Neural Networks (신경망)

5.4 – 5.6 (후반부 발표)

Contents

4 글로벌 미니멈과 로컬 미니멈

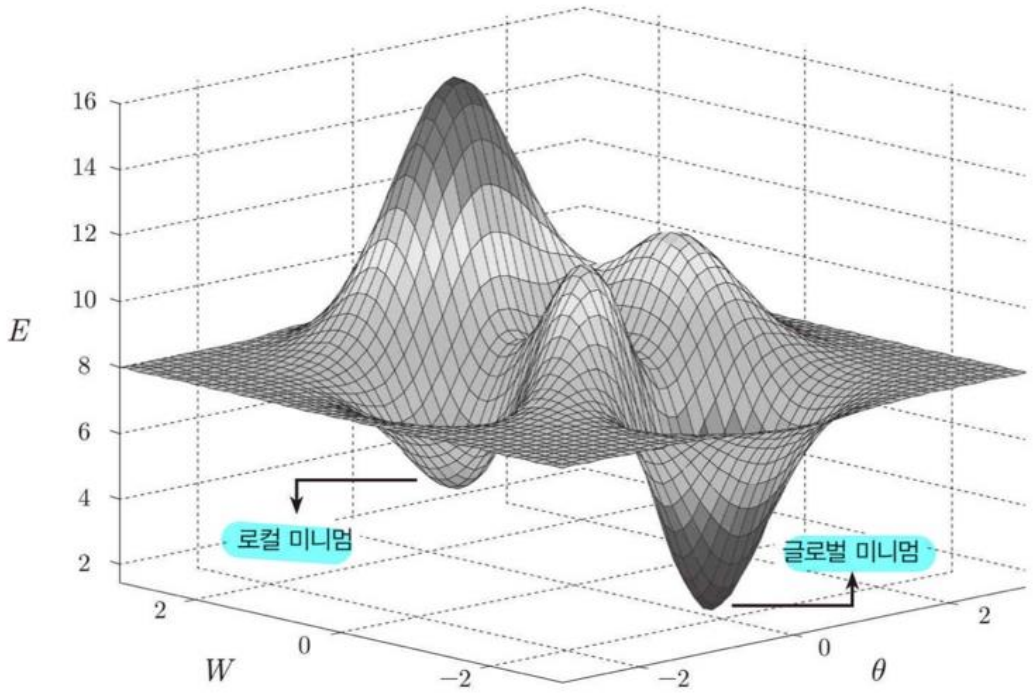
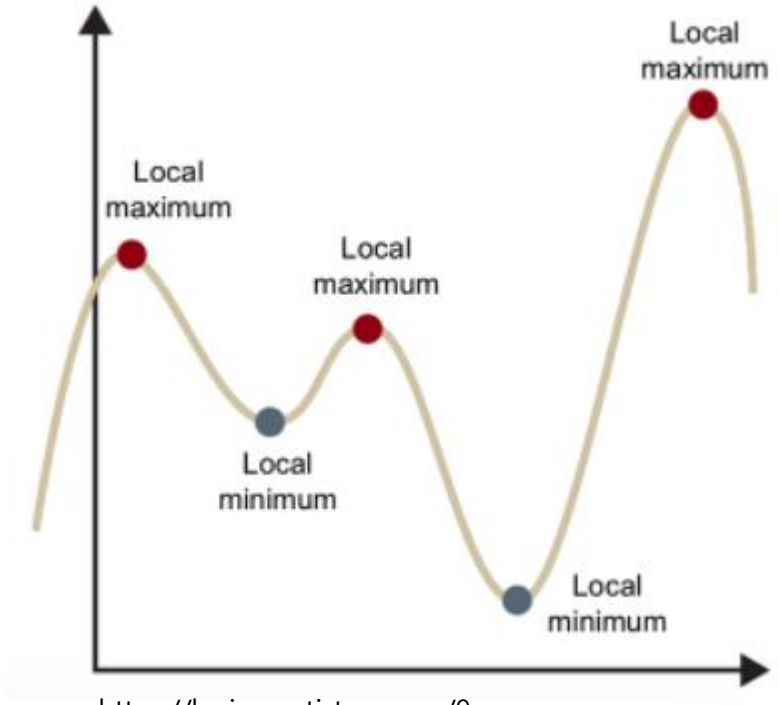
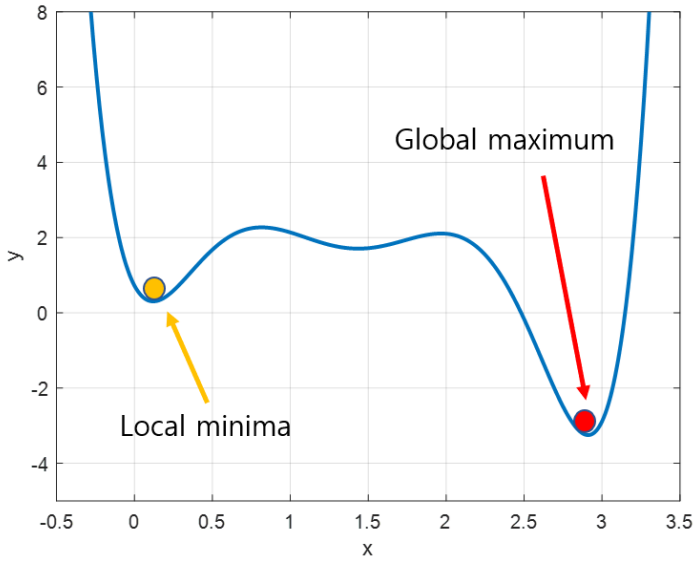


