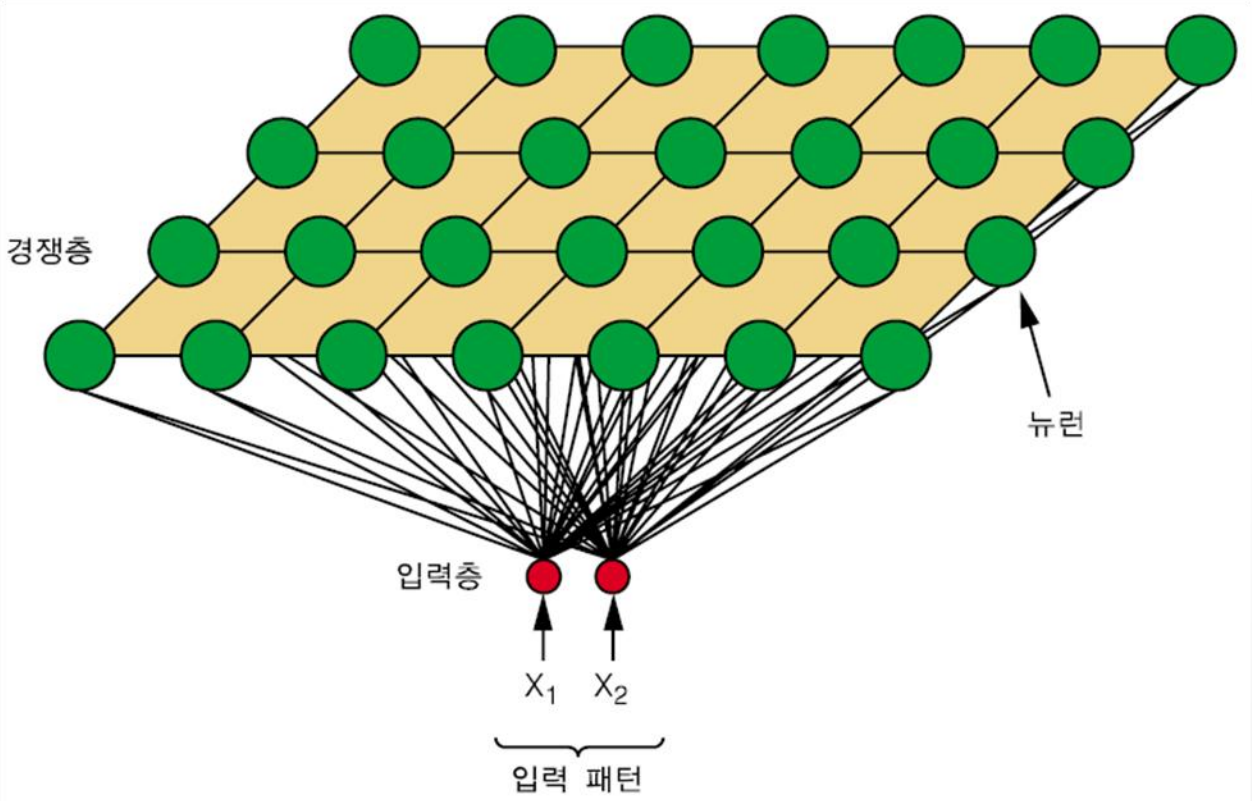
Python

자기 조직화 특징 지도

자기 조직화 특징 지도

❖ 자기 조직화 특징 지도

- 첫 번째 층은 입력층이라고 하며, 두 번째 층은 2차원의 격자로 된 경쟁층으로 되어 있다.
- 모든 연결들은 첫 번째 층에서 두 번째 층의 방향으로 완전 연결(fully connected)된 구조다.



[그림 14-2] SOFM 신경망의 구조

자기 조직화 특징 지도

- 자기 조직화 특성 지도
  - 다층 퍼셉트론 대표적인 분류기 중 하나로, 교사학습에 의해서 학습을 수행.
  - 자기 조직화 특성지도는 비 교사 학습으로, 가중치를 조정하여 비슷한 입력 값에 대하여 비슷한 출력을 낼 수 있도록 출력 층을 스스로 조직화하는 신경망.
  - 입력 층과 출력 층만을 가지는 단층 신경망으로 출력층을 2차원 격자 구조로 배치.
  - 출력 층의 하나의 뉴런은 입력 층의 모든 뉴런과 가중치를 가지고 연결됨.

자기 조직화 특징 지도

- 자기 조직화 특성 지도

1. 모든 출력 뉴런에 대하여 주어진 입력 값 x와 가중치 w_j와의 거리는 계산

$$d(x, w_j) = |x - w_j|^2 = \sum_{i=1}^{n} (x_i - w_{ij})^2$$

2. 모든 출력 뉴런들 중 거리 값이 가장 작은 출력 뉴런을 찾아 승자로 둠.

$$\omega = argmin_j\{d(x, w_j)\}$$

자기 조직화 특징 지도

- 자기 조직화 특성 지도

3. 승자가 된 뉴런만 출력 값 1을 가지고, 나머지 뉴런들은 0의 출력 값을 가짐.

$$w_j^{(r+1)} = w_j^{(r)} + \eta(x - w_j^{(r)})$$

- $w_j^{(r)}$는 j번째 출력 뉴런의 현재의 가중치 파라미터이고, $w_j^{(r+1)}$는 수정된 후 얻어지는 가중치 파라미터.
- 학습식은 결국 입력 패턴과 현재 가중치의 차이에 작은 상수 값(η, 학습률)을 곱하여 더해줌으로써 입력 패턴에 조금씩 가까워지도록 조정하는 역할을 함.

