다층퍼셉트론 신경회로망
Multilayer Perceptron Neural Network

모델 학습

비선형 2층 퍼셉트론 모델

- Nonlinear regression 인 경우,

$$y = \Theta_0 + \sum_{j=1}^{H} w_j \{g(\Theta_j + \sum_{i=1}^{p} w_{ij}\, x_i)\}$$

- Nonlinear classification 인 경우는?

Example – Using fat & salt content to predict consumer acceptance of cheese

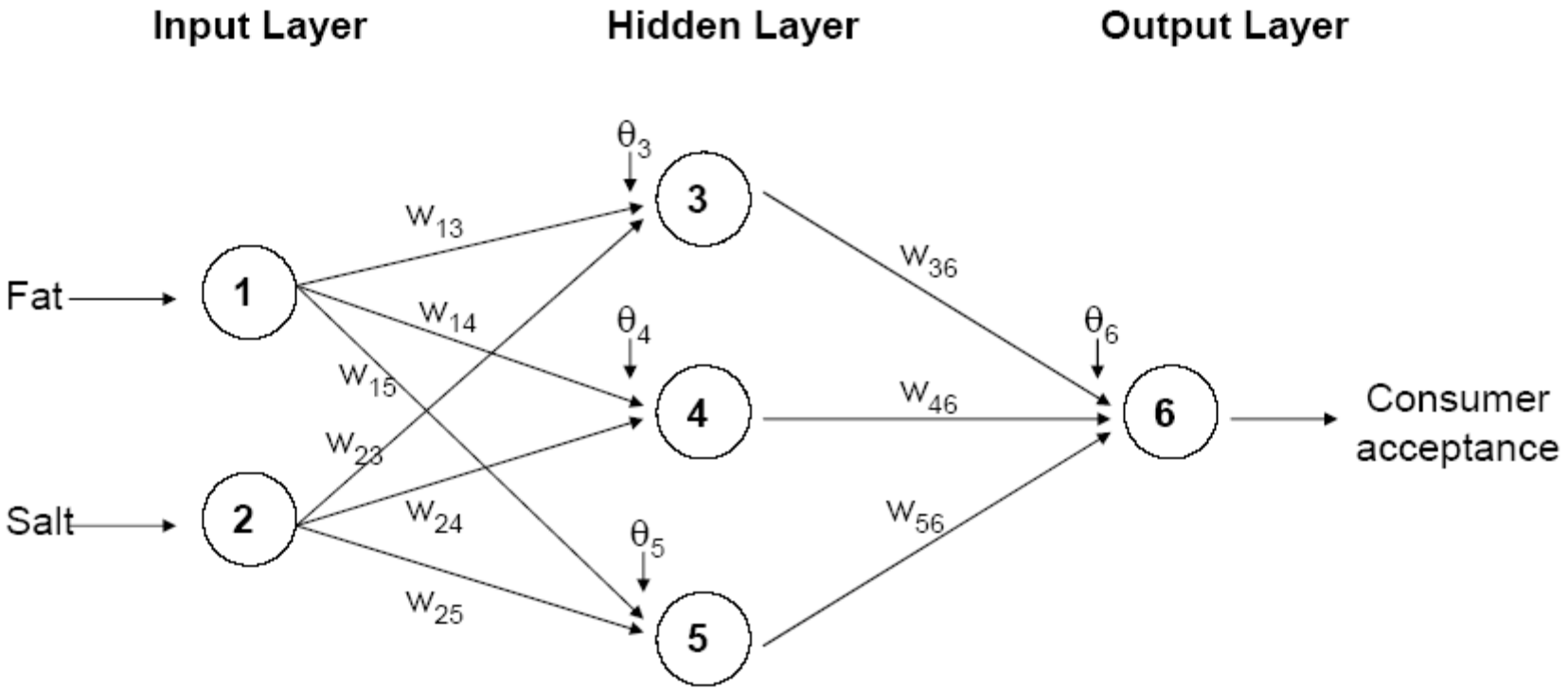


Figure 11.2: Neural network for the tiny example. Circles represent nodes, $w_{i,j}$ on arrows are weights, and θ_j are node bias values.

Example – Training Data

Obs.	*Fat Score*	*Salt Score*	*Acceptance*
1	0.2	0.9	1
2	0.1	0.1	0
3	0.2	0.4	0
4	0.2	0.5	0
5	0.4	0.5	1
6	0.3	0.8	1

학습

-
- Nonlinear regression 인 경우,

$$y = \Theta_0 + \sum_{j=1}^{H} w_j \{g(\Theta_j + \sum_{i=1}^{p} w_{ij}\, x_i)\}$$

- 학습 데이터의 x 를 입력했을 때,
- 학습 데이터의 해당 y 값이 출력으로 나오도록
- w, Θ_j 를 결정

학습

- 에러 함수 Error function 정의
  - “target” output ***t*** 와 “actual” output ***y*** 의 차이의 제곱
  - *E (w) = ½ (**t**-**y**)^2*
    - ***t***: data or target/desired output,
    - ***y***: actual model output

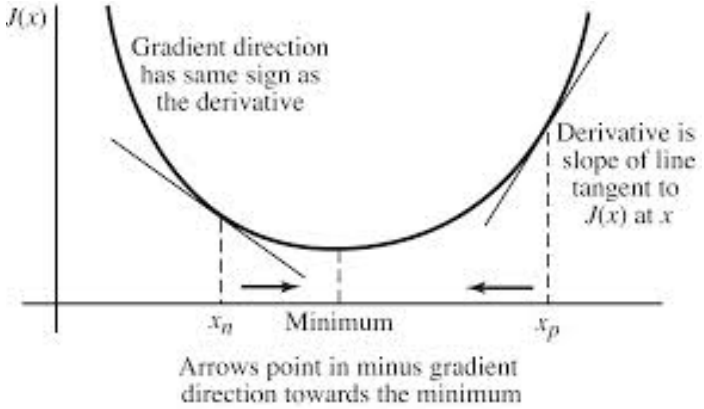
- 에러 함수 ***E***는 w 에 대한 2차 함수?
- 에러 함수를 최소로 하는 w 를 구하자!
- 비선형 최적화

