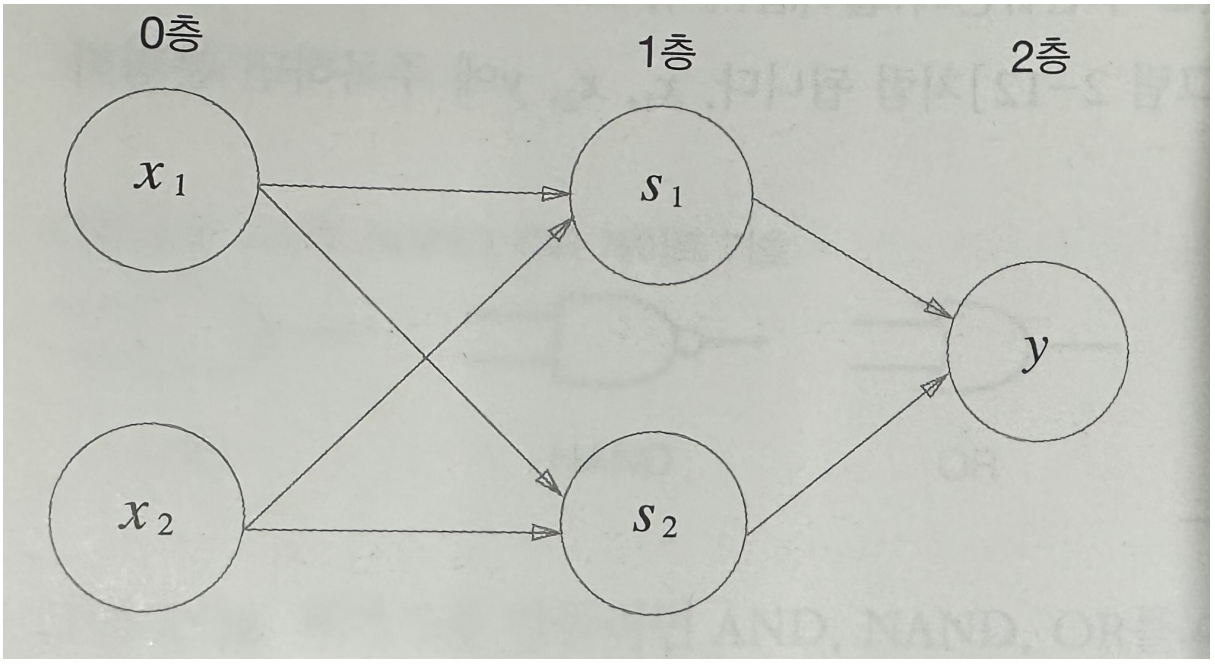
Rumelhart. et. al paper review

작성자 : 이서진 (본 1)

1. *수식 삽입의 편의를 위해 pdf Xchange가 아닌 다른 방식으로 작성했습니다. 양해 부탁드립니다.*
2. *다양한 출처로부터 설명을 넣었습니다. (주로 stackoverflow, '밑바닥부터 시작하는 딥러닝' 1권 등.)*
3. *논문 summary를 보시고자 하다면, '2) Backpropagation 3. 수식으로 이해하기' 부터 봐주세요.*
4. *오타, 오류가 있다면 개인톡 부탁드립니다.*

1) Motivation : 각 output에 대해 weight를 빠르게 (적은 연산으로) 개선할 수 있는 모델 개발

- 가장 기초적인 NN : 여러 개의 input과 weight의 선형결합으로 하나의 output 출력. input과 output 사이에 intermediate layer 존재



$$x_j = \sum_i y_i w_{ji}$$

(input y_i 와 bias w_{ji}의 곱으로 나타난다.)

$$y_j = \frac{1}{1+e^{-x_j}}$$

(NN의 마지막 단계는 활성화 함수이며, 논문에서는 sigmoid 함수를 이용했다. output은 해당 함수의 출력으로 나타난다.)