그림 5.10 ＼ 글로벌 미니멈과 로컬 미니멈



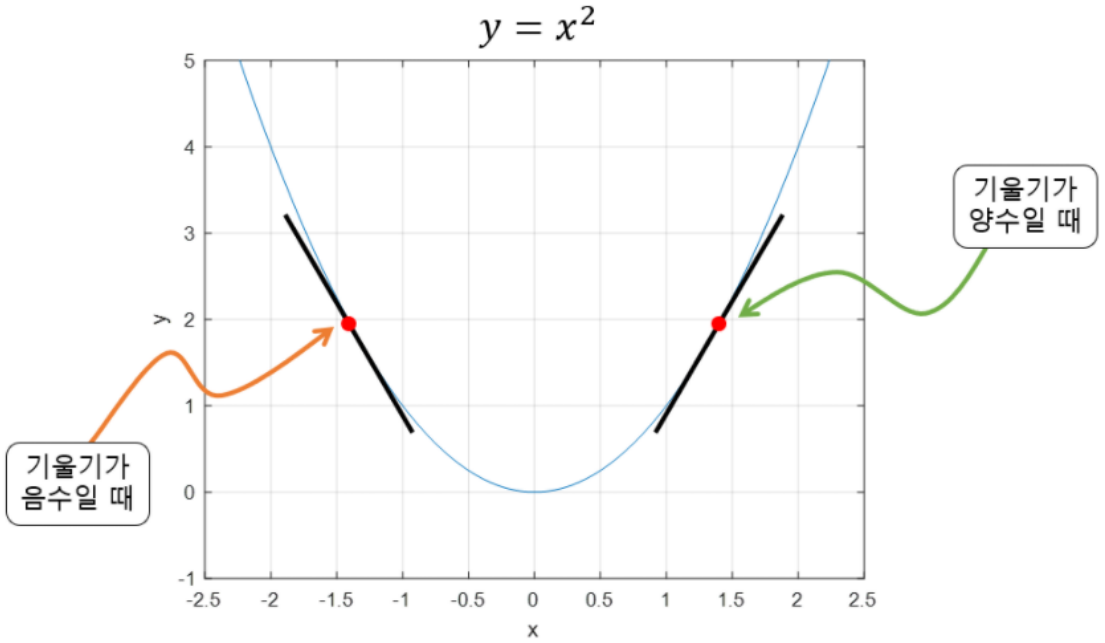
https://hwiyong.tistory.com/9

4 글로벌 미니멈과 로컬 미니멈



경사 하강법 (gradient descent)