학습 Steepest (Gradient) Descent

- 1차 도함수 Gradient 를 구하고
- w 를 gradient 의 반대 방향으로 조금씩 이동
- w 가 더 이상 변화하지 않을 때까지 (즉, 도함수=0) 반복

$$\Delta w = -\eta \frac{\partial E}{\partial w}$$

학습 Gradient Descent



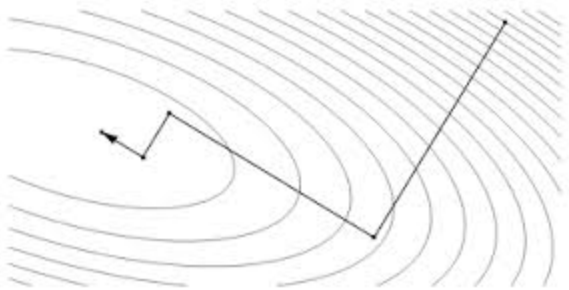
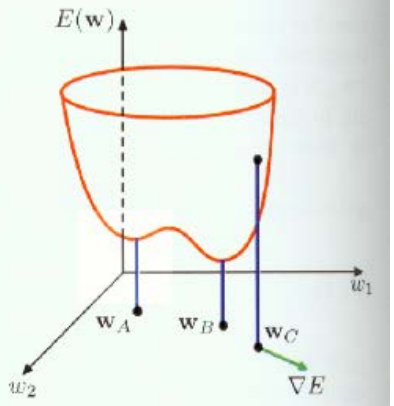


Figure 5.5 Geometrical view of the error function $E(\mathbf{w})$ as a surface sitting over weight space. Point $\mathbf{w}_A$ is a local minimum and $\mathbf{w}_B$ is the global minimum. At any point $\mathbf{w}_C$, the local gradient of the error surface is given by the vector ∇E.



Gradient Descent on MLP

$$E = \tfrac{1}{2}(t-y)^2 \qquad o_j = \varphi(\mathrm{net}_j) = \varphi\left(\sum_{k=1}^{n} w_{kj} o_k\right) \qquad \varphi(z) = \frac{1}{1+e^{-z}}$$

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}} \qquad \frac{\partial \varphi}{\partial z} = \varphi(1-\varphi)$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial \mathrm{net_j}}\frac{\partial \mathrm{net_j}}{\partial w_{ij}}$$

$$\frac{\partial \mathrm{net_j}}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}}\left(\sum_{k=1}^{n} w_{kj} o_k\right) = o_i$$

$$\frac{\partial o_j}{\partial \mathrm{net_j}} = \frac{\partial}{\partial \mathrm{net_j}}\varphi(\mathrm{net_j}) = \varphi(\mathrm{net_j})(1-\varphi(\mathrm{net_j}))$$

$$\frac{\partial E}{\partial o_j} = \frac{\partial E}{\partial y} = \frac{\partial}{\partial y}\frac{1}{2}(t-y)^2 = y - t$$

if *j* is a output node

$$\frac{\partial E(o_j)}{\partial o_j} = \frac{\partial E(\mathrm{net}_u, \mathrm{net}_v, \ldots, \mathrm{net}_w)}{\partial o_j}$$

if *j* is a hidden node, u~w are receiving nodes

$$\frac{\partial E}{\partial o_j} = \sum_{l \in L}\left(\frac{\partial E}{\partial \mathrm{net}_l}\frac{\partial \mathrm{net}_l}{\partial o_j}\right) = \sum_{l \in L}\left(\frac{\partial E}{\partial o_l}\frac{\partial o_l}{\partial \mathrm{net}_l} w_{jl}\right)$$

Gradient Descent on MLP

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial \mathrm{net_j}}\frac{\partial \mathrm{net_j}}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = \delta_j o_i \text{ where}$$

$$\delta_j = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial \mathrm{net_j}} = \begin{cases} (o_j - t_j)\varphi(\mathrm{net}_j)(1-\varphi(\mathrm{net}_j)) & \text{if } j \text{ is an output neuron,} \\ (\sum_{l \in L} \delta_l w_{jl})\varphi(\mathrm{net}_j)(1-\varphi(\mathrm{net}_j)) & \text{if } j \text{ is an inner neuron.} \end{cases}$$

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}}$$

- Meaning of δ
- How to compute the δ of hidden/inner neuron *j* ?
- Repeat of Forward phase and backward phase

신경망 학습 back-propagation

- 목표: 에러제곱합을 최소로 하는 synaptic weight 찾기

1. 학습 데이터 x 를 신경망에 입력
2. 입력층에서 은닉층을 거쳐 출력층으로 시그널을 forward propagate 시킴
3. 출력 노드 출력 값 y 와 타겟 값 t 차이를 출력 노드의 오류값 δ 로 계산
4. 이 δ 를 은닉층 쪽으로 역전파 (back propagated) 하여 모든 은닉 노드들의 "오류"값 δ 으로 분배하고,
5. 이 δ 를 연결강도 w_{ij} 수정하는데 사용
6. 이 과정을 모든 학습 데이터에 반복 적용

Online vs offline

$$E = \frac{1}{2}(t - y)^2$$

- 학습 에러함수는 학습 데이터 전체에 대해 정의해야 함
- 즉, Step 5 and 6 의 순서 바뀌어야 함
- 그런데 그냥 진행하는 것이 더 빨리 수렴
  - online learning (cf. batch)
  - See one training data, change w, see another and change w, etc.
- “Stochastic” Gradient descent (or SGD)

전처리 단계

- 0-1로 변수 조정
- 범주형 변수
- Dummy variable 생성
- 비대칭 변수를 변환 (e.g., log)

