 오차율을 낳는다. 이 결과가 매우 높은 이유는, 상기 디폴트 출력이, 상기 입력 벡터의 출력이 드러나지 않은 LTG에 대해 1이었기 때문이었다.

<557> 두 개의 나선에 대한 평균 오차율은 34.5%이다. NN (180)은, 그것이 상기 훈련 세트에 대한 수정 출력 (correction output)을 98%로 예측할 수 있을 때 훈련된 것으로 간주되기 때문에, 상기 2개의 나선 문제에 대해 유사한 오차율을 찾는 것은 어려웠다. 또한, 1은 > .5였고 0은 < .5였을 수 있기 때문에, 캐스케이드-상호작용 구조 (Cascade-Correlation Architecutre: CAS)에 대한 오차율을 찾는 것은 어려웠다. 텡 외 (Teng et al.)는, 뉴런이 > 8.일 때, 활성화되고, 그렇지 않으면 활성화되지 않는 것으로 간주한다. 푸 외 (Fu te al), 텡 외 (Teng et al), 및 팰만 외 (Fahlman et al)의 중요한 관심 대상은, 은닉 층에 있는 유니트의 수 및 NN을 훈련시키는 데 필요한 에포크의 수 또는 시간이었다. 모든 경우에, 본 발명의 DR 훈련 알고리즘 (30)은, 100% 정확하게 상기 데이터 세트를 학습하는 데 최소한의 은닉 유니트 수 및 오직 1 에포크를 필요로 했다. 상기 DR 훈련 알고리즘 (30)은 5개의 은닉 뉴런을 필요로 했고, CAS는 평균 12개의 은닉 뉴런 및 1700 에포크; 2082 ± 478 에포크 21.1 ± 2.3 하부-NN (1개 이상의 뉴런들); 및 10개의 은닉 유니트를 필요로 하고, 최소 1073.45 CPU를 학습하였다. 그것이 마지막 입력 벡터를 학습하기 위해 본 발명의 DR 훈련 알고리즘 (30)을 사용한 시간, 및 따라서 상기 이중-선형 문제에 대해 다른 입력 벡터를 학습하는 데 걸린 가장 긴 시간은 15분 54초였다. 이것은 제약 만족 라이브러리 (constraint satisfaction libraries)의 사용 때문이었고, 마지막 입력 벡터를 학습하는 은닉 층 (182)에 5개의 LTG들이 있었기 때문이었다. 입력 벡터를 테스트하는 평균 시간은 약 30분이었다. 상기 데이터 세트를 학습하는 시간은, 역전파로 학습한 시간보다 상당히 짧았는데, 그 이유는 상기 역전파로 훈련된 NN으 고정된-크기의 NN이 필요하기 때문이다.

<558> 오차율은 상기 데이터 세트 내부에 작았다. 오차율은 각각의 나선에 대한 첫 번째 1.5주기 내의 23% 및

33%였다. 오차율의 상승은, 상기 데이터 세트 내부에 있는 훈련 데이터 포인트들의 높은 밀도 때문이다.

<559> 하지만, 그것은 r = Θ인 나선을 예측하는 데 더 성공적이다. 도 19에, 상기 테스트 데이터 세트의 결과가 도시된다.

<560> 'ㅁ'의 곡선에 있는 '+'는, 정확히 확인된 입력 벡터이고, 'o'의 곡선에 있는 'x' 또한, 정확히 확인된 입력 벡터를 나타낸다. 그렇지 않으면, 그들은 부정확하게 확인된다. 여기서, 일반화하는 NN (180)의 능력을 향상시키기 위해, 추가 훈련 제공이 필요한 입력 공간의 부분들을 볼 수 있다. 이것은 전통적으로, 피드-포워드 NN을 훈련시키기에 매우 어려운 데이터 세트라는 것이 주목된다.

<561> 모든 입력 벡터에 대해, 마지막 입력 벡터를 제외하고, NN (180)의 은닉 층 (182)에 있는 하나 이상의 LTG들에 대한 불완전한 학습이 있었다. 매우 종종 LTG_{15}에 불완전한 학습이 있었다.

<562> 생산된 NN (180)은, 역전파를 사용할 때 생산된 NN보다 덜 복잡하다. 또한, 그것은 어려운 데이터 세트이다. 그 결과는, 늘어난 수의 입력 벡터로 훈련시켜 향상될 수 있는 것으로 여겨진다.

<563> 따라서, 피드-포워드 NN을 훈련시키는 다른 접근법뿐만 아니라, 본 발명의 DR 훈련 알고리즘 (30)은 일반화하는 그것의 능력에 대해 수행할 수 있다. DR 훈련 알고리즘 (30)은 유사한 구조를 훈련시키는 다른 훈련 방법이다.

<564> 이제, NN (180)에 있는 각각의 LTG들에 대한 MAV를, 각각의 LTG가 학습한 것을 결정하기 위해 조사할 것이다.

<565> **이중-나선형 문제에서 LTG들에 대한 MAV 결정**: LTG_{11} (임계치 T_{11}로)에 대해, 도 15의 방법 (150)을 사용하여 MAV를 찾는 것이 제약 수를 194개에서 29개로 줄였다. 이것은 85.1%의 감소 또는 압축이다. 앞서 논의한 것처럼, 이 LTG에 의해 학습된 다른 모든 입력 벡터는, 상기 LTG에 대한 MAV를 형성하는 남겨진 이 포인트들로부터 회상될 수 있다. 가중치-공간은 24차원을 갖는다.

<566> 상기 제약들 중, 13개의 입력 벡터가 $x_i \cdot w_{11} \geq T_{11}$을 형성하고, 다른 16개의 제약들이 $x_i \cdot w_{11} < T_{11}$을 형성한다.

<567> 상기 LTG가 학습한 것을, 상기 LTG를 활성화시키는 가중치-공간의 영역 면을 형성하는 평면을 조사하여 그래프로 나타낼 수 있다. 이러한 평면은 $x_i \cdot w_{11}$로 나타낸다. 상기 가중치-공간은 상기 입력 공간의 변형 (transformation)이다. 상기 LTG가 학습한 것을 결정할 수 있기 위해, 상기 MAV에 있는 제약들을, 그들을 형성시킨 입력 벡터로 다시 변환시킨다. 그런 후, 원래의 입력 벡터가 형성된 과정을 십진법 (decimal)에서 이진법으로 전환시킬 수 있다. 앞에, 부동-소수점 수를 준비하는 방법에 대해 논의한 부분에서, 이것을 수행하는 방법에 대해 논의하였다. 이제, 은닉 층 (182)에 있는 각각의 LTG가 본 발명의 방법을 사용하여 학습한 것을 설명할 것이다.

<568> 도 20a에서, 은닉 층 (182)에 있는 첫 번째 LTG, LTG_{11}이 학습한 것을 볼 수 있다. 'o' 및 'ㅁ'은 원래의 입력 벡터를 나타낸다. 상기 'ㅁ'은 r = Θ 나선을 나타내고, 상기 'o'은 r = -Θ을 나타낸다. '+'는 LTG_{11}에 대한 MAV를 찾은 후 남겨진 입력 벡터를 나타내고, 이 데이터 세트를 분류하는 데 두드러진 입력 벡터를 나타낸다. 'x'는, $x_i \cdot w_{11} < T_{11}$을 사용하여 학습한 제약들을 나타낸다. 상기 '+'는, $x_i \cdot w_{11} \geq T_{11}$을 사용하여 학습한 제약들을 나타낸다. 이 규정은 계속되는 LTG에 뒤따른다. 이것은, 상기 LTG_{11}이 정확하게 학습한 것이다. 또한, 입력-공간이 아닌, 가중치-공간에 정의된 체적 면으로 상기 데이터를 부호화하여, 상기 데이터 세트가 학습된다는 것은 주목할 만한 가치가 있다.

<569> 전통적으로, 상기 입력-공간은 학습되는 집합들을 선형으로 분리하기 위해 분석된다. 상기 가중치-공간은 24차원의 공간이기 때문에 자세히 조사하기가 불가능하다. 이것은 은닉 층 (182)에 있는 나머지 LTG들에 대해서도 마찬가지이다. 하지만, 이 LTG, LTG_{11}은 전체 데이터 세트를 학습할 수 없었다. 이제, LTG_{12}가 학습한 것을 조사할 것이다.

<570> LTG_{12}에 대해, 상기 MAV를 찾는 것이 제약 수를 194개에서 34개로 줄였다. 이것은 82.5%의 감소, 또는 압축이다. $x_i \cdot w_{12} < T_{12}$를 형성하는 16개의 제약을 제외하고, 모든 제약들이 $x_i \cdot w_{12} \geq T_{12}$를 형성하는 제약들을

생산하였다. 도 20b에서, 은닉 층 (182)에 있는 두 번째 LTG, LTG_{12}가 학습한 것을 볼 수 있다.

<571> 도 20a에 있는 것과 동일한 규정을 사용하여, LTG_{12}가 다른 입력 벡터를 학습한 것을 볼 수 있다. 'ㅁ' 나선, r = Θ 위에, 입력 벡터가 학습한 모든 것이 집합, 즉 $x_i.w_{12} \geq T_{12}$에 속한 것을 볼 수 있다. 하지만, 'o' 나선, r = -Θ 위에 많은 입력 벡터들이 있고, 이들 또한 상기 집합에 속한다. 이는, 이 LTG, LTG_{12}가 이 데이터 세트에 있는 모든 것을 학습할 수 없기 때문이다. 또한, 은닉 층 (182) 출력에 있는 LTG들은 LTG_{21}, 출력 LTG에 의해 AND로 함께 연결된다. 이는, 만일 상기 LTG, LTG_{12}가 상기 입력 벡터들에 대해 잘못된 결과, 즉 0 대신 1을 생산한다면, 이 층에 있는 하나의 다른 LTG가 상기 입력 벡터들을 학습할 수 있고 0을 생산한다. 이제 LTG_{13}이 학습한 것을 조사할 것이다.

<572> LTG_{13}에 대해, MAV를 찾는 것이 제약 수를 194개에서 51개로 줄였다. 이것은 제약의 73.7% 감소이다. $x_i.w_{13} < T_{13}$을 형성하는 10개의 제약들을 제외하고, 모든 제약들이 $x_i.w_{13} \geq T_{13}$를 형성하는 제약들을 생산하였다. 도 20c에서, 은닉 층 (182)에 있는 세 번째 LTG, LTG_{13}이 학습한 것을 볼 수 있다.

<573> LTG_{14}에 대해, MAV를 찾는 것이 제약 수를 194개에서 81개로 줄였다. 이것은 58%의 감소, 또는 압축이다. $x_i.w_{14} < T_{14}$를 형성하는 6개의 제약들을 제외하고, 모든 제약들이 $x_i.w_{14} \geq T_{14}$를 형성하는 제약들을 생산하였다. 도 20d에서, 은닉 층 (182)에 있는 네 번째 LTG, LTG_{14}가 학습한 것을 볼 수 있다.

<574> 이제 LTG_{15}가 학습한 것을 조사할 것이다. LTG_{15}에 대해, MAV를 찾는 것이 제약 수를 194개에서 159개로 줄였다. 이것은 18%의 감소, 또는 압축이다. 1개의 입력 벡터를 제외하고, 상기 LTG에 의해 학습된 모든 제약들이 $x_i.w_{15} \geq T_{15}$를 형성하는 제약들을 생산하였다. 도 20e에서, 은닉 층 (182)에 있는 마지막 LTG, LTG_{15}가 학습한 것을 볼 수 있다.

<575> 이 LTG, LTG_{15}의 주요 목적은, 'x'로 나타낸 입력 벡터를 학습하는 것이다. 또한 그것은, 'ㅁ'으로 나타낸 r = Θ 나선에 있는 많은 포인트들을 갖는다.