- 가장 광범위하게 사용되는 최적화 파라미터 탐색 방법
- 임의의 시작점에서 출발하여 반복적으로 최적의 파라미터 값을 찾음



- 기울기 > 0
  → x가 커질 수록 함수 값이 커짐
- 기울기 < 0
  → x가 커질 수록 함수 값이 작아짐
- 기울기 값이 크다
  → x 위치가 최소/최댓값의 x 좌표로부터 멀리 떨어져 있음

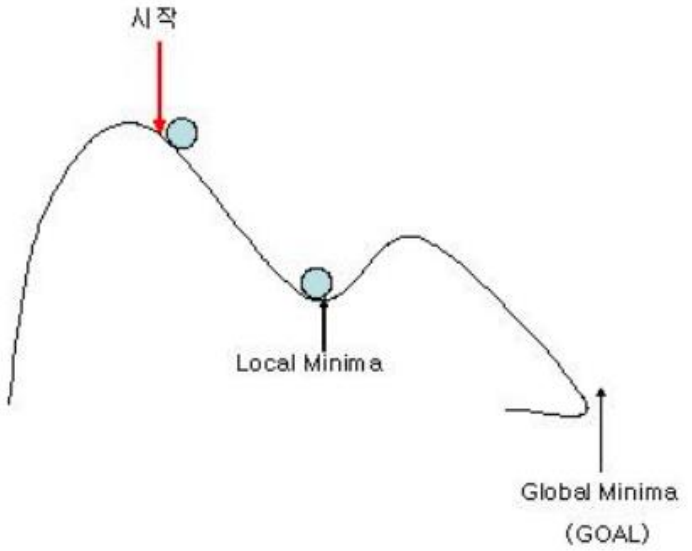
➔ *경사 하강법은 음의 기울기 방향을 따라 탐색하여 최적의 해를 구함*

➔ *로컬 미니멈 함정에 빠질 수 있음!*

https://angeloyeo.github.io/2020/08/16/gradient_descent.html

4 글로벌 미니멈과 로컬 미니멈

로컬 미니멈의 함정에서 탈출하는 방법



https://sens.tistory.com/404

[1]
- 다양한 파라미터 조합 값을 이용하여 신경망을 여러 차례 초기화
- 오차가 가장 작은 해를 최종 파라미터로 선정

[2]
- **담금질 기법 (simulated annealing)** 사용
- 각 스텝 마다 일정한 확률로 현재 해 보다 나쁜 결과를 받는 방식으로 로컬 미니멈 탈출

[3]
- **스토캐스틱 경사하강법 (stochastic gradient descent)** 사용
- 정확하게 모든 기울기 계산 X
- 입력 데이터를 하나씩만 처리하면서 가중치를 업데이트 하는 방법
- 로컬 미니멈에 빠지게 되더라도 계산 값이 0이 아닐 수 있음

5 기타 신경망

1. Radial Basis Function (RBF) 신경망
2. Adaptive Resonance Theory (ART) 신경망
3. Self-Organizing Map (SOM) 신경망
4. 중첩된 상호연관 신경망
5. 엘만 네트워크
6. 볼츠만 머신

5 기타 신경망

5.1. RBF 신경망 : 단일 은닉층 순방향 신경망

$$\varphi(x) = \sum_{i=1}^{q} \omega_i \rho(x, c_i)$$

q : 은닉층 뉴런 개수

c_i : i번째 은닉층 뉴런의 중심

ω_i : i번째 은닉층 뉴런의 가중치

$\rho(x, c_i)$: 방사 함수 ➔ 샘플 x에서 데이터 중심 c_i 사이의 거리

자주 사용되는 가우스 방사 함수 식 : $\rho(x, c_i) = e^{-\beta_i \|x - c_i\|^2}$

[훈련 방식]