- 차원 축소 dim reduction
  - 변수 선택 vs 변수 조합

Synaptic weight

θ(theta)와 w는 전형적으로 -0.05에서 +0.05 범위의 랜덤 값으로 초기화됨

이러한 초기 연결강도는 학습의 처음 단계에 사용됨

Iteration 을 멈추는 조건들

- 연결강도가 한 반복과 다음 반복 사이에 거의 변화하지 않을 때
- 오분류율이 요구된 목표치에 도달했을 때
- 실행횟수의 한계에 도달했을 때

다른 최적화 방법

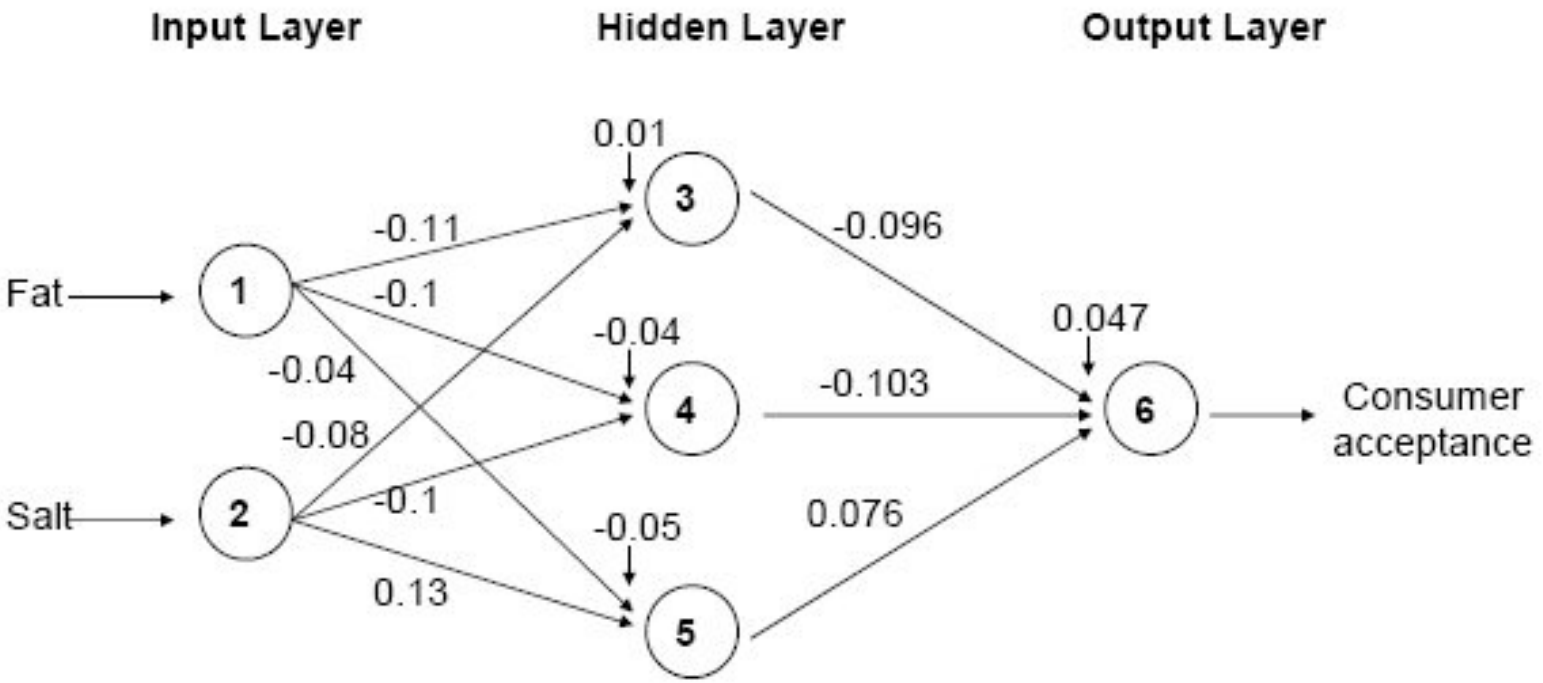
- Steepest Descent (=Back-propagation)
  - (-)
    - 느린 수렴 속도
    - 비선형 최적화 방법 가운데 가장 단순
  - (+)
    - 국지 계산 Local computation
    - 생물학적 가능성 biological plausibility 및 chip 설계

$$\delta_j = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial \mathrm{net_j}} = \begin{cases} (o_j - t_j)\varphi(\mathrm{net}_j)(1-\varphi(\mathrm{net}_j)) & \text{if } j \text{ is an output neuron,} \\ (\sum_{l\in L}\delta_l w_{jl})\varphi(\mathrm{net}_j)(1-\varphi(\mathrm{net}_j)) & \text{if } j \text{ is an inner neuron.} \end{cases}$$

다른 최적화 방법

- 수 많은 다른 방법들 제안 및 시도
  - Conjugate Gradient 등
  - Newton's Method
    - 수렴 iteration 수
    - 2차 도함수 Hessian Matrix 계산 시간 김
- Levenberg-Marquadt
  - Newton's Method 의 2차 도함수를 구하지 않고 추정
  - 속도 up
  - SGD 에 비해 10~100배 빠름
  - 그러나 생물학적 의미는 전혀 없음

지방/염분 예: 마지막 연결강도



모델 선택 model selection

과적합 Overfitting

- With sufficient hidden nodes and training iterations, neural net can easily overfit the data

- 학습 데이터와 검증/테스트 데이터가 공유하는 내재 함수 성질만 학습하는 것이 아니라,
- 학습 데이터만이 가지고 있는 특이성(noise)도 학습