<576> 은닉 층 (182)에 있는 LTG들에 대해, 그들이 2개의 나선 곡선의 다른 부분들을 학습한 것을 볼 수 있다. NN (180)이 100% 정확하게 학습한 것을 재생산할 수 있다는 것을 상기하는 것이 중요하다.

<577> 또한, 상기 MAV는 LTG_{21}, 또는 NN (180)의 출력 LTG에 대해 찾아졌다. MAV가 그것의 모든 입력 연결들 사이에 AND를 형성함에 따라, 32개의 제약들이 있었다. 이것은 상기 MAV에서 6개의 제약들로 줄었다. 이것은 81%의 감소, 또는 압축이다.

<578> 함수 최소화는 MAV를 정의하기 위해 찾아진 입력 벡터에 적용될 수 없는데, 이는 상기 데이터가 독립적이기 때문에, 즉 상기 입력 벡터가 단일 값을 정의하고, 따라서 함수 최소화가 무의미한 정보를 나타낼 것이기 때문이다.

<579> **독일 신용 문제**

<580> 이 데이터 세트는 그것과 연관된 코스트 매트릭스 (cost matrix)를 갖는다. 이 코스트 매트릭스는 표 3에 나와있다. 하기 표의 세로는 예측된 집합 (predicted class)이고, 가로는 진짜 집합 (true class)이다. 만일 고객이 신용도가 높게 예측되고 대출 상환에도 높은 점수를 갖는다면 코스트가 없고, 마찬가지로 만일 고객이 신용도가 낮게 예측되고 대출 상환에도 낮은 점수를 갖는다면 코스트가 없다. 하지만, 만일 누군가 실제로는 신용도가 높음에도 불구하고 신용도가 낮게 예측된다면, 이는 대출기관 (lending institution)에 이자를 잃게 할 것이다. 하지만 더 좋지 않은 상황은, 고객이 실제로는 신용도가 낮음에도 불구하고 신용도가 높게 예측되는 경우이다. NN의 코스트를 계산할 때 고려될 필요가 있는 오차의 2개의 집합이 있다.

표 3

<581>

	높음	낮음
높음	0	1
낮음	5	0

<582> **독일 신용 문제 데이터 세트에 대한 코스트 매트릭스**

<583> **알려진 알고리즘의 공개된 결과:** 역전파에 대한 오차율은 표 4에 나와있다. 역전파가 상기 코스트 매트릭스에 필요한 대로 오차 집합 사이를 구별하지 못하기 때문에, 이 숫자는 상기 코스트 매트릭스를 포함하지 않는다.

표 4

<584>

	훈련 오차	테스트 오차
역전파	0.446	0.772

<585> **역전파에 대한 오차율**

<586> 이 데이터 세트로 역전파를 사용하여 NN을 훈련시키고 테스트하는 데 필요한 시간은 표 5에 나와있다.

표 5

<587>

	훈련 시간 (초)	테스트 시간 (초)
역전파	5950.0	3.0

<588> **역전파에 대한 훈련 시간**

<589> **본 발명의 DR 훈련 알고리즘의 실험 결과:** 이 데이터 세트로의 훈련이, 하나의 출력 및 2개의 은닉 층 LTG를 갖는 NN을 생산하였다. 상기 은닉 층 LTG들을 OR 연결을 통해 함께 연결하였다. 상기 데이터 세트에 1000개의 입력 벡터들이 있다. 입력 벡터 100개의 테스트 세트를 상기 벡터 1000개의 데이터 세트로부터 무작위로 선택되었다. 훈련 후에 생산된 NN (190)의 개략도는 도 21에 도시된다.

<590> 테스트 목적으로 남겨둔 100개의 입력 벡터들 가운데, 출력에 대해 0을 생산하는 89개의 입력 벡터들 중 부정확하게 확인된 4개의 입력 벡터들이 있다. 따라서, 85개의 입력 벡터들이 정확하게 확인되었다. 상기 테스트 세트에서 출력에 대해 1을 생산하는 11개의 입력 벡터들 중 정확하게 확인된 9개의 입력 벡터들이 있다. 따라서, 2개의 입력 벡터가 부정확하게 확인되었다. 이 결과를 표 6에 요약하였다.

표 6

<591>

	부정확	정확	총	퍼센트
출력 1	2	9	11	18%
출력 0	4	85	89	4.4%
총	6	94	100	6%

<592> **독일 신용 문제에 대한 결과 요약**

<593> 1을 생산하는 입력 벡터들에 대한 오차율은 조금 높은 18%로 나타났다. 0을 생산하는 입력 벡터들에 대한 오차율은 4.4%이다. 총 오차는 6%이다. 추가 훈련이, 상기 두 종류의 출력을 지닌 입력 벡터에 대한 오차율을 감소시키는 것으로 여겨진다. NN (190)은 100% 정확하게 학습한 것을 재생산할 수 있다.

<594> 모든 오차 결과는 역전파에 대한 오차 결과 (훈련 오차: 0.446 및 테스트 오차: 0.772)보다 나았다. 훈련 및 테스트 결과는 표 4에 나와있다.

<595> 이 실험은, 본 발명의 DR 훈련 알고리즘 (30)이 역전파와 같은 종래의 알려진 훈련 알고리즘보다 상당한 이점을 제공한다는 것을 보여준다. 가중치가, 학습하고자 하는 데이터 세트에 내재한 규칙들을 부호화하려는 평균 값을 나타내기 때문에, 상기 오차는 역전파에 대해 결코 0으로 줄어들 수 없다. 역전파를 사용할 때 각각의 퍼셉트론에 있는 단일 가중치 값은, 각각의 입력 벡터 대한 정확한 출력 생산을 학습하게 만들 각각의 입력 벡터에 대해, 필요한 가중치 값을 정확하게 나타낼 수 없다. 이것은, 가중치-공간이 각각의 입력 벡터에 대해 정확한 출력을 허용하기 때문에, 상기 가중치-공간의 영역을 찾는 이점들 중 하나이다.

<596> NN (190)을 총 268개의 입력 벡터로 훈련시키는데, 이들 입력 벡터 중 168개가 1을 생산했고, 다른 100

개의 입력 벡터가 0을 생산했다. 이들 입력 벡터들을, 테스트를 위해 사용되지 않은 나머지 900개의 입력 벡터들로부터 무작위로 선택하였다. 더 많은 입력 벡터들이 테스트를 위해 사용될 수도 있었다.

<597> NN (190)을 훈련에 이용가능한 데이터 세트의 < 1/3로 훈련시켰고, 이들은 역전파에 대한 오차율보다 작은 오차율을 생산하였다.

<598> 이 실험의 결과는, 만일 본 발명의 DR 훈련 알고리즘 (30)이 실제 데이터 세트에 사용된다면, 빠른 제약 테스트 방법이 바람직하다는 것을 보여주었다.

<599> 또한, 상기 실험의 결과는, 이미 학습된 입력 벡터의 수에 따라 증가된 각각의 입력 벡터를 학습하는데 필요한 시간을 보여주었다. 이것을 향상시킬 수 있는 방법은 적어도 2가지가 있다: (1) 병렬 처리기 (parallel processor) 사용; 또는 더 효과적인 제약 테스트 알고리즘 사용. 또한, 비싼 최신식 처리기가 DR 훈련 알고리즘 (30)의 작동을 향상시킬 수 있는 것으로 여겨진다.

<600> 테스트를 위해 남겨둔 입력 벡터 100개 중, 11개의 입력 벡터가 1의 출력을 생산하고, 다른 89개의 입력 벡터가 출력으로서 0을 생산한다. 상기 총 데이터 세트에서 1을 생산하는 입력 벡터는 비례적(proportionally)으로 거의 없었다. 하지만, 상기 출력이 되어야만 하는 것이 드러나지 않았을 때, 0을 출력하는 쪽으로의 치우침 (bias)이 있기 때문에, 1을 출력하는 비례적으로 더 많은 수의 입력 벡터로 훈련시키는 것이 결정되어, NN (190)이 성공적으로 1을 출력하는 방법을 학습할 것이다. 1보다는 오히려 0을 출력하는 치우침이 선택되는데, 그 이유는, 음성 오류 (false negative) 대 양성 오류 (false positive)에 대한 5 대 1 선호가 있기 때문이다. 이것은, 음성 오류 : 양성 오류 = 5:1의 분류 오차를 선호하는 것을 나타내는 코스트 매트릭스를 기초로 한다. 즉, 실제로는 신용도가 낮임에도 불구하고 그 고객을 신용도가 높은 사람으로 잘못 분류하기보다는 오히려, 실제로는 신용도가 높음에도 불구하고 그 고객을 신용도가 낮은 사람으로 분류하는 것을 선호할 것이다.

<601> 따라서, 표 3에서 코스트 매트릭스를 적용할 때, 그 코스트는 14이다. NN을 테스트할 때 오직 평균 오차가 수집되기 때문에, 종래의 NN 훈련 알고리즘으로 상기 코스트를 결정하는 것은 불가능하다.

<602> 비록 상기 훈련 시간이, 제약들을 테스트하는 라이브러리 함수 (library function)의 결과로 역전파에 대한 시간보다 더 길더라도, 본 발명의 DR 훈련 알고리즘 (30)은, 상기 입력 벡터를 학습하는 데 오직 하나의 데이터 세트 패스를 필요로 했고, NN (190)을 훈련시키는 데 이용가능한 데이터 세트의 오직 3/1을 필요로 했다.

<603> **독일 신용 문제에서 LTG에 대한 MAV 결정:** 도 15의 MAV를 결정하기 위한 방법 (150)을 사용하여, NN (190)에 있는 LTG들에 대한 MAV를 찾았다. 훈련 동안 LTG, 상기 층에 있는 마지막 LTG 또는 현재의 출력 LTG에 대한 MAV를 찾는 것은, NN (190)이 학습한 것을 잊을 것이라는 것을 의미할 수 있다. 앞서, LTG를 NN에 추가할 때 논리를 실행하는 방법을 논의한 부분에서 설명한 것처럼, 이는 제약들이 층에 새롭게 추가된 LTG로 복사될 때 변경되기 때문이다.

<604> LTG_{11}에 대한 MAV를 찾은 후, 훈련 동안 형성된 268개의 제약들 중 45개의 제약들이 남았다. 이것은 상기 LTG에 대한 가중치-공간을 정의하는 제약들의 83% 감소이다. 이들 제약들 중, 18개가 $x_i . w_{11} \geq T_{11}$ 형태의 제약들을 생산했다.

<605> 훈련 동안 형성된 제약들을 검사할 때, 상기 LTG가 학습한 것은 ($x_i . w_{11} \geq T_{11}$ OR $x_{i+1} . w_{11} \geq T_{11}$ OR ... OR $x_{i+n} . w_{11} \geq T_{11}$) AND NOT ($x_j . w_1 < T_{11}$) AND NOT ($x_{j+1} . w_{11} < T_{11}$) AND NOT ...의 형태로 읽힐 수 있다.

<606> 그것이 이러한 형태에 있기 때문에, 논리 분석에 적합하고, 이 데이터 세트 분류와 무관한 변수들이 유도될 수 있다.

<607> 상기 입력이 여기에 있는 일반적인 경우와 마찬가지로 독립적이라면, 특정 관심사인 변수들을 찾기 위해, 퀸-맥클러스키 또는 반복된 합의와 같은 함수 최소화 기법을 사용한다. 모든 잉여 입력 벡터들이 상기 MAV를 찾아 제거될 수 있기 때문에, 함수 최소화 기법을 사용하는 일을 훨씬 더 쉽게 만들고, 그들의 잠재적 지수 복잡성 (exponential complexity)을 극복하는 데 도움을 준다.

<608> 하지만, 원래의 데이터 세트에 있는 일부 변수들이 다수의 비트 위치로 변환되기 때문에, 상기 제약들이 나타내는 것을 아는 것이 더 흥미롭다.

