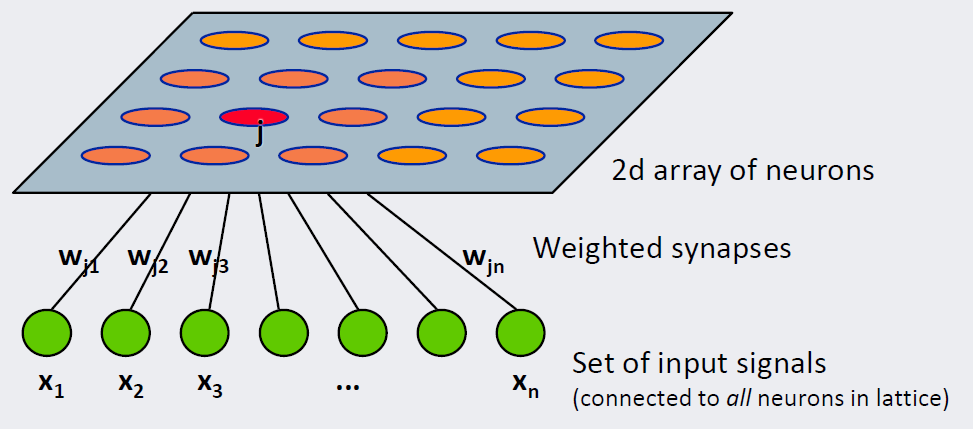
SOM

* 자기조직화 지도

사람이 눈으로 볼 수 있는 저차원(2차원 내지 3차원) 격자에 고차원 데이터의 각 개체들이 대응하도록 인공신경망과 유사한 방식의 학습을 통해 군집을 도출해내는 기법

저차원 격자에서의 유사도는 고차원 입력 공간에서의 유사도를 최대한 보존하도록 학습



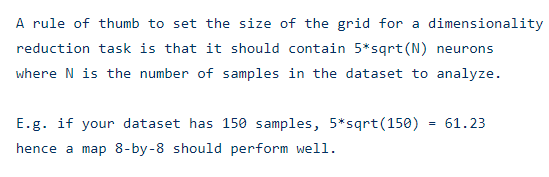
격자(점) 하나에 여러 개의 입력 벡터(row) 속할 수 있음

하나의 격자에 속한 입력벡터(row)들 끼리는 서로 가까운 곳에 있음

입력값이 들어오면 가장 가까운 격제를 찾음 <- Winning node

* 격자에 입력값 할당하면 군집화

격자 개수 공식



Sqrt = 루트

학습과정

1. 격자 개수 먼저 정하기
2. 격자 위치 랜덤으로 초기화
3. 학습데이터를 추가하며 데이터에 맞게 격자의 위치를 조정 (데이터에 가장 가까운 격자가 많이 움직이고 멀수록 움직이지 않는다.)
4. 학습이 끝난 후 조정된 격자에 데이터 할당하여 클러스터링 or 차원축소

<https://github.com/JustGlowing/minisom> <- Github주소

**import** **pandas** **as** **pd**

*# data from*

*# https://archive.ics.uci.edu/ml/datasets/Sales\_Transactions\_Dataset\_Weekly*

sales\_transaction = pd.read\_csv('데이터')

som = MiniSom(맵의 x, 맵의 y, ‘데이터 row 1개에 들어있는 데이터 shape’, sigma=2., learning\_rate=0.5,

neighborhood\_function='gaussian', random\_seed=10)

som.pca\_weights\_init(data)

print("Training...")

som.train\_batch(data, 50000, verbose=**True**) *# random training*

print("**\n**...ready!")

input은 array 형식으로 넣어주세요

sigma는 어느정도 까지의 거리의 격자를 움직일 것이냐를 결정(default=1.0, 사용자가 결정해야함)

neighborhood\_function은 'gaussian', 'mexican\_hat', 'bubble', 'triangle'에서 골라서 쓰세요

activation\_distance 은'euclidean', 'cosine', 'manhattan'에서 적절한거 골라서 쓰세요

pca\_weights\_init : Initializes the weights to span the first two principal components

train\_batch(데이터,반복횟수(=epoch),vervose=) vervose가 True이면 각 반복횟수마다 그때의 상태 출력