> *반드시 위와 같은 함수를 사용하지 않아도 되지만, 모델의 성능을 위해 비선형 함수를 출력 과정에 이용한다. 선형 함수만을 사용한다면 XOR 연산과 같은 비선형 문제를 처리하지 못하기 때문이다. 앞 단계에서는 선형 함수를(input과 bias의 선형 결합) 마지막 단계에서 비선형 함수를(sigmoid) 이용한 것은 계산의 편의를 위해서이다.*

- NN, 즉 학습 모델의 목적은 모델의 output(y)과 정답(d, desired state) 사이의 오차를 줄이는 것이다. 각 학습 과정에서 output과 정답 사이의 오차를 반영하여, parameter를 조정한다.

  (*ML에서 parameter란 모델이 데이터로부터 학습하는 값들로, 상기 모델에서는 bias (w_{ji})를 뜻한다)

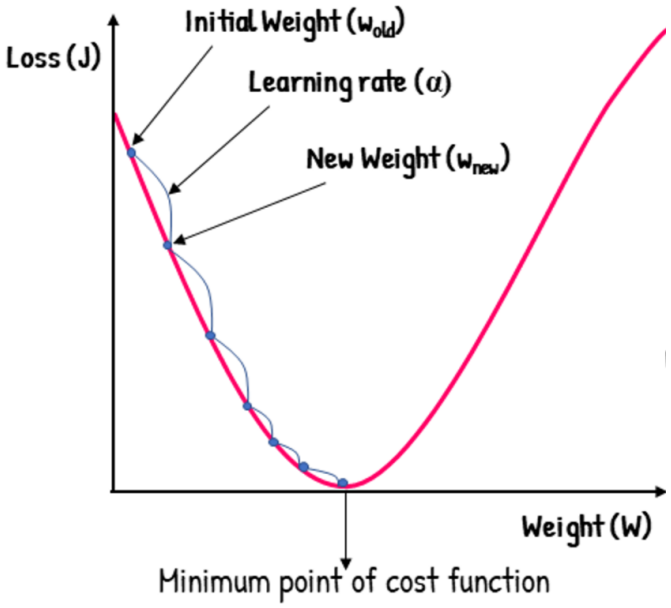
$$E = \frac{1}{2} \sum_c \sum_j (y_{j,c} - d_{j,c})^2$$

- 오차는 오차제곱합(Sum of squares for error, SSE)으로 정의된다. y는 output, d는 정답이다.

  (c는 dataset의 개수를, j는 output의 개수를 의미한다. 예를 들어 100개의 data로부터 5지선다를 해야 한다면, $c = 100, j = 5$일 것이다.)

- E는 y에 대한 함수이고, y는 w에 대한 함수이기 때문에 E를 최소화하는 것은 w에 대한 최적화 문제이다.

따라서 최적의 w값을 찾기 위해서는 j*i개의 변수 w에 대한 gradient descent를 시행해야 한다.



(gradient descent에 대한 더 자세한 설명은 생략한다. E에 대한 w의 편미분값을 조금씩 E에서 빼며, 최솟값에 도달할 때까지 반복 시행하는 방법이다)

- 그러나 각각의 시행에 있어 E에 대한 모든 w의 편미분값을 구하는 것은 매우 많은 계산을 요구한다.

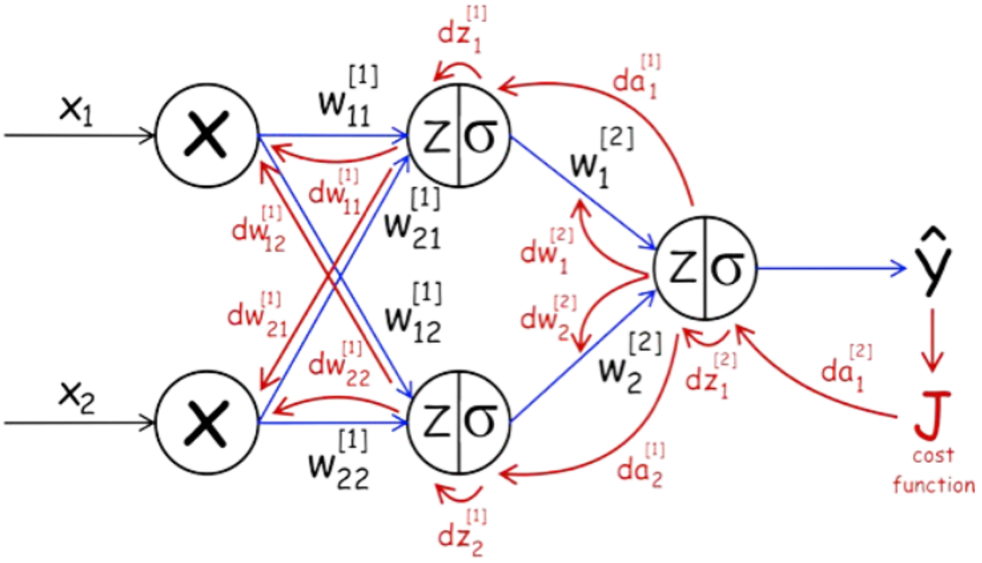
  (실제로, colab으로 연산을 시행한 결과, backpropagation으로 10초 안에 풀리는 계산을 2시간 넘게 풀지 못했다.)