<609> 두 번째 LTG, LTG_{12}를 은닉 층 (192) 및 새로운 출력 LTG에 추가시키는 것은 입력 벡터 [10001110101010000010001100110101010001010001] → 1이었다. 상기 벡터를 필드 [10 00 111 0101 01 000 001 00 011 00 11 01 01 00 01 0 10 00 1]로 나누었다. 이 벡터는 다음과 같이 번역된다: '이 고객은 당좌 예금 (checking account)이 없고, 대출은 < 12개월이 되고, 모든 기존의 대출금이 제때에 상환되었고, 가구/장비 마련을 위해 대출을 희망하고, 1000 내지 5000 DM을 대출받기 희망하고, < 1년 동안 직장에서 근무하고 있고, 가처분 소득 (disposable income) 비율이 < 1%이고, 기혼/이혼 여성이고, 보증인이 없고, 4년 이상 동일한 주소에 거주하고 있고, 약간의 저축 또는 생명 보험이 있고, 25세 내지 40세의 여성이고, 할부 구매가 없고, 집을 소유하고 있고, 이 은행에 다른 채무가 없고, 숙련된 근로 여성이고, 누군가를 부양할 의무가 없고, 자신 명의의 전화기가 없고, 외국인 근로자이다.'

<610> LTG_{12}에 대한 MAV를 찾은 후, 훈련 동안 형성된 268개의 제약들 중 139개가 남았다. 이것은 상기 LTG에 대한 가중치-공간을 정의하는 제약들의 48% 감소이다. 이들 제약들 중, 14개가 $x_i . w_{11} \geq T_{12}$ 형태의 제약들을 생산했다.

<611> 상기 출력 LTG, LTG_{21}은 그것의 입력 연결들 사이에 OR을 형성하였다. 그 결과, 그것은 4개의 제약들을 형성하였다. 상기 MAV를 결정한 후, 상기 제약 수가 3개로 줄어들었다. 이것은 제약 수의 25% 감소이다.

<612> 이와 같은 데이터 세트에 대해, 42차원으로, 이 과정을 자동화하는 것이 매우 바람직하다. 은닉 층 (192)에 있는 각각의 LTG에 대한 MAV를 찾은 후에라도, 검사하기 위한 45개 및 139개의 제약들 또는 규칙들이 있고, 이 과정을 자동화하지 않고는 그것이 어려울 수 있다. 하지만, 신용도가 높은 고객인지를 결정하기 위해, 상기 LTG가 학습한 것을 기초로 한 대략적 규칙은 다음처럼 무언가로 말해질 수 있다: '그들 자신의 집을 소유하거나 임대하지 않거나 재산/저축이 있거나 (많은 부채 및 보증인)'

<613> 하지만, NN (190)이 상기 테스트 세트로부터 2개의 입력 벡터를 정확하게 분류하지 않았다면, NN (190)이 아직 학습하지 않은 데이터 세트에 적어도 하나의 추가 특징이 있다.

<614> **상기 2개의 데이터 세트에 대한 실험 결과 요약**

<615> 상기 2가지 경우에서, NN (180),(190)이 100% 정확하게 훈련 세트를 재생산할 수 있다면, 비교 기준을 기초로, 상기 경우들에서 학습된 규칙의 정확도는 매우 높다. 이것은, 훈련 동안 역전파가 확인한 평균 가중치 값과 대조된다. 역전파를 사용하는 경우, NN을 훈련시킨 데이터에서 NN이 테스트될 때, 출력에는 반드시 오차가 어느 정도 있을 것이다. 입력 벡터가 분류되는 속도 (speed)는, 실행하기 위해 루틴 (routine)을 처리하는 제약들에 필요한 시간을 기초로 한다. 또한, 상기 알고리즘이 라이브러리를 처리하는 제약들에 의존한다면, 데이터 세트를 학습하는 시간은 비교적 느리다. 하지만, 상기 알고리즘 (30)을 수행하는 데 사용된 적당한 코드 (code) 및/또는 하드웨어는 1초 내에 입력 벡터를 학습할 것이다. 또한, 상기 데이터 세트는 그것의 단일 패스로 학습될 수 있다. 이것은, NN이 상기 데이터 세트를 언젠가 학습할 것인지가 드러나지 않는 역전파와 대조된다.

<616> 또한, 훈련 동안 학습된 규칙들은 매우 이해가능했다는 것을 볼 수 있었다. 반면, 역전파와 같은 수치 알고리즘으로 훈련된 NN에 부호화된 규칙들은 거의 완전히 이해불가능하다.

<617> 따라서, 본 발명은 피드-포워드 NN의 사용과 연관된 많은 이점들을 제공한다. 주요 이점들은 다음과 같이 요약될 수 있다: (a) 뉴런, 바람직하게는 LTG를 훈련시키는 신규한 방법; (b) 신규한 LTG 훈련 방법을 기초로 피드-포워드 NN을 훈련시키는 알고리즘 (30), 즉: (i) 데이터 세트를 학습하기 위해 필요한 대로 LTG를 동적으로 할당; (ii) 단일 패스로 학습; 및 (iii) 훈련 동안 학습된 규칙이 그 결과로 생긴 NN으로부터 쉽게 읽히도록 결정하기 위한 간단한 방법 (150); 및 (c) 상기 LTG가 학습한 것을 분석하는 간단한 방법 (130).

<618> 상기 신규한 LTG 훈련 방법은 각각의 LTG의 가중치와 임계치 사이의 관계를 찾는다. 이를 통해, LTG가 입력을 정확히 학습하고 회상한다. 즉, 종래의 뉴런 훈련 방법들이 생산하길 희망하는 근삿값 (approximate value) 대신, 상기 입력이 정확히 재생산될 수 있다.

<619> 본 발명에 따른 LTG 훈련 방법은, 상기 LTG를 활성화시키고, LTG로의 입력들 사이의 관계를 체적 면으로 부호화하는 각각의 LTG의 가중치-공간에서 체적을 찾는다.

<620> 상기 방법은, LTG가 입력 벡터를 학습하거나 학습할 수 있는지, 즉 입력 벡터를 분류하는 방법을 아는지 결정하기 위해 조사받도록 한다. 이 테스트는, NN에 LTG를 할당하는 것이 필요할 때를 결정하는 데 쉬운 방법을 제공하다.

<621> 상기 방법은, LTG가 종래의 방법으로 훈련된 뉴런들이 수행하는 모든 기능들, 예를 들면 이전에 학습된 입력을 회상하고 일반화하는 것을 수행하도록 한다.

<622> 본 발명에 따라 뉴런, 바람직하게는 LTG를 훈련시키는 신규한 훈련 방법의 주요 애플리케이션은, 데이터 세트를 학습하기 위해 NN에 필요한 대로 뉴런을 할당하는 DR 훈련 알고리즘 (30)의 개발이다. 이것은 NN 분야에 있어 상당한 공헌이다.

<623> 이러한 피드-포워드 NN 훈련 방법은, NN에 너무 많거나 너무 적은 뉴런을 가질 수 있는 고정된-크기의 NN 문제를 해결한다.

<624> 본 발명의 DR 훈련 알고리즘 (30)의 가장 중요한 특징 중 하나는, 단일 패스로 데이터 세트를 학습하는 그것의 능력이다. 이것은 잠재적으로 급격한 훈련 시간의 문제를 제거함에 따라, NN 분야에 대한 주요 공헌이다. 입력이 학습되거나 학습될 수 있는지를 결정하기 위해, 상기 LTG 조사에 필요한, 소프트웨어를 처리하는 제약들의 속도에 따라 훈련 시간이 다르며, 그것은 NN이 결정적 (deterministic) 시간 내에 학습할 것을 의미한다.

<625> 또한, 훈련에 앞서 데이터를 적당한 포맷으로 변환시키는 유용한 방법이 제공되는데, 이 방법은 NN의 훈련 시간을 개선하는 데 이용될 수 있다. 마찬가지로, 사전-분류 데이터가 NN에 의해 훈련되도록 하는 유용한 방법이 제공되는데, 이 방법은 데이터 분류 능률을 향상시키는 데 이용될 수 있다. 이러한 방법들은 모든 NN 훈련 알고리즘에 유용한 것으로 여겨진다.

<626> 본 발명의 또 다른 주요 이점은, 피드-포워드 NN의 작동, 특히 훈련 과정 동안 학습된 규칙들에 대한 통찰 (insight)을 제공하는 것이다. 상기 훈련 방법이 상관적임 (relational)에 따라, 그것이 입력 사이의 관계들을 찾는 것을 의미하고, 이러한 관계들은, 상기 LTG를 활성화시키는 체적의 가중치-공간의 영역 면으로 저장된다. 하나의 유용한 방법 (150)이 제공되는데, 이 방법은 MAV를 찾아 상기 관계들이 회복되도록 하고, 종래의 민감도 분석을 수행하는 데 사용될 수 있다. 또한, LTG를 층에 연결시키기 위해 사용된 논리적 관계들은 NN으로부터 직접적으로 읽힐 수 있다.

<627> 피드-포워드 NN에 대한 종래의 훈련 방법은, 훈련 동안 학습된 규칙들을, 상기 데이터 세트에 대한 많은 정보가 손실된 단일 숫자 값으로 압축한다. 종래의 훈련 방법으로부터, 상기 숫자 값이 그것이 나타내고자 하는 가능한 최고의 평균을 얼마나 정확히 나타내는지를 결정하는 것은 불가능하다.

<628> 본 발명의 DR 훈련 알고리즘 (30)은, 모든 입력 벡터들을 제약들로 변환시키고, 상기 제약들 내에 포함된 관계들을, 상기 LTG를 활성화하는 가중치-공간의 체적 면으로 저장하는 것이 바람직하다. 이것은 모든 입력이 회상되도록 하고, 본 발명의 또 다른 양태에 따라 MAV를 결정하는 방법 (150)을 사용하여, 제약 세트를 최소의 제약들로 줄일 수 있는 방법을 제공한다. LTG의 MAV를 찾음으로써, 상기 LTG의 제약 세트로부터 제약들이 제거될 때 어떠한 정보도 손실되지 않는다.

<629> 상기 MAV를 찾는 방법 (150)은 NN에 한정되지 않는다. 상기 MAV를 찾는 방법은 또한, 최적화에 사용된 CSP 및 운영 연구 유형 문제와 같은 제약들의 시스템을 사용하는 다른 분야에 유용한 것으로 여겨진다.

<630> 수행된 실험은, 훈련될 NN에 있는 가중치들보다 더 많은 입력 벡터들이 NN을 훈련시키기 위해 반드시 필요한 것은 아니라는 것을 보여준다. 이는, 각각의 입력 벡터가 NN에 있는 각각의 가중치를 훈련시키기 때문이다. DR 학습은, 입력 벡터로 하여금 LTG가 NN에 추가되는 것을 쉽게 확인하기 위해 간단한 테스트를 제공한다. 항상 가득 찬 데이터 세트로 훈련시킬 필요는 없다.

<631> 앞서, NN이 일반화에 실패하는 잠재적 원인을 논의한 부분에서, NN이 불충분하게 훈련되어, 그 결과 보이지 않는 입력 벡터의 출력을 알지 못한다는 것을 언급하였다. 반대로, 본 발명의 또 다른 양태에 따라, 방법 (130)이 제공되는데, 이 방법은, NN이 보이지 않는 입력 벡터에 대한 출력에 대해 아는지, 그리고 미지의 입력 벡터를 분명히 확인할 수 있는지를 결정하는 데 사용될 수 있다. 따라서, NN은, 추가 훈련이 필요한 입력 벡터를 확인할 수 있다.

<632> 입력 벡터가 드러나는지 드러나지 않는지 결정하는 방법 (130)은 NN에 한정되지 않는다. 또한 입력 벡터를 분류하는 방법은, 예를 들어 DNA와 같은 데이터의 문자열 분석과 같은 제약들의 시스템을 사용하는 다른 분야에 유용한 것으로 여겨진다. 마찬가지로, 입력 벡터를 분류하는 방법은 CSP 및 운영 연구 애플리케이션에 또한 사용될 수 있다.