1. 뉴런 중심 c_i를 결정함 (랜덤 샘플링, 클러스터링 등 사용)
2. 오차 역전파 알고리즘 사용하여 파라미터 ω_i, β_i 선택

5 기타 신경망

5.2. ART 신경망 : 경쟁형 학습의 중요한 알고리즘 중 하나

*** 경쟁형 학습 : 신경망의 출력 뉴런들이 서로 경쟁하여 단 하나의 승자 뉴런만 살아남고, 다른 뉴런들은 억제 상태에 들어감 (승자 독식 원칙)**

- 비교층, 식별층, 식별 임곗값, 치환 모듈로 구성됨.

  - 비교층: 입력 샘플을 받는 역할, 이를 식별층 뉴런으로 전달
  - 식별층: 각 뉴런에 대응하는 하나의 패턴 부여

- 비교층의 입력 신호를 받은 다음, 식별층 뉴런들은 서로 경쟁하여 승자 뉴런을 생성함
- 경쟁 → 입력 벡터와 각 식별층 뉴런에 대응하는 대표 벡터 간의 거리를 계산하여, 거리가 가장 작은 것이 승리

- 승자 뉴런은 다른 식별층 뉴런들을 향해 신호 보내고 활성 억제함.

* **식별 임곗값** → ART 신경망 성능에 큰 영향 미침

- ➢ 비교적 높음 : 입력 샘플은 더 많고 자세한 패턴이 생성될 것임
- ➢ 비교적 낮음 : 양이 적고 비교적 간단한 패턴이 생성될 것임

5 기타 신경망

5.2. ART 신경망

[안정성-유연성 딜레마]

- 안정성 : 비정상 데이터가 입력되어도 시스템의 동작이 변하면 안됨
- 유연성 : 의미 있는 데이터가 입력되었을 때 시스템은 이를 받아들여 동작을 변화시켜야 함
- 딜레마 : 유연성과 안정성 사이에 어떻게 균형을 잡아야 하는지?

- ART 신경망 → 안정성-유연성 딜레마를 비교적 잘 완화한 신경망

[ART 신경망 종류]

- 초기 ART (ART-1) : 바이너리 형태의 데이터만 입력 가능함
- ART-2 : 아날로그 형태의 데이터 입력 가능
- ART-3 : 시냅스의 화학적 구성을 모방한 시스템
- FuzzyART : 모자이크 처리 결합
- ARTMAP : 지도 학습이 가능한 신경망

5 기타 신경망

5.3. SOM 신경망 : 경쟁 학습형 비지도 학습 신경망

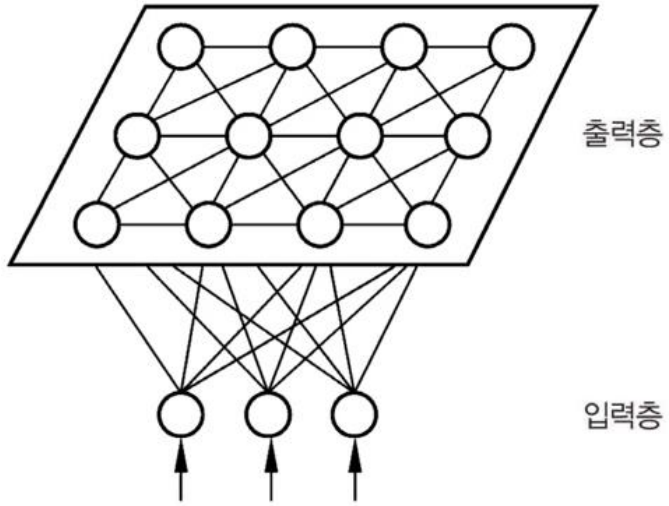


그림 5.11 ＼ SOM 네트워크 구조

- 각 뉴런은 하나의 가중치를 가짐
- 신경망은 입력 벡터를 받은 다음 출력층의 승자 뉴런을 결정함
- 승자 뉴런은 해당 입력 벡터의 저차원 공간에서의 위치를 결정