Avoiding Overfitting 원인 제거

1. Too many hidden nodes (모델이 너무 복잡)

   Limit complexity of network

   다양한 hidden node 수를 검증 validation

"최적"의 히든 노드 수

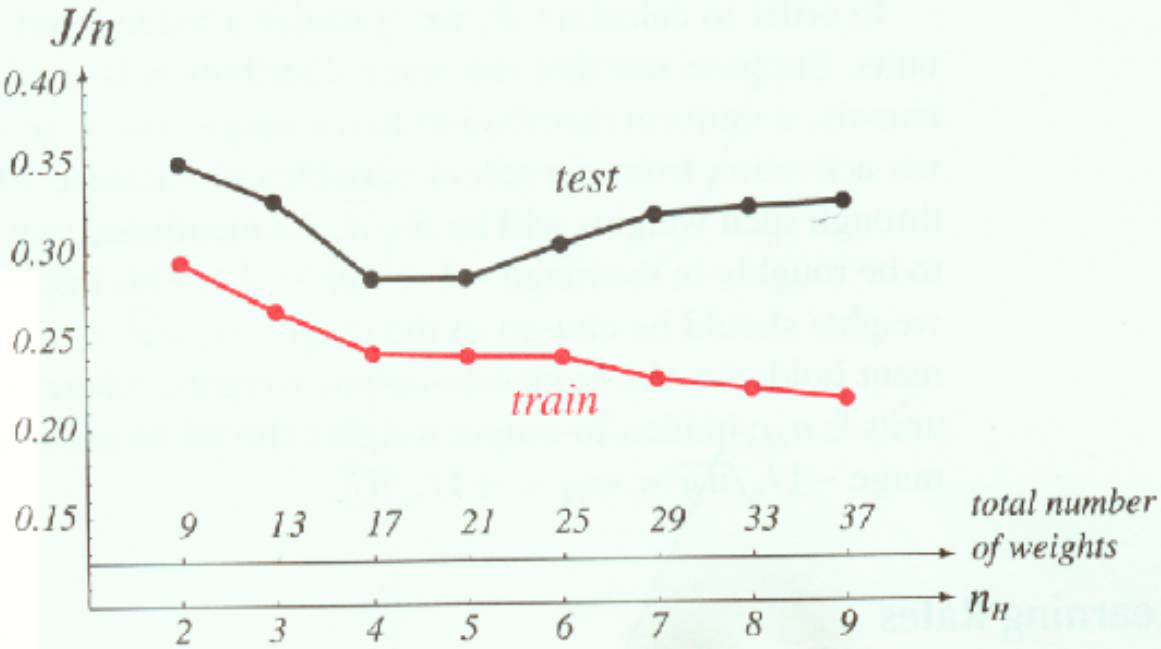


FIGURE 6.15. The error per pattern for networks fully trained but differing in the numbers of hidden units, n_H. Each $2 - n_H - 1$ network with bias was trained with 90 two-dimensional patterns from each of two categories, sampled from a mixture of three Gaussians, and thus $n = 180$. The minimum of the test error occurs for networks in the range $4 \leq n_H \leq 5$, i.e., the range of weights 17 to 21. This illustrates the rule of thumb that choosing networks with roughly $n/10$ weights often gives low test error.

Avoiding Overfitting

1. Too long training (모델의 복잡도를 100% 사용)
   학습 도중에 계속적으로 validation error 측정
   도중에 학습 중지 early stopping

Early stopping

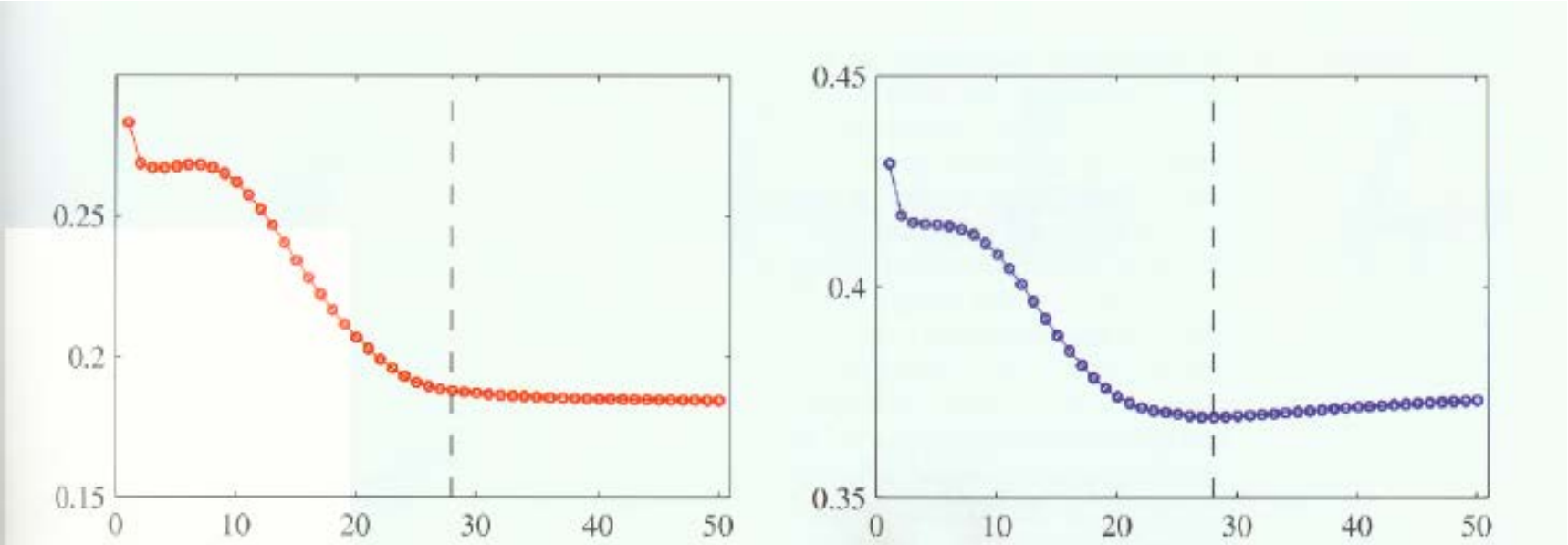


Figure 5.12 An illustration of the behaviour of training set error (left) and validation set error (right) during a typical training session, as a function of the iteration step, for the sinusoidal data set. The goal of achieving the best generalization performance suggests that training should be stopped at the point shown by the vertical dashed lines, corresponding to the minimum of the validation set error.

망 구조 결정

중간층의 개수

- 가장 일반적인 경우 – 하나의 은닉층 (cf Deep Belief Network)

중간층에서 노드의 개수

- 노드가 많을수록 복잡성을 잡아내기 쉽지만, 과적합의 가능성도 높아진다.

출력노드의 개수

- 분류에서, 클래스마다 하나의 노드(또한 이항 응답에서 하나를 사용할 수 있다)
- 수치형 예측에서 하나를 사용

망 구조 (계속)

"학습률" (*l*)

- 낮은 값은 각 반복에서의 오류로부터 새로운 정보의 "반영도를 줄인다".
- 이는 학습을 늦추지만, 국부구조에서 과적합 경향을 축소시킨다.

"관성"

- 높은 값은 이전 반복에서와 같은 방향으로 연결강도를 계속 변화시킨다.
- 이 또한, 국부구조에서 과적합을 피하도록 돕지만, 또한 학습을 저하시킨다.

자동화

- 몇몇 소프트웨어는 입력 파라미터의 최적선택을 자동화한다.
- XLMiner는 중간층의 개수와 노드의 개수를 자동화한다.

결론

장점

- 예측 능력이 좋음
- 복잡한 관계를 잡아낼 수 있음

단점

- “블랙박스” 예측변수와 결과 사이의 관계에 대한 직관을 제공하지 못한다.
- 변수선택 메커니즘이 없다.
- 변수가 많다면 계산량이 늘어 남 (특히 dummy var 은 시냅스의 수를 극적으로 증가시킨다).