<633> 이상 설명한 바와 같이, 본 발명은 상술한 특정 실시예에 한정되지 않고, 본 발명의 광범위한 범위 및

요지를 벗어남 없이, 당업자에게 손쉽게 이용가능한 다양한 변형의 실시가 가능하고, 그와 같은 변경은 특허청구 범위 내에 있는 것으로 이해될 것이다.

<634>

도면의 간단한 설명

<118> 이제, 본 발명이 더 명확하게 이해되고 실제적 효과를 거둘 수 있도록, 본 발명에 따른 NN 훈련 방법 및/또는 시스템의 바람직한 구성이 자세히 설명될 것이다. 본 발명은 첨부된 도면을 참조하여 비-제한적인 실시예들로 상세히 설명된다.

<119> 도 1은 2-입력, 1-출력 피드-포워드 NN의 기본 구조 예를 개략적으로 도시한다;

<120> 도 2는 본 발명의 바람직한 실시예에 따라 만들어진, NN 훈련 방법을 도시한 흐름도이다;

<121> 도 3은 도 2의 NN 훈련 방법에 따라 훈련된 NN의 출력에 대한 단일 패턴을 학습하는 바람직한 방법을 도시한 흐름도이다;

<122> 도 4는 도 2의 NN 훈련 방법에 따라 훈련된 NN의 은닉 층에 새로운 뉴런을 할당하는 바람직한 방법을 도시한 흐름도이다;

<123> 도 5는 도 2의 NN 훈련 방법에 따라 훈련된 NN에 새로운 출력 뉴런을 할당하는 바람직한 방법을 도시한 흐름도이다;

<124> 도 6a 및 6b는 본 발명의 NN 훈련 방법의 바람직한 실시예에 따라 NN에 새로운 뉴런을 할당하는 방법을 개략적으로 도시한다;

<125> 도 7은 2-입력 LTG NN의 기본 구조 예를 개략적으로 도시한다;

<126> 도 8은 모듈로 (Modulo)-8-문제를 사용하여, 본 발명의 NN 훈련 방법에 따라 훈련된 세 개의 출력을 지닌 NN의 바람직한 실시예를 개략적으로 도시한다;

<127> 도 9는 본 발명의 NN 훈련 방법의 바람직한 실시예에 따라 NN의 은닉 층에 뉴런을 할당하는 방법을 개략적으로 도시한다;

<128> 도 10a 및 10b는 본 발명의 NN 훈련 방법의 바람직한 실시예에 따라 NN에 새로운 출력 층을 추가하는 방법을 개략적으로 도시한다;

<129> 도 11은 3-입력 LTG NN의 기본 구조 예를 개략적으로 도시한다;

<130> 도 12는 본 발명의 바람직한 실시예에 따라 만들어진, 제약 세트의 입력 벡터가 드러나는지 드러나지 않는지를 결정하는 방법을 도시한 흐름도이다;

<131> 도 13은 훈련된 LTG의 가중치-공간의 일반화된 개략도를 도시한다;

<132> 도 14a 내지 14g는, 바람직한 실시예에 따라, 모듈로-8 문제를 해결할 수 있는 본 발명의 NN 훈련 방법으로 훈련된 NN을 개략적으로 도시한다;

<133> 도 15는 본 발명의 바람직한 실시예에 따라 만들어진, 제약 세트의 최소 활성화 체적 (MAV)을 결정하는 방법을 도시한 흐름도이다;

<134> 도 16a 내지 16e는 본 발명의 바람직한 실시예에 따라 NN을 훈련시키는 동안 학습된 다른 제약들뿐만 아니라 활성화 체적의 일반화된 개략도를 도시한다;

<135> 도 17은 NN의 훈련을 테스트하기 위해 사용된 인정된 데이터 세트인 이중 나선형 문제 (Two-Spiral Problem)에 대한 데이터 세트의 개략도를 도시한다;

<136> 도 18은 도 17의 테스트 데이터 세트 이중 나선형 문제를 해결하는 NN을 개략적으로 도시하며, 여기서 상기 NN은 본 발명의 NN 훈련 방법의 바람직한 실시예에 따라 생산된다;

<137> 도 19는 도 17의 이중 나선형 문제 테스트 데이터 세트로 훈련된, 도 18의 NN 결과의 개략도를 도시한다;

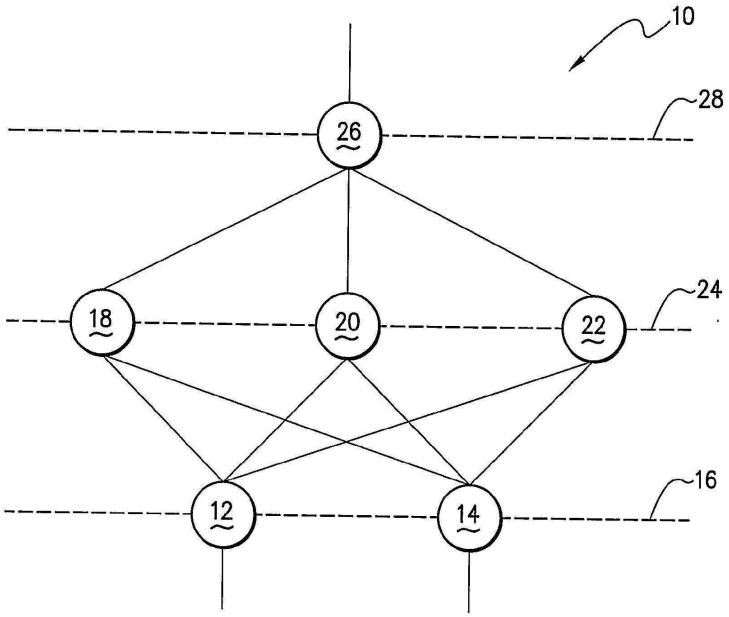
<138> 도 20a 내지 20e는 도 18의 NN 은닉 층에 있는 각각의 뉴런이, 도 17의 이중 나선형 문제 테스트 데이터 세트로

훈련됐을 때, 본 발명의 NN 훈련 방법을 사용하여 학습한 것의 개략도를 도시한다;

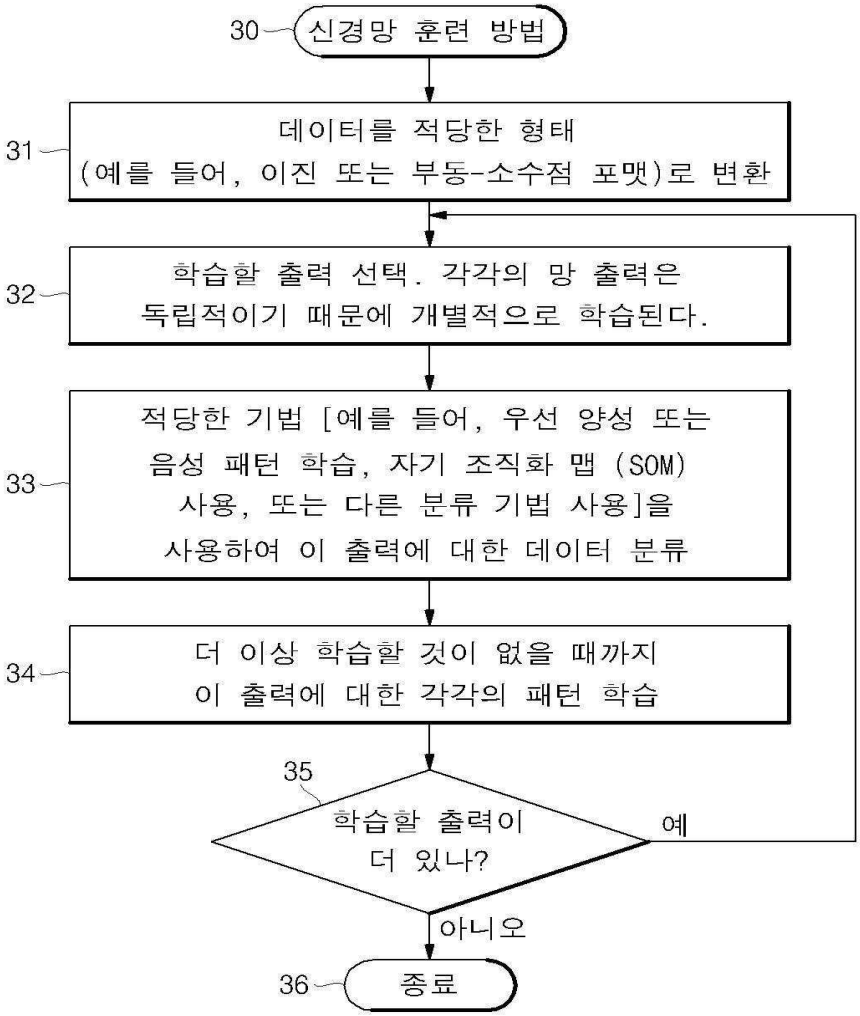
<139> 도 21은 독일 신용 데이터 세트 문제 (German Credit Data Set Problem)를 해결하는 NN을 개략적으로 도시하며, 여기서, 상기 NN은 본 발명의 NN 훈련 방법의 바람직한 실시예에 따라 생산된다.