SOM 신경망의 훈련 목적 : 각 출력층 뉴런을 위해 적절한 가중치를 찾아 위상학적 구조를 유지시키는 것

[훈련 방법]

1. 하나의 훈련 샘플을 받은 후, 각 출력층 뉴런은 해당 샘플과 원래 가지고 있던 가중치 벡터 간의 거리 계산
2. 거리가 가장 가까운 뉴런이 승자 뉴런이 됨 (최적 매칭 유닛; best matching unit)
3. 최적 매칭 유닛과 이웃 뉴런들의 가중치는 조정됨
4. 이 가중치와 입력 샘플의 거리를 축소시킴
5. 1~4 과정을 수렴할 때 까지 계속 반복

5 기타 신경망

5.4. 중첩된 상호연관 신경망

- '**중첩**', '**상호연관**'의 두 주요 성분으로 구성되어 있음
  - 중첩 : 겹겹이 연결되어 만들어진 계층 구조
  - 상호 연관 : 새로운 뉴런 출력의 최대화를 통해 네트워크 오차 간의 상관성으로 관련 파라미터를 훈련시킴

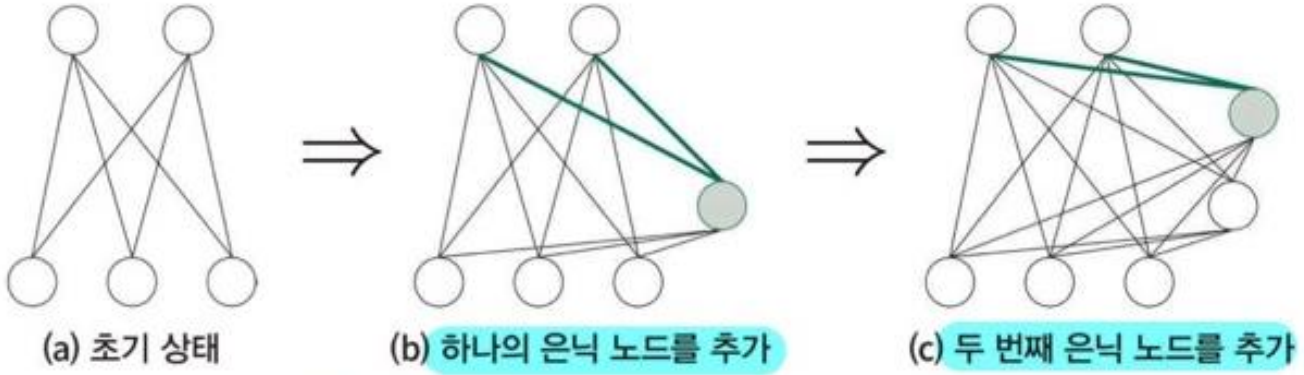


그림 5.12 ＼ 중첩된 상호연관 신경망의 훈련 과정

- 일반적인 순방향 신경망과 비교 시, 중첩된 상호연관 신경망은 네트워크 층 수, 은닉층 뉴런 수를 설정하지 않아도 되며 훈련 속도가 비교적 빠름
- 데이터가 적을 때에는 쉽게 과적합에 빠질 수 있음

5 기타 신경망

5.5. 엘만 네트워크 : 순환 신경망 중 한 종류

* 순환 신경망 : 어떤 뉴런의 출력은 입력 신호로 다시 되돌아감.
t 시점의 네트워크 출력 상태가 t-1 시점의 네트워크 상태와도 연관될 수 있음