요약

- 신경망은 분류와 예측에 사용될 수 있다.
- 타겟 변수와 일련의 예측변수들 사이의 매우 유연하고 복잡한 관계를 잡아낼 수 있다.
- 가장 강력한 예측 모델
- 주요 위험: 과적합
- 대용량 데이터가 필요하다.
- 예측성능이 좋지만, 본질상 “블랙박스”

요약

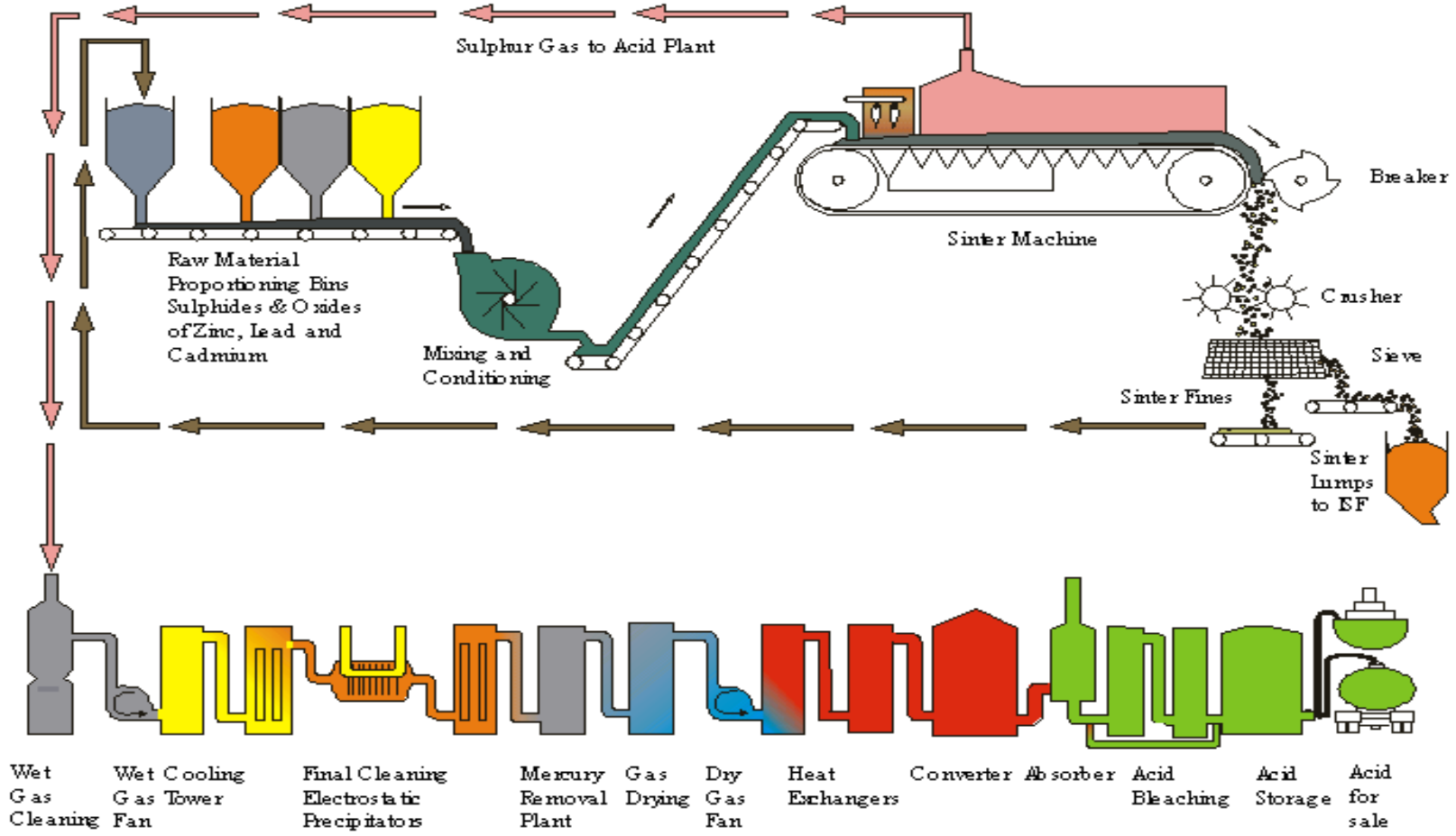
- 수 십 종류의 신경망 모델 존재함
- 이 가운데 가장 많이 사용되는 모델인 “다층 퍼셉트론”을 소개함.
- 다른 교사학습 신경망 모델들은
  - Single layer perceptron
  - Radial basis function networks
  - Probabilistic neural network
  - Recurrent network
- 비교사 학습 신경망 모델들은
  - Self Organizing Map network
  - Hopfield network , Boltzmann Machine

사례

소결 공정

- 분말 형태의 원광석은 고로에 넣어서 생산할 수 없음
- 이를 덩어리 형태로 구워내는 공정

소결 공정의 팰릿 속도 제어 자동화



소결 공정의 팰릿 속도 제어 자동화

- 팰릿의 속도 제어가 가장 중요한 태스크
  - 너무 느리면: 품질 저하, 그리고 먼지, CO 및 CO2 증가
  - 너무 빠르면: 생산량 저하
  - 최적의 속도로 운영하는 것이 생산량, 품질, 비용, 환경 모든 면에 중요함