도면

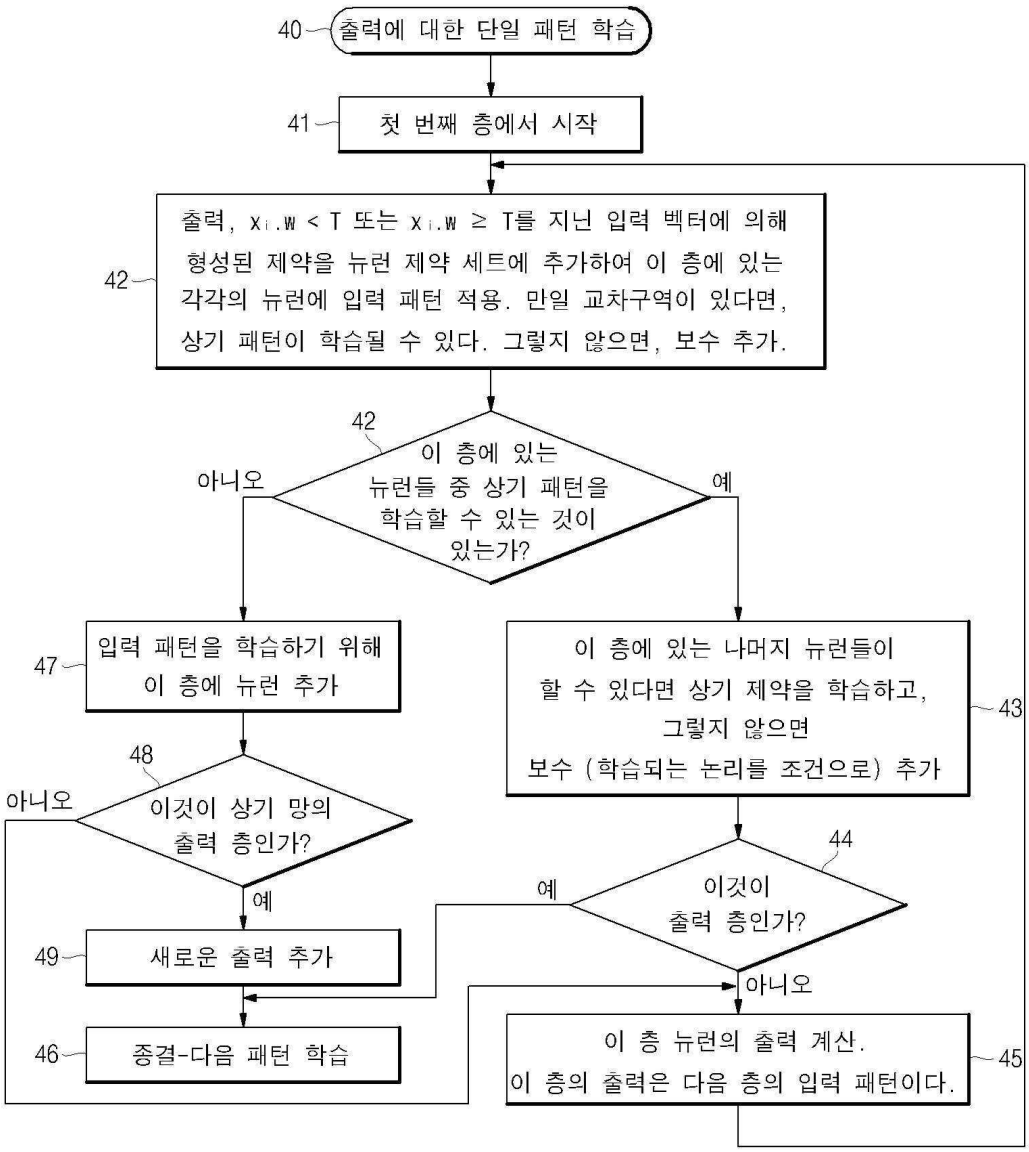
도면1



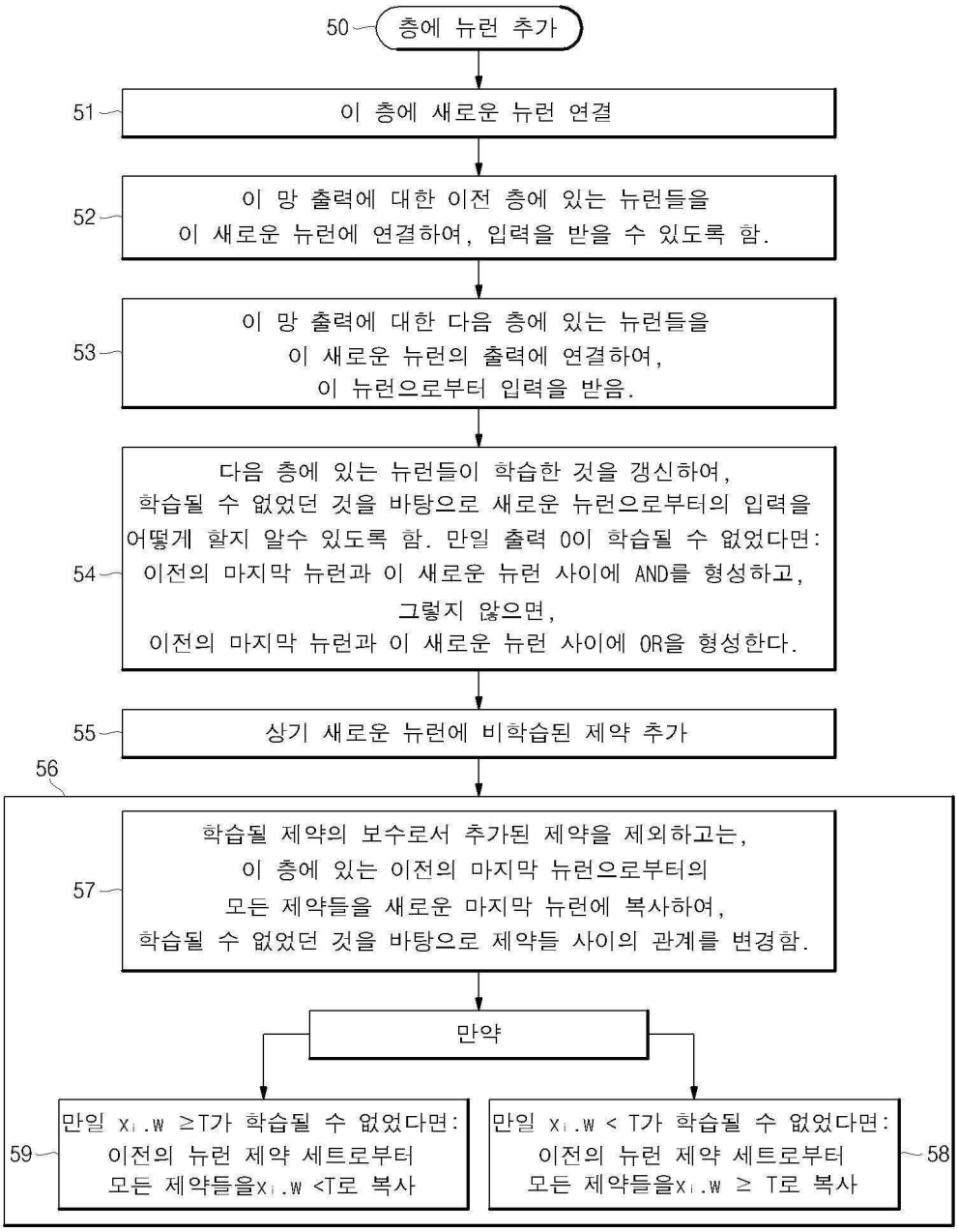
도면2


30
신경망 훈련 방법
31
데이터를 적당한 형태
(예를 들어, 이진 또는 부동-소수점 포맷)로 변환
32
학습할 출력 선택. 각각의 망 출력은
독립적이기 때문에 개별적으로 학습된다.
33
적당한 기법 [예를 들어, 우선 양성 또는
음성 패턴 학습, 자기 조직화 맵 (SOM)
사용, 또는 다른 분류 기법 사용]을
사용하여 이 출력에 대한 데이터 분류
34
더 이상 학습할 것이 없을 때까지
이 출력에 대한 각각의 패턴 학습
35
학습할 출력이
더 있나?
예
아니오
36
종료

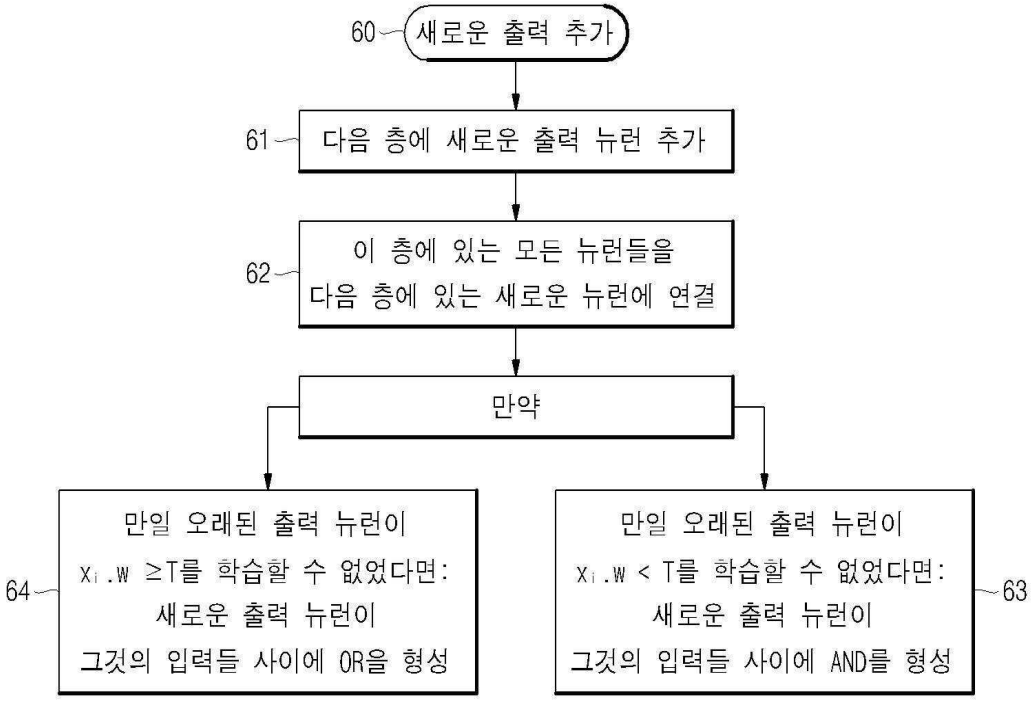
도면3


40
출력에 대한 단일 패턴 학습
41
첫 번째 층에서 시작
42
출력, $x_i.w < T$ 또는 $x_i.w \geq T$를 지닌 입력 벡터에 의해 형성된 제약을 뉴런 제약 세트에 추가하여 이 층에 있는 각각의 뉴런에 입력 패턴 적용. 만일 교차구역이 있다면, 상기 패턴이 학습될 수 있다. 그렇지 않으면, 보수 추가.
42
이 층에 있는 뉴런들 중 상기 패턴을 학습할 수 있는 것이 있는가?
아니오
예
47
입력 패턴을 학습하기 위해 이 층에 뉴런 추가
43
이 층에 있는 나머지 뉴런들이 할 수 있다면 상기 제약을 학습하고, 그렇지 않으면 보수 (학습되는 논리를 조건으로) 추가
48
이것이 상기 망의 출력 층인가?
아니오
예
44
이것이 출력 층인가?
예
아니오
49
새로운 출력 추가
46
종결-다음 패턴 학습
45
이 층 뉴런의 출력 계산. 이 층의 출력은 다음 층의 입력 패턴이다.

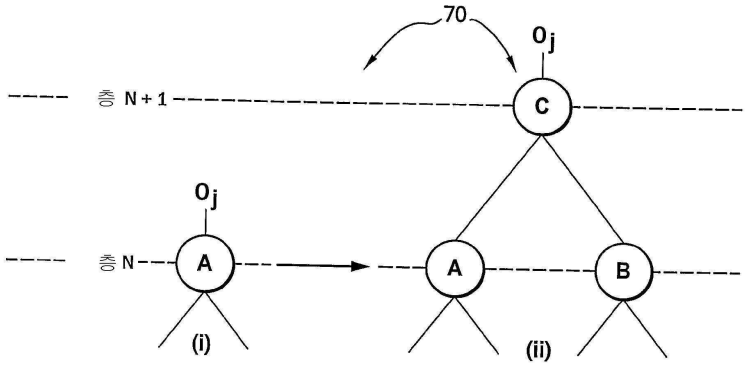
도면4


50
층에 뉴런 추가
51
이 층에 새로운 뉴런 연결
52
이 망 출력에 대한 이전 층에 있는 뉴런들을
이 새로운 뉴런에 연결하여, 입력을 받을 수 있도록 함.
53
이 망 출력에 대한 다음 층에 있는 뉴런들을
이 새로운 뉴런의 출력에 연결하여,
이 뉴런으로부터 입력을 받음.
54
다음 층에 있는 뉴런들이 학습한 것을 갱신하여,
학습될 수 없었던 것을 바탕으로 새로운 뉴런으로부터의 입력을
어떻게 할지 알수 있도록 함. 만일 출력 0이 학습될 수 없었다면:
이전의 마지막 뉴런과 이 새로운 뉴런 사이에 AND를 형성하고,
그렇지 않으면,
이전의 마지막 뉴런과 이 새로운 뉴런 사이에 OR을 형성한다.
55
상기 새로운 뉴런에 비학습된 제약 추가
56
57
학습될 제약의 보수로서 추가된 제약을 제외하고는,
이 층에 있는 이전의 마지막 뉴런으로부터의
모든 제약들을 새로운 마지막 뉴런에 복사하여,
학습될 수 없었던 것을 바탕으로 제약들 사이의 관계를 변경함.
만약
59
만일 $x_i.w \geq T$가 학습될 수 없었다면:
이전의 뉴런 제약 세트로부터
모든 제약들을$x_i.w < T$로 복사
58
만일 $x_i.w < T$가 학습될 수 없었다면:
이전의 뉴런 제약 세트로부터
모든 제약들을$x_i.w \geq T$로 복사