- 은닉층 뉴런의 출력이 다시 돌아와 다음 시간대의 입력층 뉴런에 제공되는 신호와 함께 다음 시간대의 입력이 되어 은닉층으로 흘러 들어감

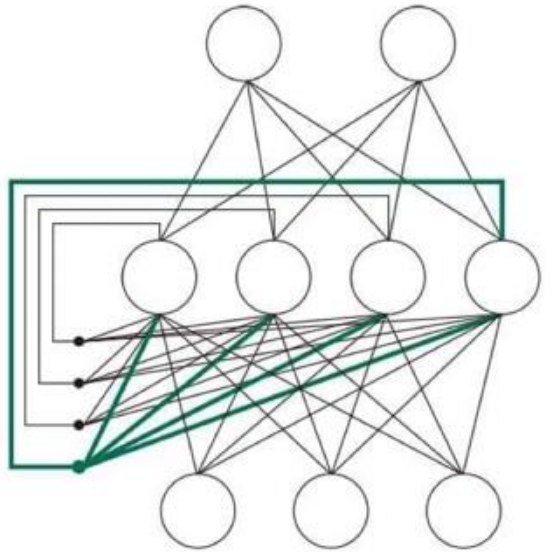


그림 5.13 ㅣ 엘만 네트워크 구조

- 은닉층 뉴런 → 일반적으로 시그모이드 활성화 함수를 사용
- 네트워크의 훈련 → 오차 역전파 알고리즘을 활용하여 진행됨

5 기타 신경망

5.6. 볼츠만 머신 : 에너지 기반 모델 중 한 종류

*** 에너지 기반 모델 : 네트워크 상태를 하나의 에너지라고 정의하였을 때, 에너지 함수를 최소화 하는 모델**

- 뉴런은 가시층, 은닉층으로 구성되어 있음
- 볼츠만 머신의 뉴런은 모두 바이너리 형태 (0: 억제, 1: 활성화)

볼츠만 머신의 훈련 과정 : 각 훈련 샘플을 하나의 상태 벡터라고 간주하고 얻을 수 있는 확률을 최대화 하는 것

일반 볼츠만 머신 -> 완전 연결 그래프 -> 네트워크 훈련 복잡도 매우 높음

➔ 제한된 볼츠만 머신을 주로 더 많이 사용!

(* 제한된 볼츠만 머신 : 입력층과 은닉층 사이의 연결만 보존됨.)

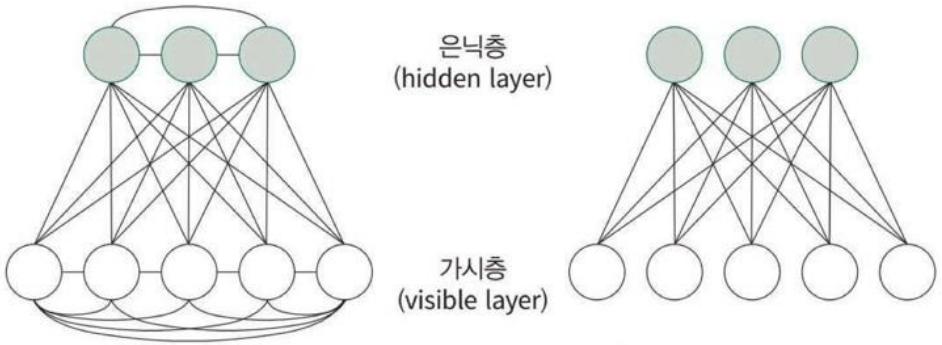


그림 5.14 볼츠만 머신과 제한된 볼츠만 머신

6 딥러닝

- 전형적인 딥러닝 모델 : 깊은 층을 쌓는 신경망
- 신경망 모델의 능력을 향상시킬 수 있는 가장 간단한 방법 → 은닉층의 개수를 늘리는 것

- 은닉층 뉴런의 수를 증가시키는 것 vs 은닉층 개수를 증가시키는 것 ?
  → 은닉층 개수를 늘리는 것이 모델 복잡성을 증가시키는데 더 효율적임
  → 이유 : 은닉층 수 증가 → 활성화 함수 가진 뉴런 개수 증가 & 활성화 함수가 내장된 층 수도 증가

But, 여러 은닉층 가진 신경망 → 전통적인 알고리즘 (일반 오차 역전파 알고리즘)을 사용하여 훈련시키기 힘듦
이유 : 오차가 은닉층에서 역전파될 때 수렴하지 못하고 발산하는 경우가 생김