자기 조직화 특징 지도

- 자기 조직화 특성 지도

4. α는 승자 뉴런과의 이웃관계에 따라 수정의 폭을 결정하는 값이 됨.

$$w_j^{(r+1)} = w_j^{(r)} + \eta\alpha(x - w_j^{(r)})$$

- 만약 j 번째 뉴런이 승자 뉴런의 이웃이 아니라면 α = 0으로 둠으로써 학습이 일어나지 않도록 함.
- j 번째 뉴런이 승자 뉴런의 이웃인 경우에는 0에서 1 사이의 값으로 두어 승자 뉴런보다는 작인 폭의 수정이 일어날 수 있도록 하면 됨.

자기 조직화 특징 지도

- 자기 조직화 특성 지도

5. 뉴런들 사이의 거리가 계산되어야 함.

$$d(j,w) = (r_j - r_w)^2 + (c_j - c_w)^2$$

- 2차원 배열된 출력 층에서의 거리이므로, j번째 출력 뉴런의 2차원 배열에서의 위치를 (r_j 행, c_j 열)로 나타내면, j번째 출력 뉴런에서 승자 뉴런 w까지의 거리

6. 거리에의 소속 도는 가우시안 함수를 사용하여 다음과 같이 정의할 수 있음.

$$\alpha_j = \frac{1}{\sqrt{2}\sigma} \exp\{-\frac{1}{2}\frac{\big(d(j,w)\big)^2}{\sigma^2}\}$$

자기 조직화 특징 지도

- 자기 조직화 특성 지도
  - σ는 소속도 함수의 퍼진 정도를 나타내는 값으로 그 값이 크면 넓은 범위까지 이웃으로 간주하는 것이 되며, 그 값이 작으면 이웃의 범위를 좁히는 결과가 됨.
  - σ의 값은 학습초기단계에서는 크게 주어 이웃의 범위를 넓히고, 학습이 진행됨에 따라 점점 줄여서 학습의 안정도를 높이는 방식을 취함.
  - σ는 학습횟수 t 에 대한 함수로 다음 식을 이용하여 조정할 수 있음.

$$\sigma(t) = \sigma_0 \exp\{-\frac{t}{T}\}$$

  - T는 설계자가 결정하는 상수 값.
  - j번째 출력 뉴런에 대한 소속도 α_j는 승자 뉴런 w와 학습횟수 t에 의존하는 함수가 되어 $\alpha_{wj}{}^{(t)}$로 나타낼 수 있음.

자기 조직화 특징 지도

- 자기 조직화 특성 지도

```
Dim = 2   # 입력 차원
N = 1000  # 학습데이터 수
X = 2 * np.random.rand(N, Dim) - 1 # 입력데이터 생성
INP = X.shape[1]
R = 10 # 입력 뉴런수
C = 10 # 입력 뉴런수
OUT = R*C #출력 뉴런 수
w = np.random.rand(INP, OUT) * 0.6 - 0.3
eta = 0.5 # 학습률
sig = 1 # 이웃소속도
mStep = 10 # 반복 횟수
```

자기 조직화 특징 지도

```
for j in range(2, mStep, 1): # 반복 학습 시작
    for i in range(0, N, 1): # 각 데이터에 대한 학습 시작
        x = X[i, :]        # 입력 데이터 선택
        dist = np.zeros((1, OUT))
        for k in range(0, OUT, 1):  # 출력 뉴란 가중치와의 거리계산
            dist[0, k] = np.dot((x - np.array([w[:, k]])), (x - np.array([w[:, k]])).T)
        wini = np.argmin(dist) # 승자뉴런 선정

        alpha = np.zeros((1, OUT))
        dw = np.zeros((INP, OUT))
        for k in range(0, OUT, 1):
            alpha[0, k] = som_alpha(wini, k, sig, C) # 소속도 계산
            dw[:, k] = (x - np.array([w[:, k]])) * alpha[0, k] # 가중치 수정항 계산

        w = w + eta * dw # 가중치 수정
        sum_dw = np.zeros((1, N))
        sum_dw[0, i] = np.trace(np.dot(dw.T, dw)) # 가중치 변화량 계산

    sig = sig*0.9 # 이웃소곡도 파라미터 감소
    if np.mean(sum_dw) < 0.001: # 가중치 변화 없을 시 학습 완료
        j = mStep + 1
```

자기 조직화 특징 지도

```
def som_alpha(i, j, sig, NumC):
    x1 = np.zeros((1, 2))
    x2 = np.zeros((1, 2))

    x1[0, 0] = np.floor(i / NumC)
    x1[0, 1] = np.mod(i, NumC)
    x2[0, 0] = np.floor(j / NumC)
    x2[0, 1] = np.mod(j, NumC)

    r = np.dot((x1 - x2), (x1 - x2).T)
    return np.exp(-r / (2 * pow(sig, 2)))*(1 / (np.sqrt(2*np.pi) * sig))
```

자기 조직화 특징 지도

- Exercise
  - 주어진 코드를 이용하여 입력 층 2, 출력 층 2 X 4의 SOM을 학습하고 결과값을 시각적으로 출력하시오.