- **본 논문은, 비교적 적은 계산으로도 gradient descent를 시행할 수 있는 backpropagation (역전파)의 개념을 소개하는 논문이다.**

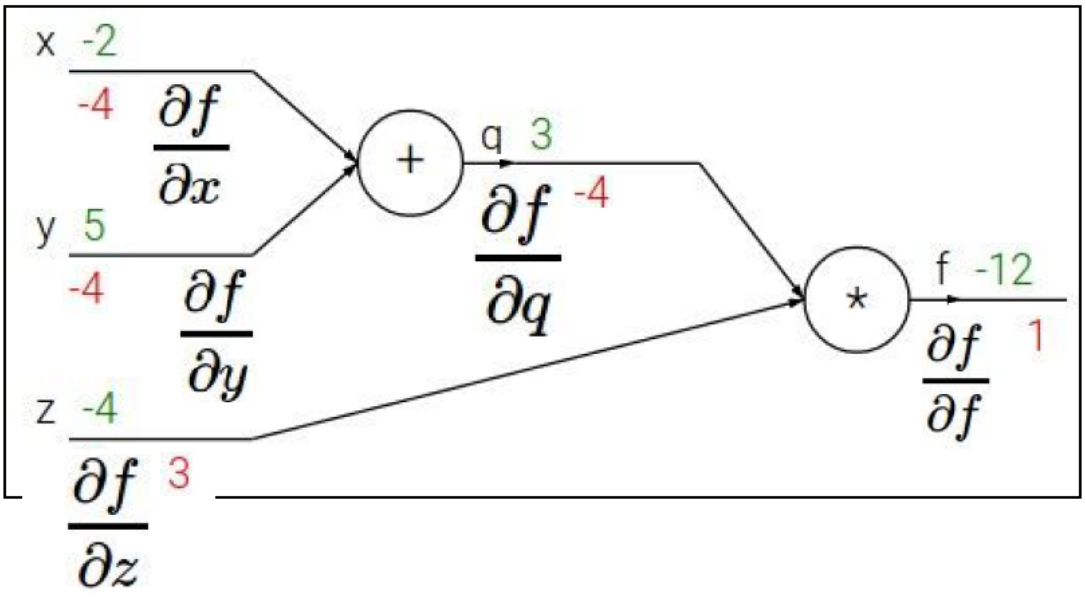
2) Backpropagation (역전파)

1. **역전파는 무엇인가?**

   퍼셉트론의 정보는 input layer에서부터 output layer까지 순차적으로 이동한다. 정보 이동의 반대 방향으로 오차에 대한 정보(즉 오차/전 계층의 편미분값)가 전달되기 때문에 backpropagation이라 부른다.



2. **왜 역전파를 이용하는가?**



Chain rule에 의해 각 계층의 연산을 분해하여 처리할 수 있기 때문이다.

궁극적으로 gradient descent에서 필요로 하는 것은 각각의 w에 대한 E의 변화량이다.

오차 함수 E는 j개의 입력층 (x_j), i개의 출력층 (y_i) 에 해당하는 jXi개의 w에 대한 함수.

오차 함수 E를 한 번에 계산하는 것은, 수많은 w에 대한 E의 변화량을 한 번에 연산해야 하므로 모든 변수를 메모리에 기억해야 함 → 연산의 비효율이 존재하며, 연

산별로 분해하여 각각의 w에 대한 E의 변화량을 따로따로 구해 처리하는 것이 더 효율적임.