➔ *'사전훈련 + 파인튜닝' 방법 / 가중치 공유 방법*

6 딥러닝

비지도 레이어-와이즈 훈련 (unsupervised layer-wise training) → '사전훈련 + 파인튜닝' 방법

: 다수의 은닉층을 가진 네트워크를 훈련하는 데 유용한 수단

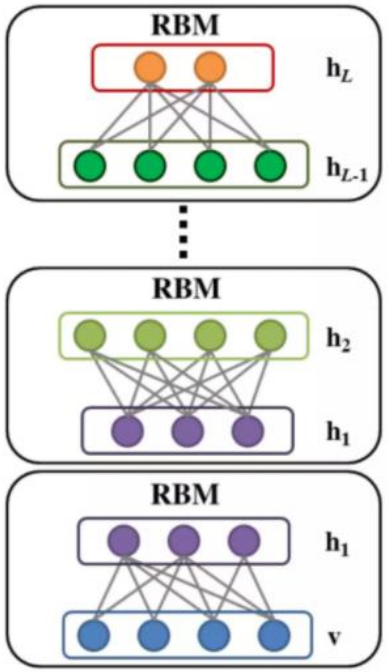
- 한 번에 한 개의 은닉 노드 훈련
- 훈련 시 앞 층 은닉 노드의 출력을 입력으로 받음
- 해당 층의 은닉 노드 출력은 다음 층 은닉 노드의 입력이 됨.

} **사전 학습 (pre-training)**

- 사전 학습이 모두 완료된 후, 다시 모든 네트워크에 대해 **파인 튜닝 (fine-tuning)** 훈련을 진행

[예시 : 심층 신뢰망 (Deep Belief Network; **DBN**)]

- DBN : 입력층과 은닉층으로 구성된 RBM(제한된 볼츠만 머신)을 블록처럼 여러 층으로 쌓은 신경망



- 첫 번째 층을 훈련함.
- 이는 훈련 샘플의 RBM 모델에 대한 것이며, 일반적인 RBM을 따라 훈련시킴.
- 첫 번째 사전훈련이 완료된 은닉 노드를 두 번째 층의 입력 노드로 보고 두 번째 층에 대해 사전 훈련을 진행함
- 각 층에 대한 사전 훈련이 완료된 후 오차 역전파 알고리즘을 활용하여 모든 네트워크에 대해 파인튜닝 진행함

http://blog.skby.net/%EC%8B%AC%EC%B8%B5%EC%8B%A0%EB%A2%B0%EB%A7%9D-dbn-deep-belief-network/

6 딥러닝

가중치 공유 (weight sharing)

- 같은 조합에 속한 뉴런에 같은 가중치 사용
- 합성곱 신경망 (CNN)에서 중요한 역할을 함

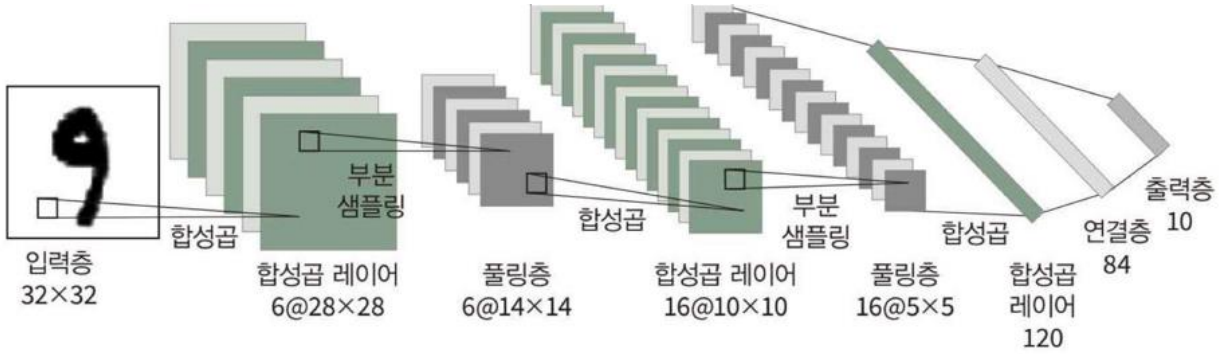


그림 5.15 ＼ 수기 숫자 이미지 인식에 사용되는 CNN[LeCun et al., 1998]