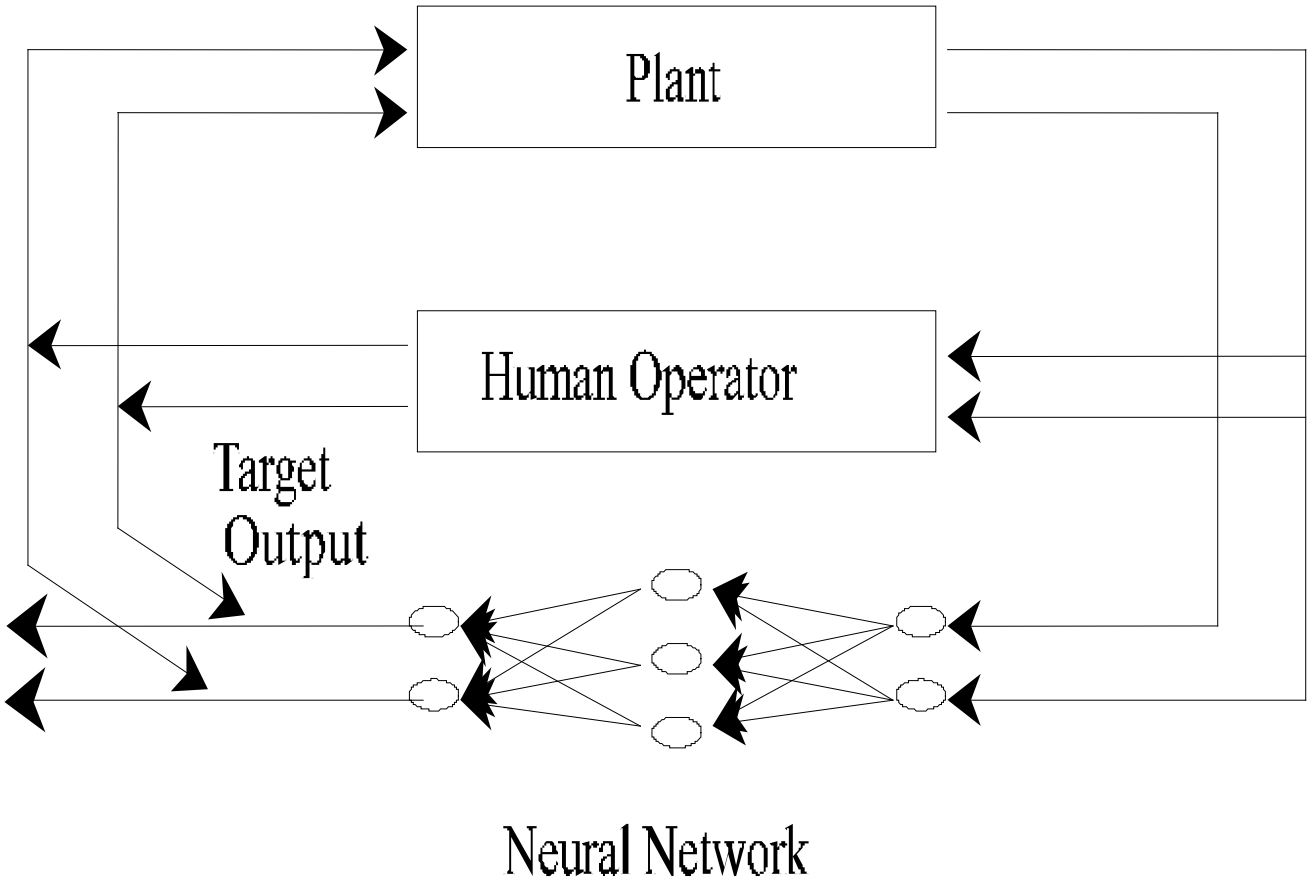
소결 공정의 팰릿 속도 제어 자동화

- 경력 20~30년의 조업자가 매뉴얼로 제어하고 있음
  - 공정 유지 수준의 일관성 확보 어려움
  - 조업자 은퇴 후, 후계자가 없음

- 자동화 모델 필요하여 조업 데이터로부터 조업자와 같은 제어를 할 수 있는 제어 모델 구축 (copy control, 다음 페이지 그림 참조)

- 속도 = f (원자재 관련 변수 120 개 => 5개) 로 모델 구축 (다음 페이지 그림 참조)

소결 공정 Copy Controller



소결공정: 중요 변수 선택

Signif-icance level (%)	Factor	Nermal Range	Update freque-ncy	NN con-trol-ler
19.9	Hot zone properties (height)	180±5cm	1 min	
12.9	Windbox temperature profile	400±20°C	1 min	O
12.4	Bed height	550-10mm	constant	
8.7	Quick lime ratio	1.5±0.2%	10 days	O
8.2	Main Blower air pressure	1525±25 mmH_2O	1 min	O
6.3	Main Blower power	2.9MW± 0.1MW	1 min	O
6.1	Coke size	1.40mm± 0.2mm	10 days	
5.9	SI	91±0.5%	4 hours	
5.2	Return fine ratio	13+2 %	10 days	
3.7	Screening ratio	17.5±2.5%	10 days	
3.5	Blending material moisture	6.0±0.3 %	1 min	O
2.3	Coke ratio	3.7±0.1%	10 days	
2.3	Feeding density	1.955±0.05t on/m^3	2 ~ 3 days	
1.2	RDI	33.2+2.0%	8 hours	

냉간 압연

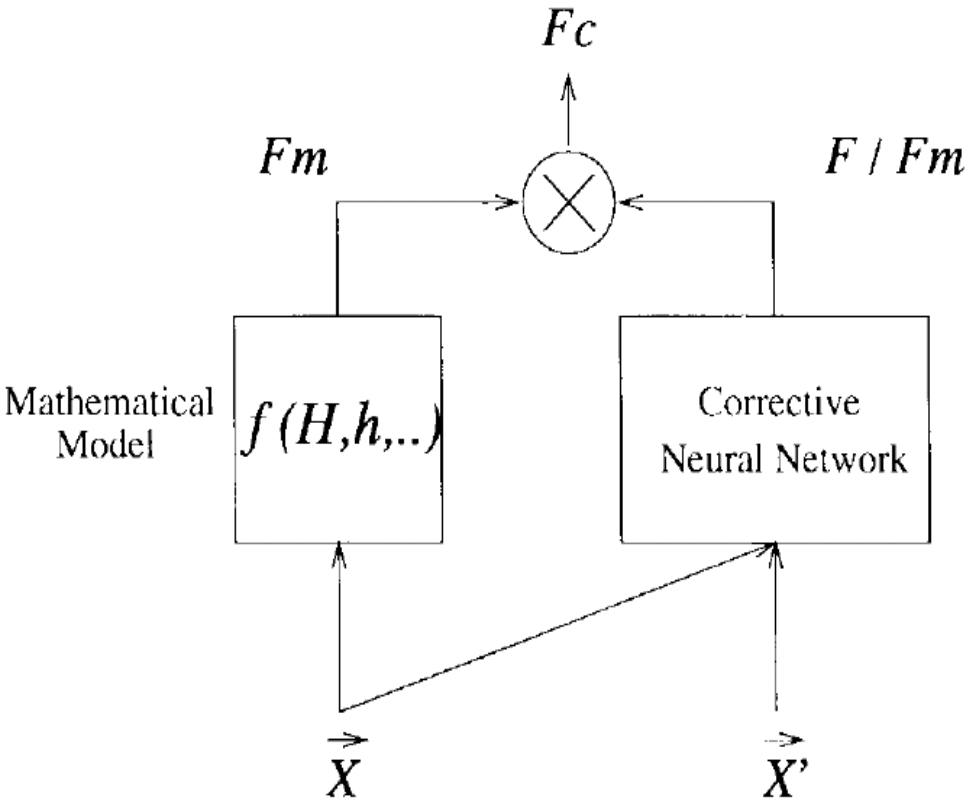
- 상온의 금속재료를, 회전하는 2개의 롤 사이로 통과시켜서 여러 가지 형태의 재료, 즉 판(板)·봉(棒)·관(管)·형재(形材) 등으로 가공