3. **수식으로 이해하기 *(여기서부터가 논문 Summary입니다)***

$$E = \frac{1}{2}\sum_{c}\sum_{j}(y_{j,c} - d_{j,c})^2$$

역전파의 첫 단계는 모든 오차(E)를 output(y)에 대해 미분하는 것이다. 하나의 case에 대해 (c 고정) 위 식을 y_j에 대해 미분하면 다음과 같다.

$$\partial E/\partial y_j = y_j - d_j$$

Chain rule을 적용하여 $\partial E/\partial x_j$를 구해보자

$$\partial E/\partial x_j = \partial E/\partial y_j \cdot \mathrm{d}y_j/\mathrm{d}x_j$$

y는 x에 대한 sigmoid 함수이기에, 우리는 dy_j/dx_i를 알며 이를 다음과 같이 정리할 수 있다.

$$\partial E/\partial x_j = \partial E/\partial y_j \cdot y_j(1 - y_j)$$

위 식은 input x가 변화할 때 단위 output E의 변화를 보여준다.

그러나 우리가 원하는 것은 w_{ji}에 대한 단위 output이므로,

$$\partial E/\partial w_{ji} = \partial E/\partial x_j \cdot \partial x_j/\partial w_{ji}$$
$$= \partial E/\partial x_j \cdot y_i$$

w는 jXi 행렬로 나타나므로, 하나의 unit만 고정해서 살펴보자. y_i만 고정하여 본다면,

$\partial E/\partial y_i$에 위와 동일하게 chain rule을 적용할 수 있다.

$$\partial E/\partial x_j \cdot \partial x_j/\partial y_i = \partial \cdot E/\partial x_j \cdot w_{ji}$$

그렇다면 하나의 y에 대한 $\partial E/\partial y_i$로 정보를 전달하는 모든 input (x_j)의 오차는 다음과 같이 나타낼 수 있다.

$$\partial E/\partial y_i = \sum_j \partial E/\partial x_j \cdot w_{ji}$$

따라서 우리는 모든 개별적인 중간 층에 대하여, w에 따른 E의 변화량을 구할 수 있게 되었다. 같은 방법으로 모든 개별적인 input에 대해서도 w에 따른 E의 변화량을 구할 수 있다.

4. **오차 역전파와 gradient descent**

   $\partial E/\partial w$를 알았을 때 gradient descent의 방법을 알아보자.

   - 가장 일반적인 gradient descent 방법은 다음과 같다.

     $\Delta w = -\epsilon \partial E/\partial w$

     앞에서 설명했던 것처럼 E의 변화량으로부터 w를 조금씩 이동시키며 최적점을 찾는 과정이다.

- 논문은 가속도 개념을 응용한 개선된 방법을 또한 제시한다. (accelerated gradient descent)

  $$\Delta w(t) = -\epsilon \partial E / \partial w(t) + \alpha \Delta w(t-1)$$

  여기서 α는 0과 1 사이로 설정된 exponential decay factor이며, input-output set에 대해 1회 학습할 때마다 t가 1씩 증가한다. 가속도 모델에서는, 학습을 1회 진행할수록 gradient descent 정도가 이전 학습량에 비례하여 감소하며, 점점 천천히 최솟값에 도달한다.

3) Figures (참고)

Figure에서는 backpropagation 모델로 해결한 문제와, 구체적인 모델을 설명한다.