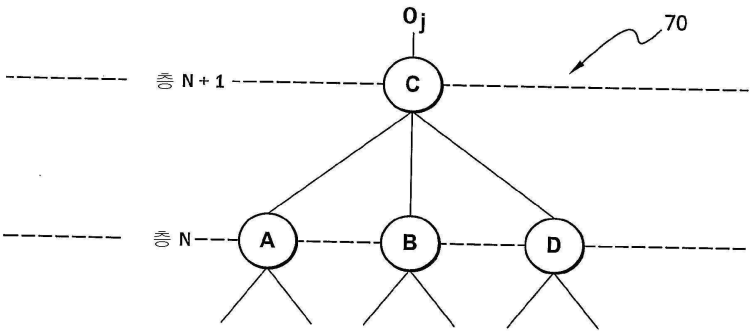
도면5


60
새로운 출력 추가
61
다음 층에 새로운 출력 뉴런 추가
62
이 층에 있는 모든 뉴런들을
다음 층에 있는 새로운 뉴런에 연결
만약
64
만일 오래된 출력 뉴런이
$x_i.w \geq T$를 학습할 수 없었다면:
새로운 출력 뉴런이
그것의 입력들 사이에 OR을 형성
만일 오래된 출력 뉴런이
$x_i.w < T$를 학습할 수 없었다면:
새로운 출력 뉴런이
그것의 입력들 사이에 AND를 형성
63

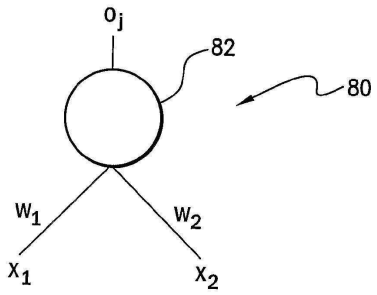
도면6a


70
O_j
층 N+1
C
O_j
층 N
A
A
B
(i)
(ii)

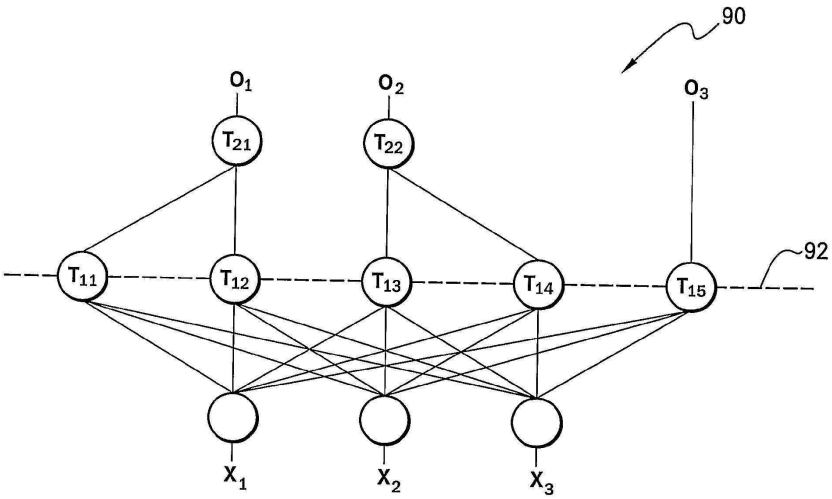
도면6b


O_j
70
층 N + 1
C
층 N
A
B
D

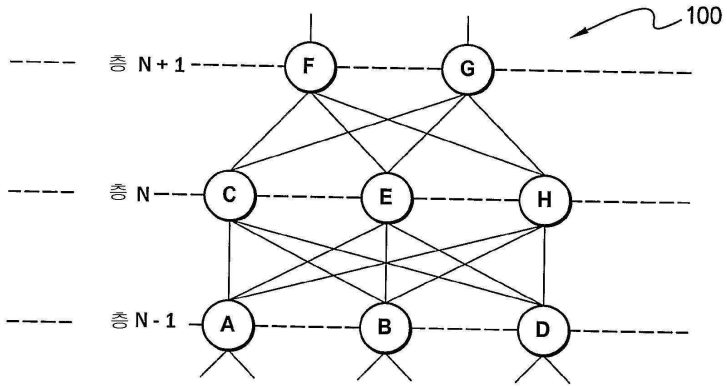
도면7


O_j
82
80
W_1
W_2
X_1
X_2

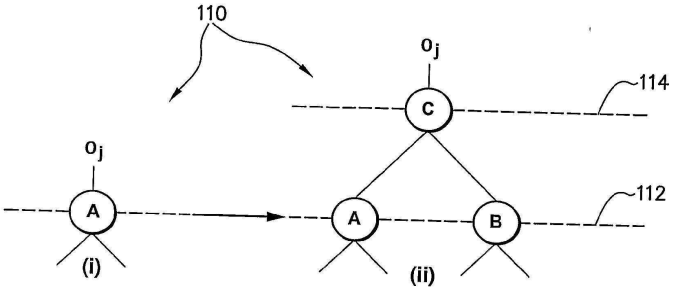
도면8


90
O_1
O_2
O_3
T_{21}
T_{22}
T_{11}
T_{12}
T_{13}
T_{14}
T_{15}
92
X_1
X_2
X_3

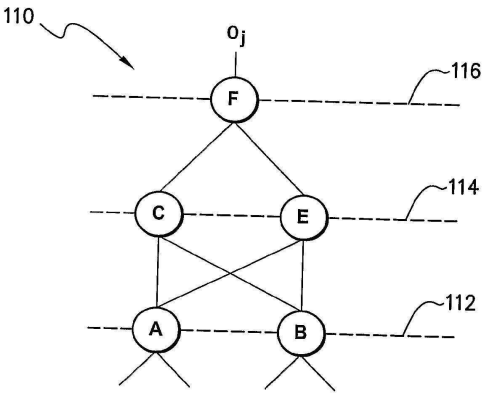
도면9



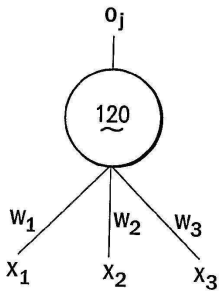
도면10a



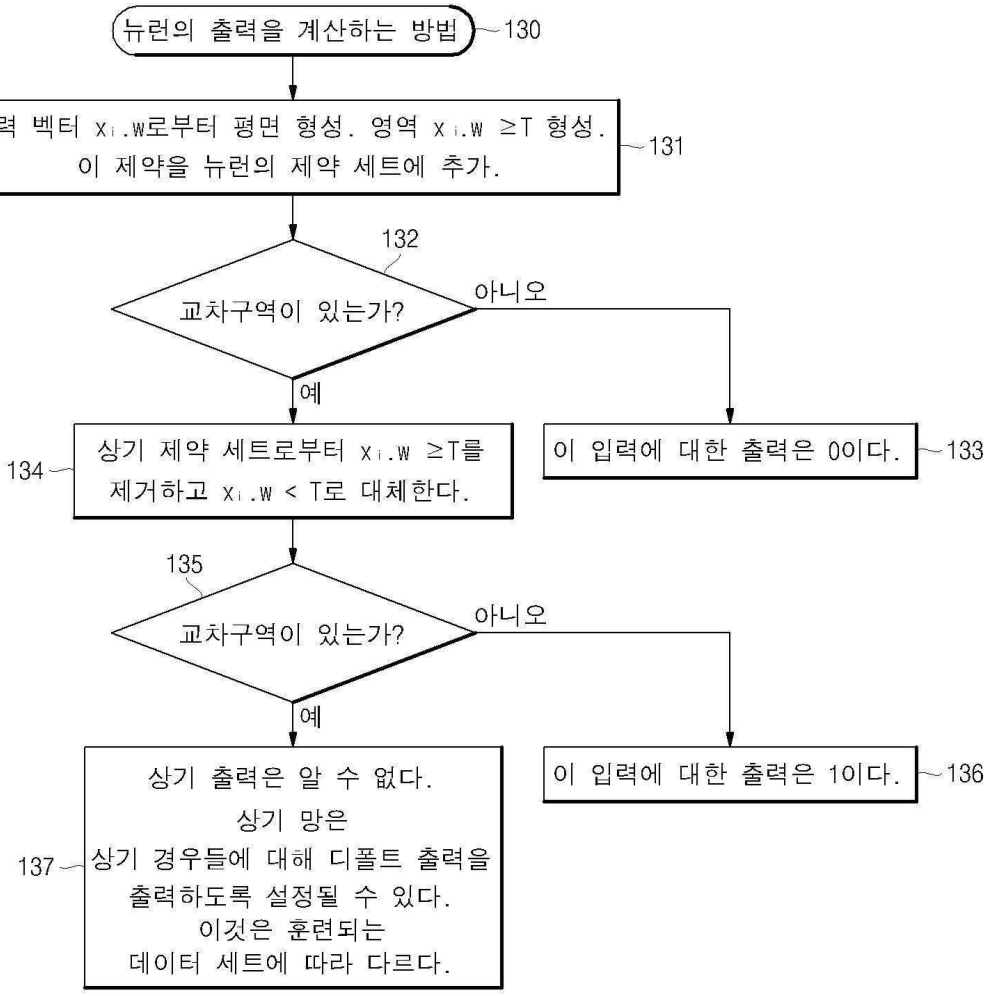
도면10b



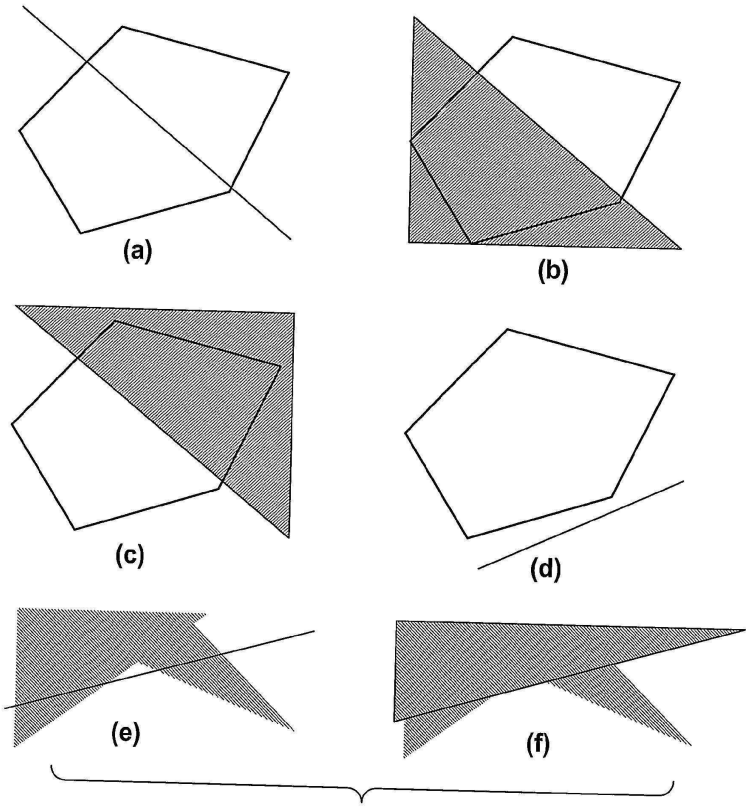
도면11


o_j
120
w_1
w_2
w_3
x_1
x_2
x_3

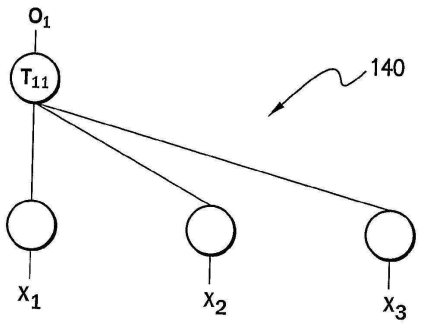
도면12


뉴런의 출력을 계산하는 방법
130
입력 벡터 $x_i.w$로부터 평면 형성. 영역 $x_i.w \geq T$ 형성.
이 제약을 뉴런의 제약 세트에 추가.
131
132
교차구역이 있는가?
아니오
예
상기 제약 세트로부터 $x_i.w \geq T$를
제거하고 $x_i.w < T$로 대체한다.
134
이 입력에 대한 출력은 0이다.
133
135
교차구역이 있는가?
아니오
예
상기 출력은 알 수 없다.
상기 망은
상기 경우들에 대해 디폴트 출력을
출력하도록 설정될 수 있다.
이것은 훈련되는
데이터 세트에 따라 다르다.
137
이 입력에 대한 출력은 1이다.
136