- 주어진 강판을 원하는 두께로 만들기 위해 적용해야 할 Roll의 최적 압력 계산 필요하나 현재 사용하는 물리적 모델은 최적 압력을 계산해내지 못함

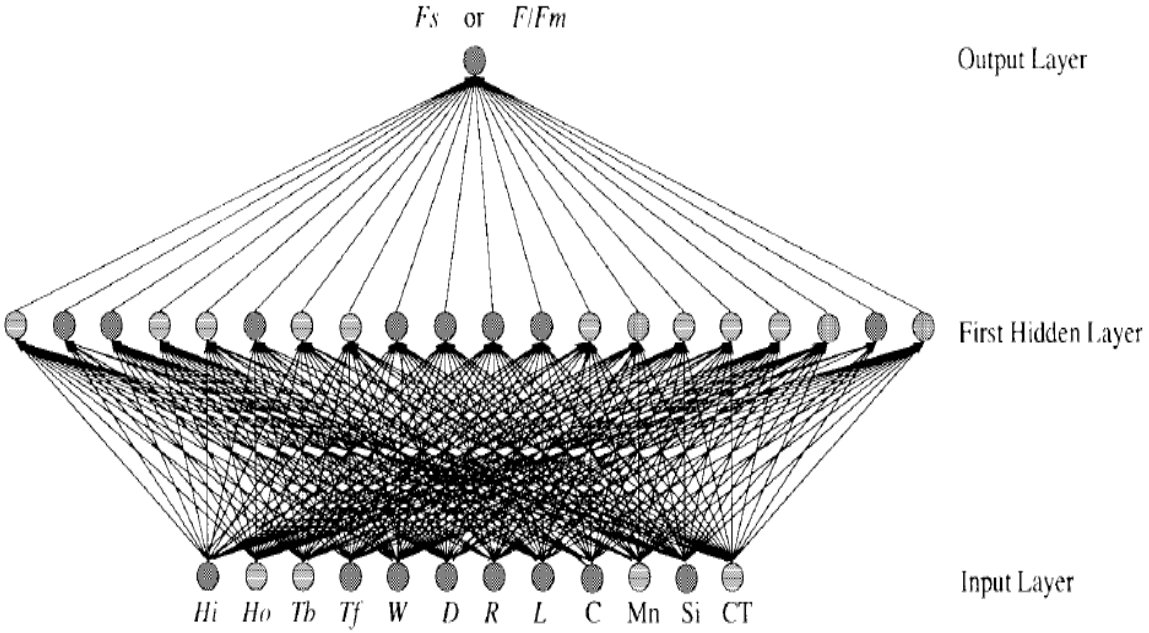
냉간 압연

- 물리적 모델의 압력 보정 값을 타겟으로 하는 신경망 모델을 조업 데이터로부터 구축.
- 즉, 압력_보정치 = NN (강판 두께, 전후 텐션, 넓이, 롤 지름, 성분-탄소,망간,실리콘)
- 이를 이용하여
  - 최적 Roll 압력치 (=수식기반 압력 + 압력_보정치) 계산하여
  - 최적 조업 수행 가능

보정 모델


Fc
Fm
F / Fm
Mathematical
Model
f (H,h,..)
Corrective
Neural Network
X
X'

신경회로망


Fs or F/Fm
Output Layer
First Hidden Layer
Input Layer
Hi Ho Tb Tf W D R L C Mn Si CT

예측 결과

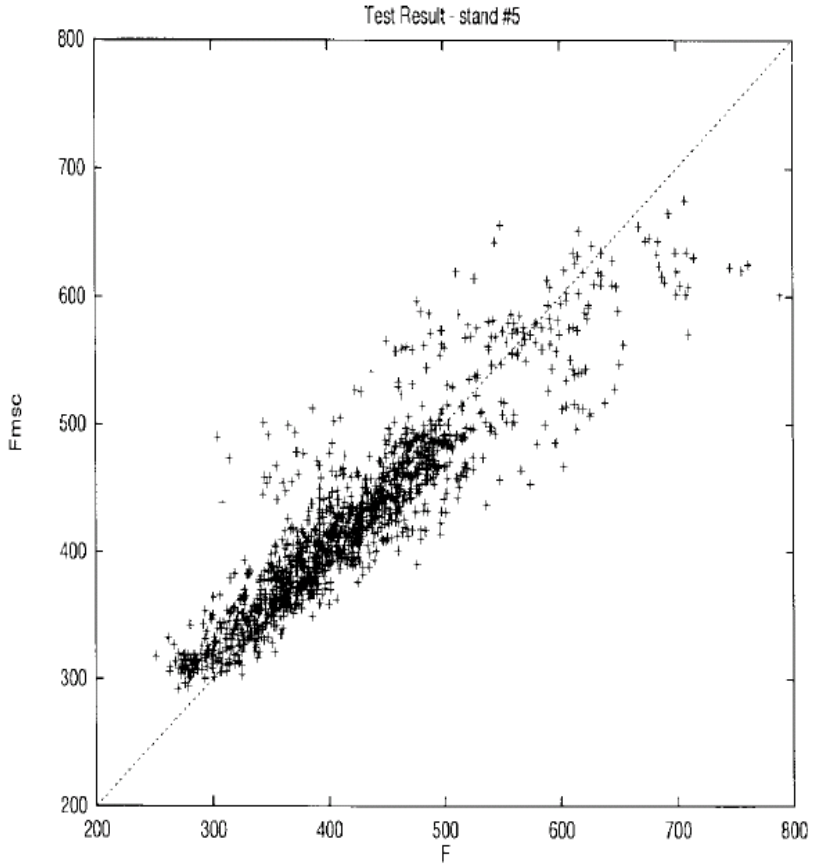
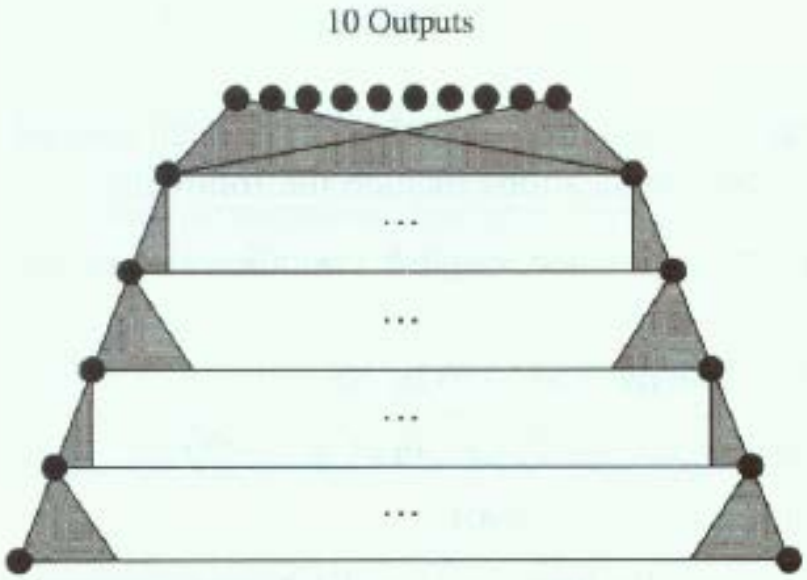