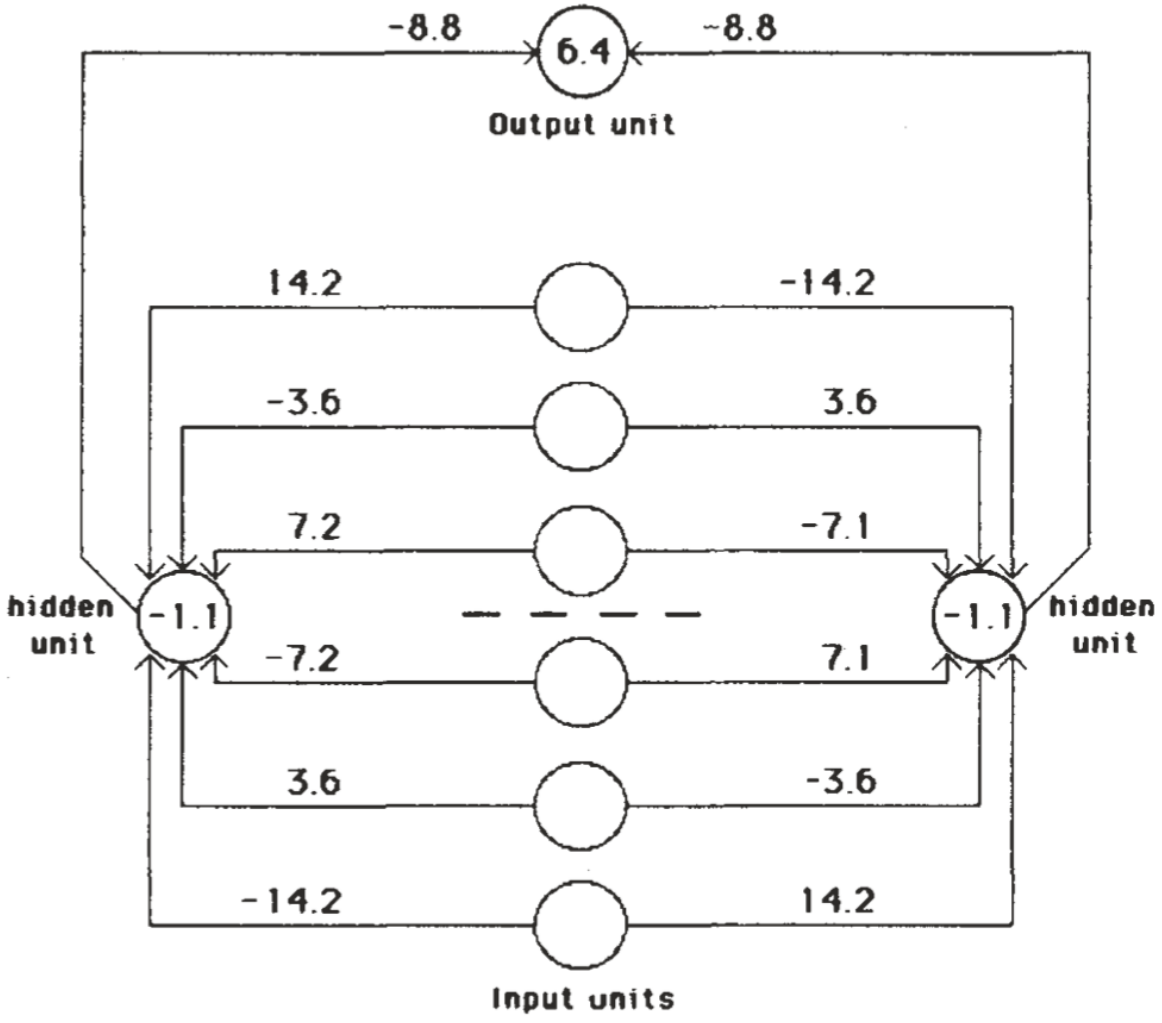


Fig. 1은 input vector의 mirror symmetry를 구분하는 모델이다.