[예시 : 합성곱 신경망 (Convolution Neural Network; **CNN**)]

- CNN : 다수의 '합성곱 레이어'와 '샘플링 레이어'를 쌓아 올려 입력 신호를 가공하고 연결층에서 출력 목표 간의 매핑을 실현

  - 첫 번째 합성곱 레이어 → 6개의 피처맵으로 구성
  - 각 피처맵은 28x28 크기의 뉴런 행렬
  - 각 뉴런은 필터를 통해 부분 특성을 추출 (샘플링 레이어; 풀링 층)
  - 이를 반복한 후, 원본 이미지 데이터는 120 차원의 특성 벡터로 매핑됨.
  - 마지막으로 84개의 뉴런으로 구성된 연결층과 출력층을 연결하여 판별 문제를 완료

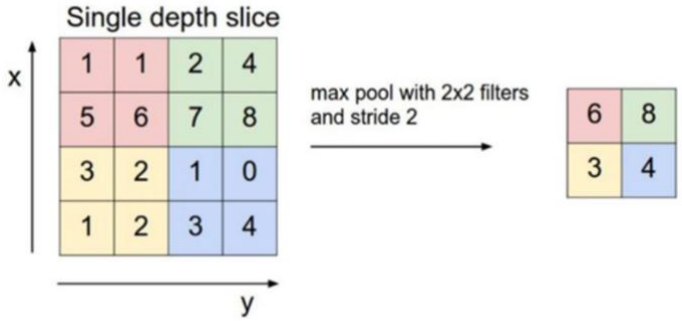


그림 5. Subsampling Layer의 동작과정 중 Max Pooling을 활용한 특징 추출의 예

https://m.blog.naver.com/rainyday1983/222011720826

7 예제 문제 (SOM 구현)

miniSom 코드 : https://github.com/JustGlowing/minisom/minisom.py

```
class MiniSom(object):
    def __init__(self, x, y, input_len, sigma=1.0, learning_rate=0.5,
                 decay_function=asymptotic_decay,
                 neighborhood_function='gaussian', topology='rectangular',
                 activation_distance='euclidean', random_seed=None):
```

```
x : int
    x dimension of the SOM.
y : int
    y dimension of the SOM.
input_len : int
    Number of the elements of the vectors in input.
sigma : float, optional (default=1.0)
    Spread of the neighborhood function, needs to be adequate
    to the dimensions of the map.
    (at the iteration t we have sigma(t) = sigma / (1 + t/T)
    where T is #num_iteration/2)
learning_rate : initial learning rate
    (at the iteration t we have
    learning_rate(t) = learning_rate / (1 + t/T)
    where T is #num_iteration/2)
decay_function : function (default=None)
    Function that reduces learning_rate and sigma at each iteration
    the default function is:
                learning_rate / (1+t/(max_iterarations/2))
    A custom decay function will need to to take in input
    three parameters in the following order:
    1. learning rate
    2. current iteration
    3. maximum number of iterations allowed
    Note that if a lambda function is used to define the decay
    MiniSom will not be pickable anymore.
neighborhood_function : string, optional (default='gaussian')
    Function that weights the neighborhood of a position in the map.
    Possible values: 'gaussian', 'mexican_hat', 'bubble', 'triangle'
topology : string, optional (default='rectangular')
    Topology of the map.
    Possible values: 'rectangular', 'hexagonal'
activation_distance : string, optional (default='euclidean')
    Distance used to activate the map.
    Possible values: 'euclidean', 'cosine', 'manhattan', 'chebyshev'
random_seed : int, optional (default=None)
    Random seed to use.
```