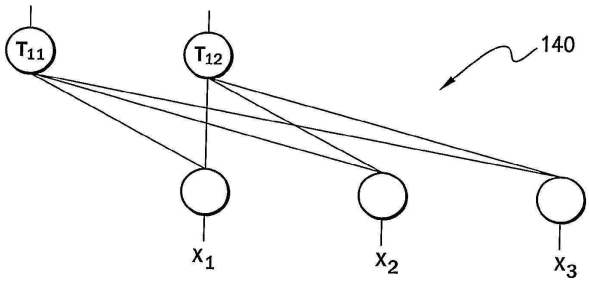
도면13



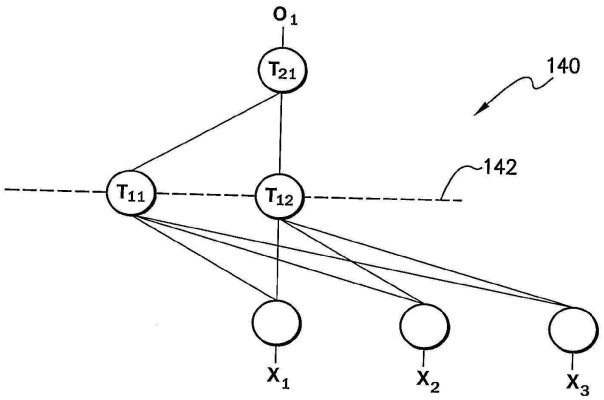
도면14a



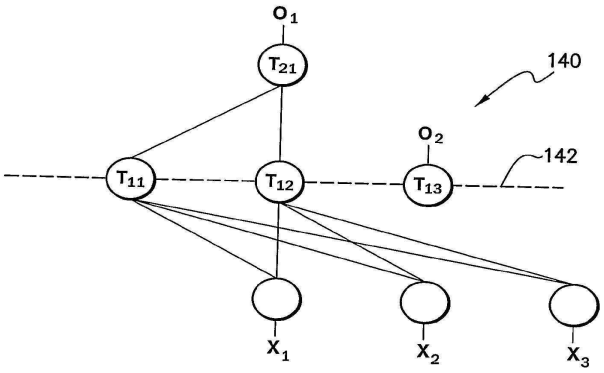
도면14b



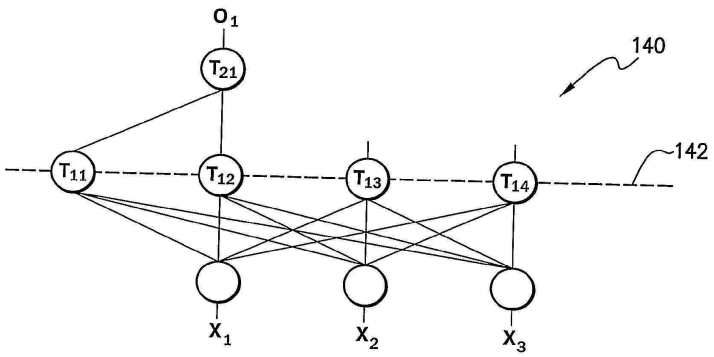
도면14c



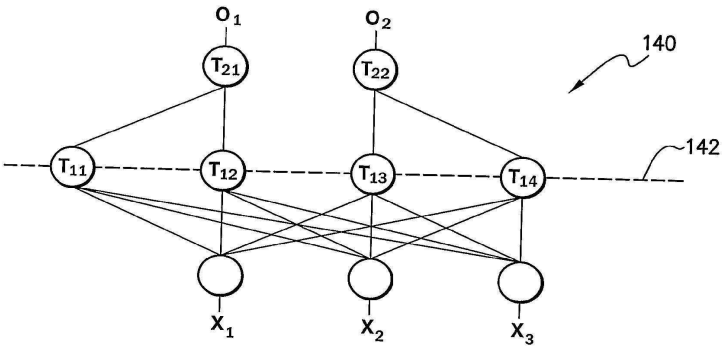
도면14d



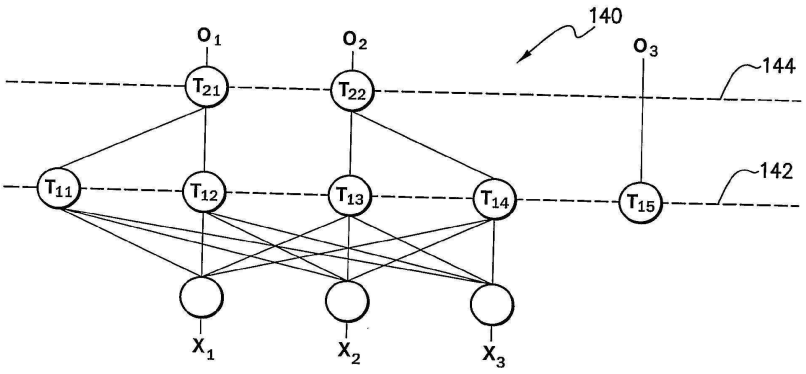
도면14e



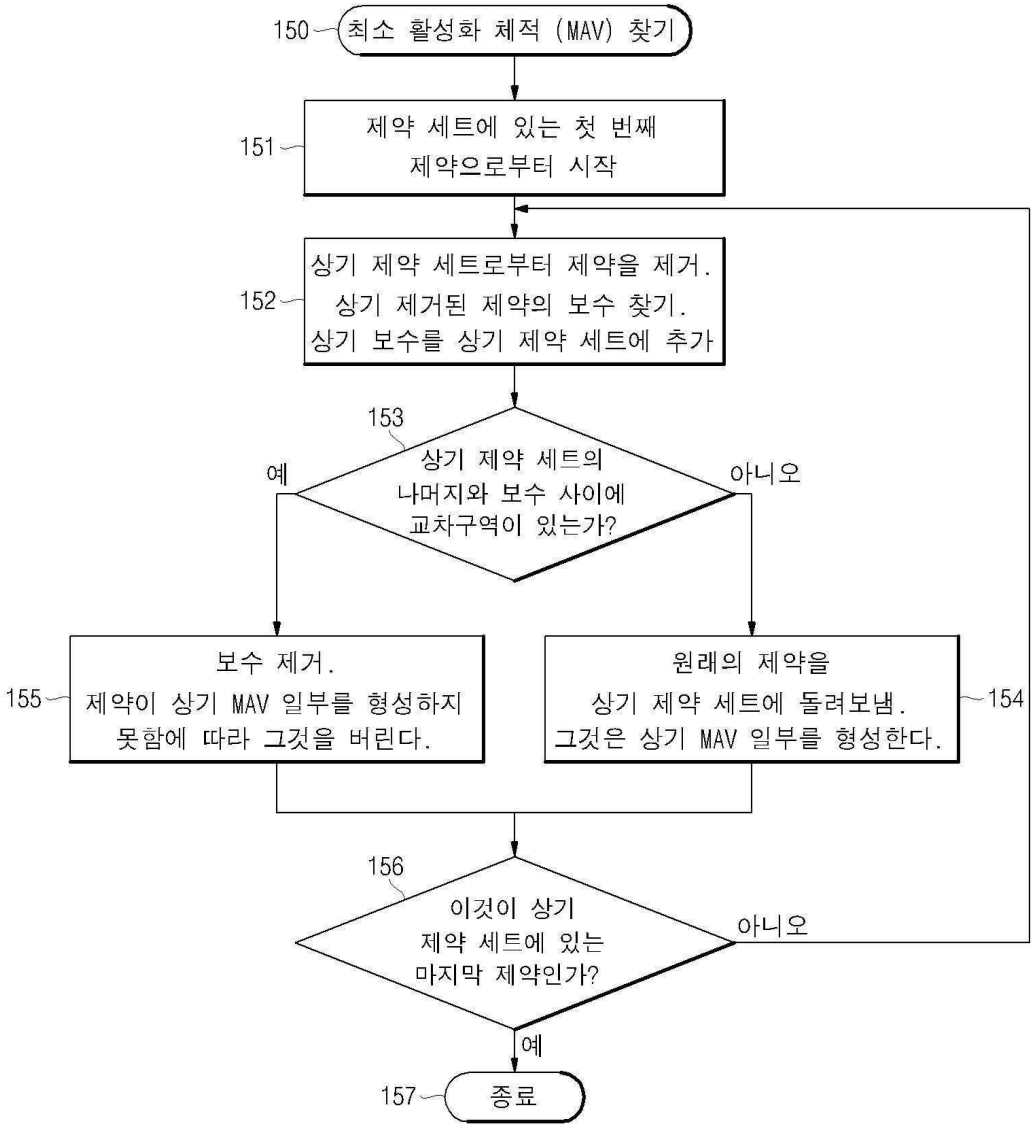
도면14f



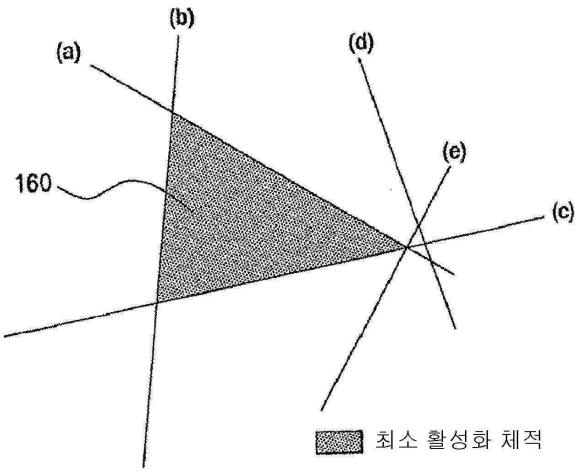
도면14g



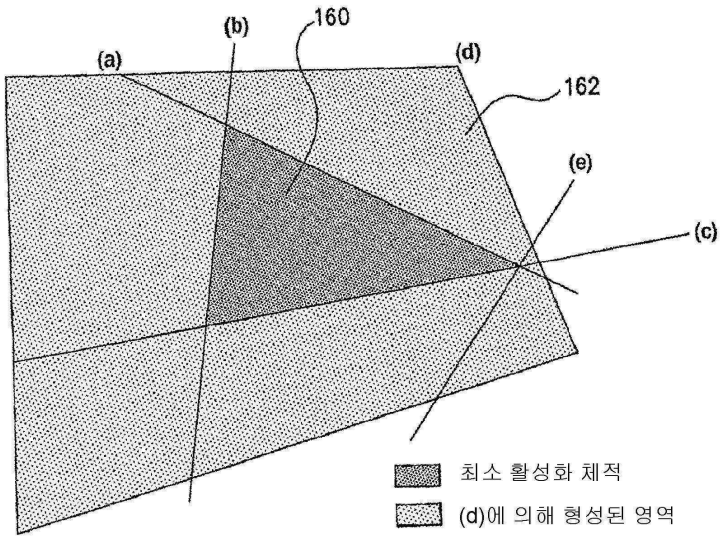
도면15


150
최소 활성화 체적 (MAV) 찾기
151
제약 세트에 있는 첫 번째 제약으로부터 시작
152
상기 제약 세트로부터 제약을 제거. 상기 제거된 제약의 보수 찾기. 상기 보수를 상기 제약 세트에 추가
153
상기 제약 세트의 나머지와 보수 사이에 교차구역이 있는가?
예
아니오
155
보수 제거. 제약이 상기 MAV 일부를 형성하지 못함에 따라 그것을 버린다.
154
원래의 제약을 상기 제약 세트에 돌려보냄. 그것은 상기 MAV 일부를 형성한다.
156
이것이 상기 제약 세트에 있는 마지막 제약인가?
아니오
예
157
종료