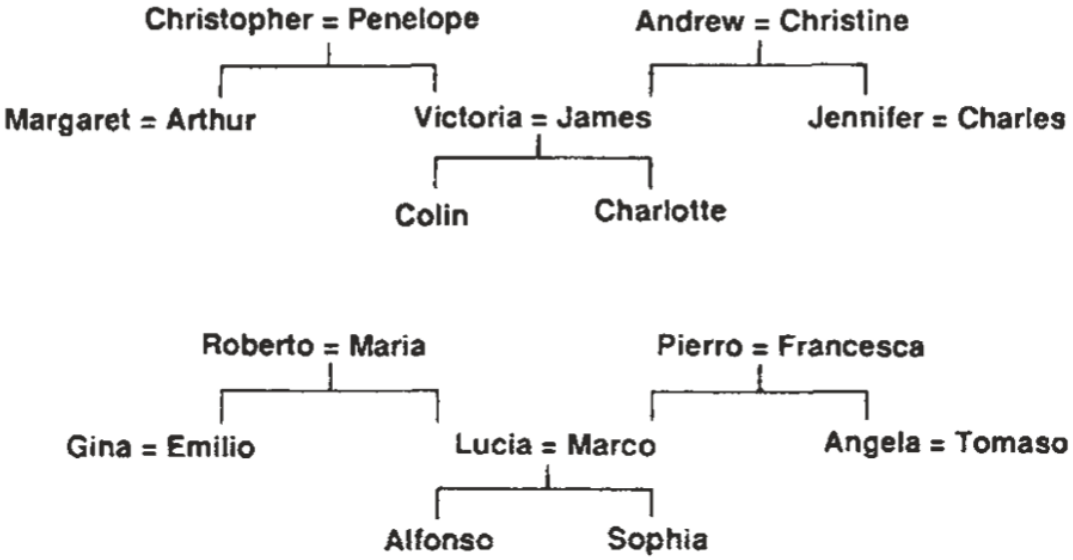


Fig. 2는 (가족 이름), (관계), (관계에 해당하는 가족의 이름)의 3차원 벡터를 input으로 받고,

앞 2개 데이터로 세 번째 가족의 이름을 맞추는 모델이다.

위 가계도는 영국인, 아래 가계도는 이탈리아인이며 성공적으로 학습되었다면 모델은 이름과 관계없이 두 가계도를 같은 형태로 인식할 것이다.

ex) Colin, Mother → Victoria

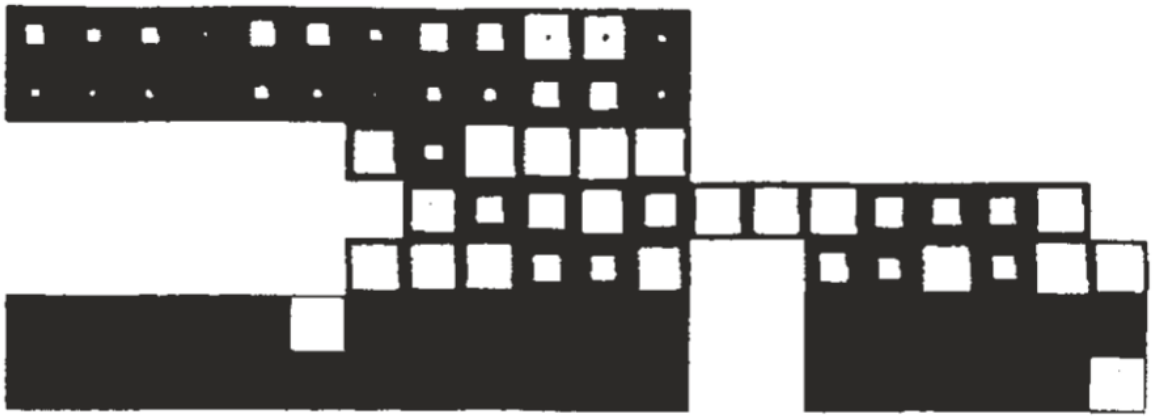


Fig. 3은 Fig. 2 5-layer model의 activity level을 보여준다.

맨 아랫줄은 '가족 이름'에 해당하는 왼쪽 24개의 input과 '관계'에 해당하는 오른쪽 12개의 input이다.

두 번째 층은 input과 완전히 연결되어 있으며, 세 번째 층은 6개의 unit으로 구성되어 있다.