Fig. 11. Predictions of the M-S-C model on *FieldTestSet* at stand no. 5.

숫자 자동 인식



Layer	Neurons	Synapses
5	10	3,000
4	300	1,200
3	1,200	50,000
2	784	3,136
1	3,136	78,400

FIGURE 6.30 General structure of the optical character recognition (OCR) network. (From E. Säckinger et al., 1992a, with permission of IEEE.)

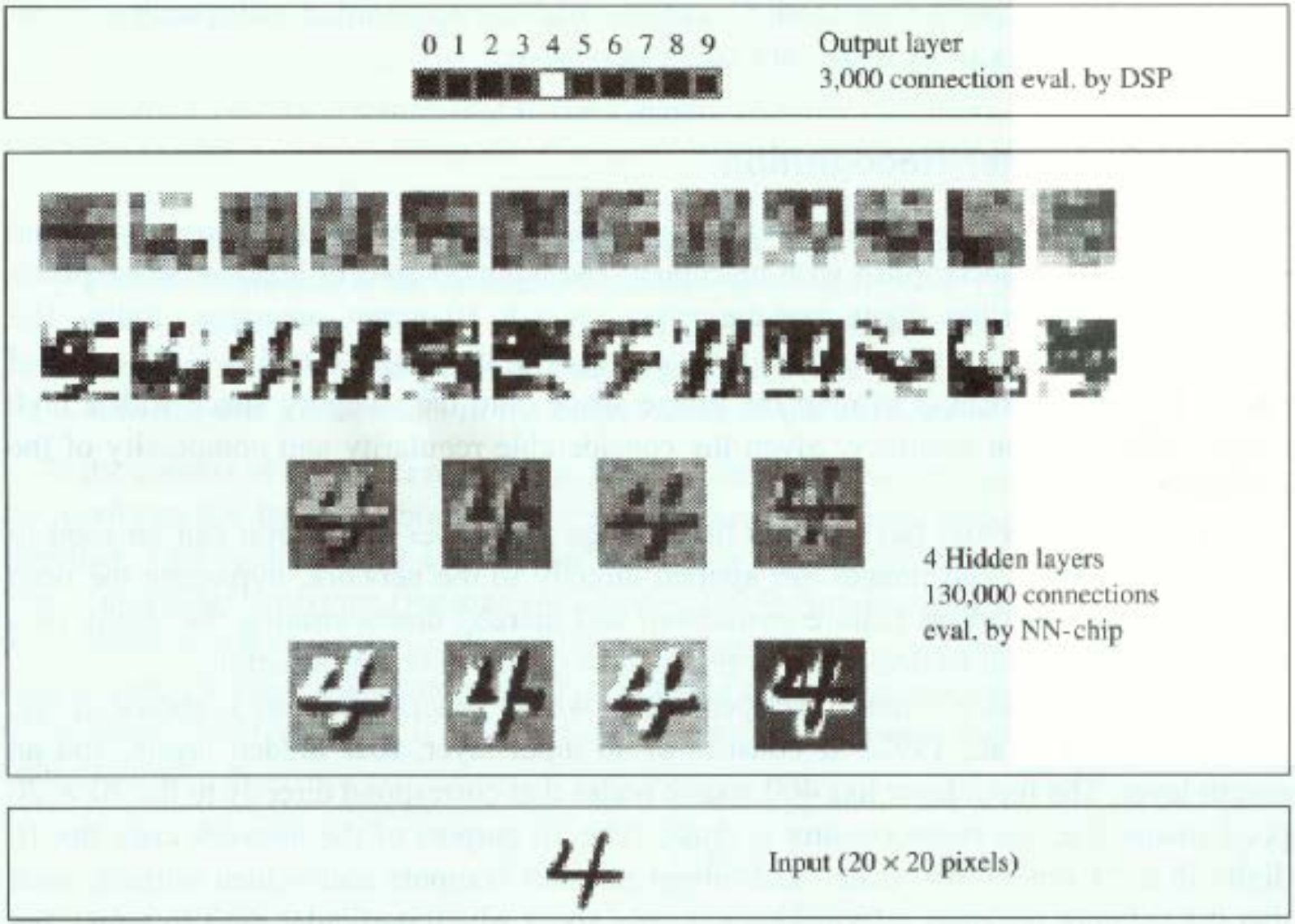


FIGURE 6.31 Example for the states of the OCR network with a number four as input. (From E. Säckinger et al., 1992a, with permission of IEEE.)