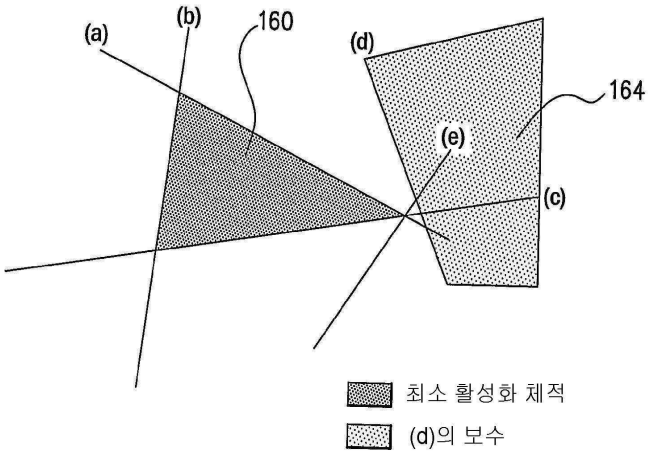
도면16a


(a)
(b)
(c)
(d)
(e)
160
최소 활성화 체적

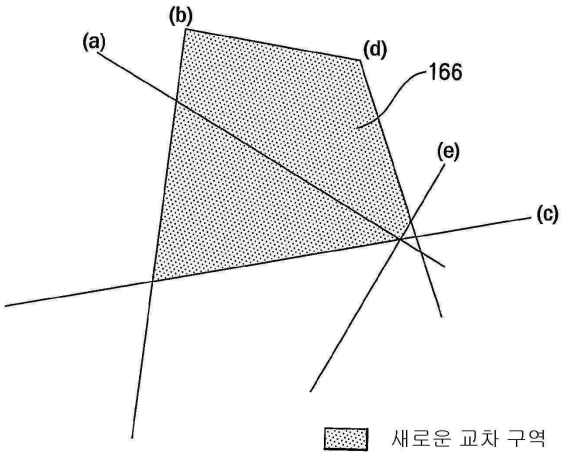
도면16b



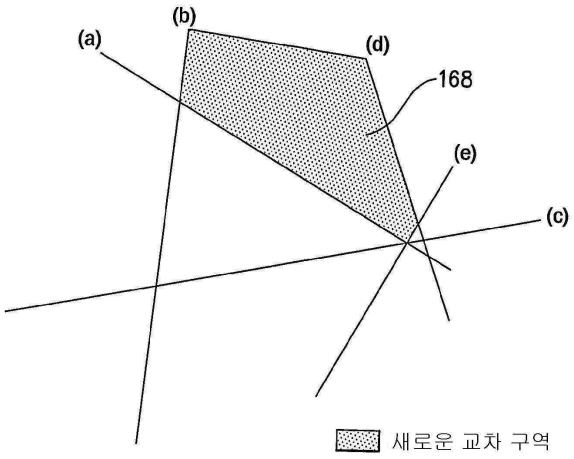
도면16c



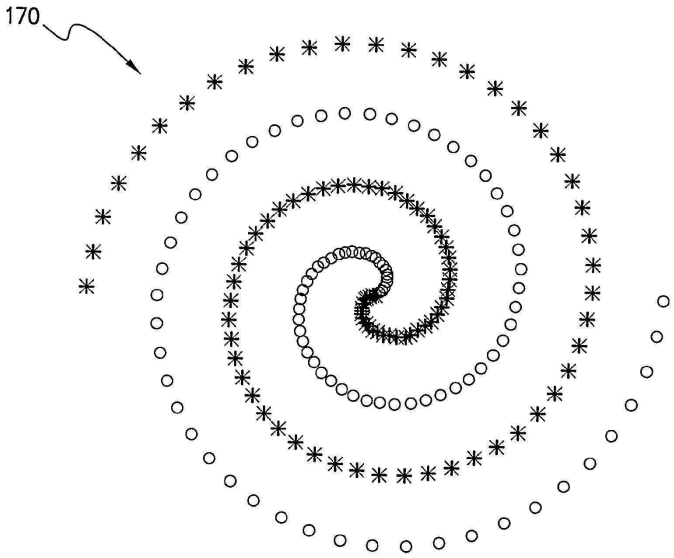
도면16d



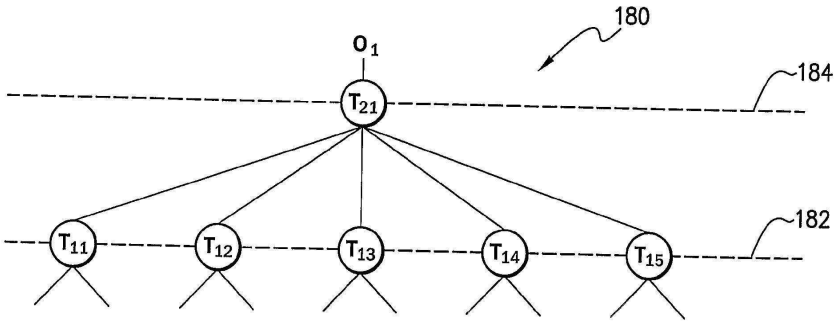
도면16e



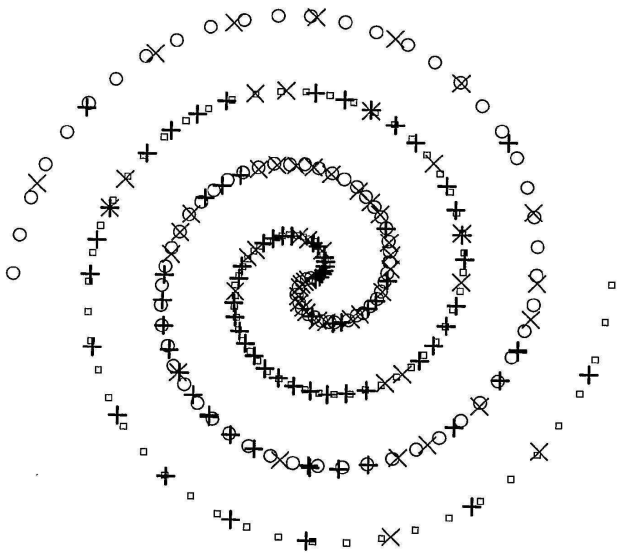
도면17



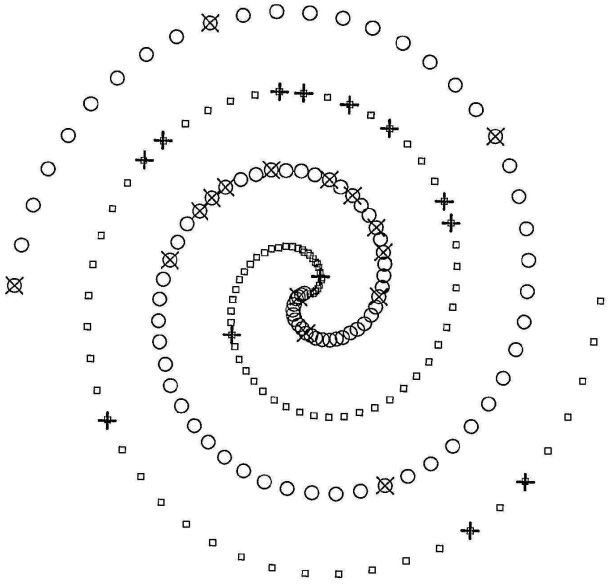
도면18



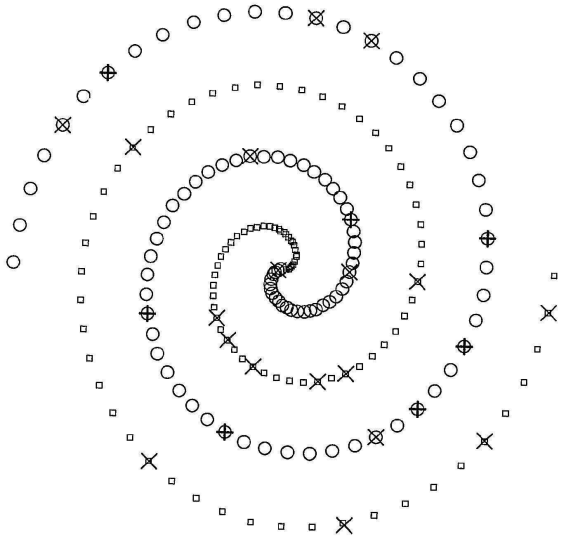
도면19



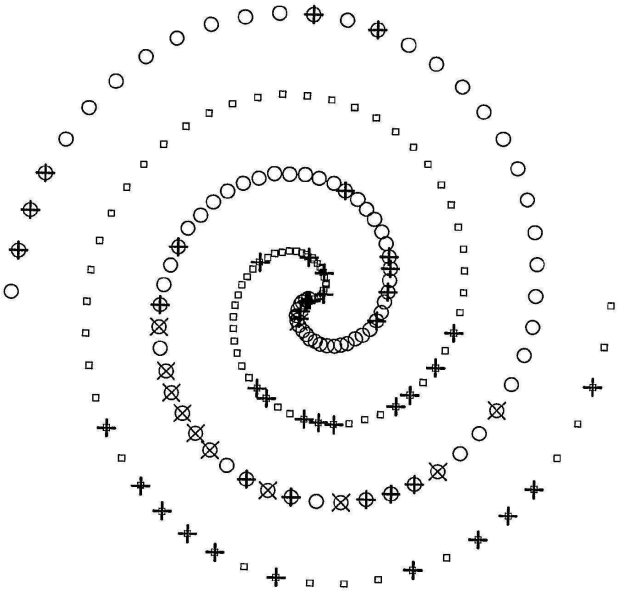
도면20a



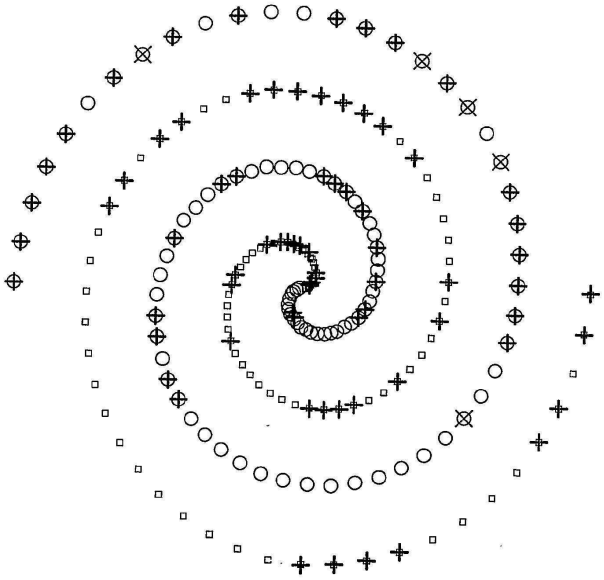
도면20b



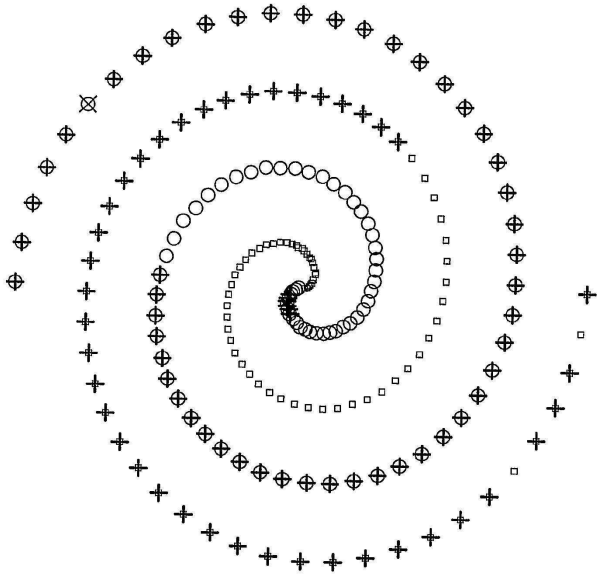
도면20c



도면20d



도면20e



도면21