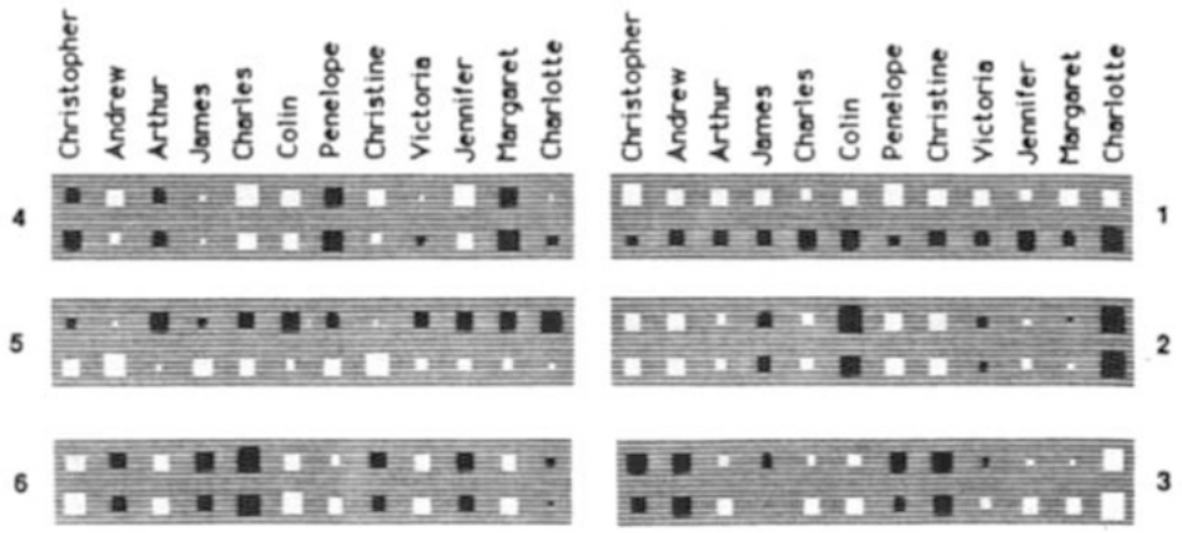


Fig. 4는 Fig. 2의 weight 분포를 나타낸다. 흰 사각형은 “흥분” (활성화), 검은 사각형은 “억제”를 나타낸다.

각 두 줄 중 윗줄은 영국, 아랫줄은 이탈리아인인데 모델은 성공적으로 두 그룹을 구분하면서, 두 가계도를 같은 형태로 인식했다. 연구진은 accelerated gradient descent로 1,500번 학습을 했으며, 첫 20회동안은 e = 0.005, a = 0.5, 나머지 학습은 e = 0.01, a = 0.9로 진행했다.

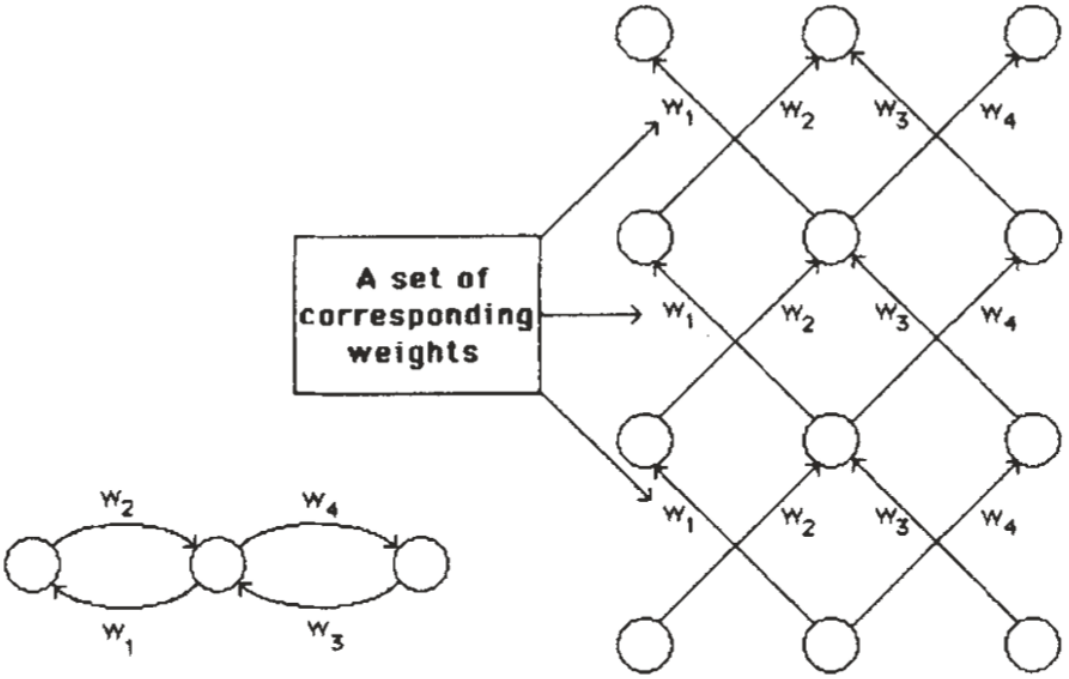


Fig. 5. Synchronous iterative net.