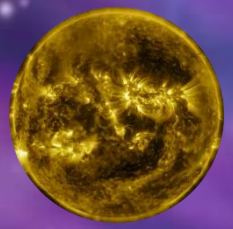
METODY INŻYNIERII WIEDZY

KNOWLEDGE ENGINEERING AND DATA MINING

NEURONOWE MAPY SAMOORGANIZUJĄCE SIĘ

ĆWICZENIA



Self-Organizing Maps SOM

Adrian Horzyk

Akademia Górniczo-Hutnicza

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Katedra Automatyki i Inżynierii Biomedycznej, Laboratorium Biocybernetyki

30-059 Kraków, al. Mickiewicza 30, paw. C3/205

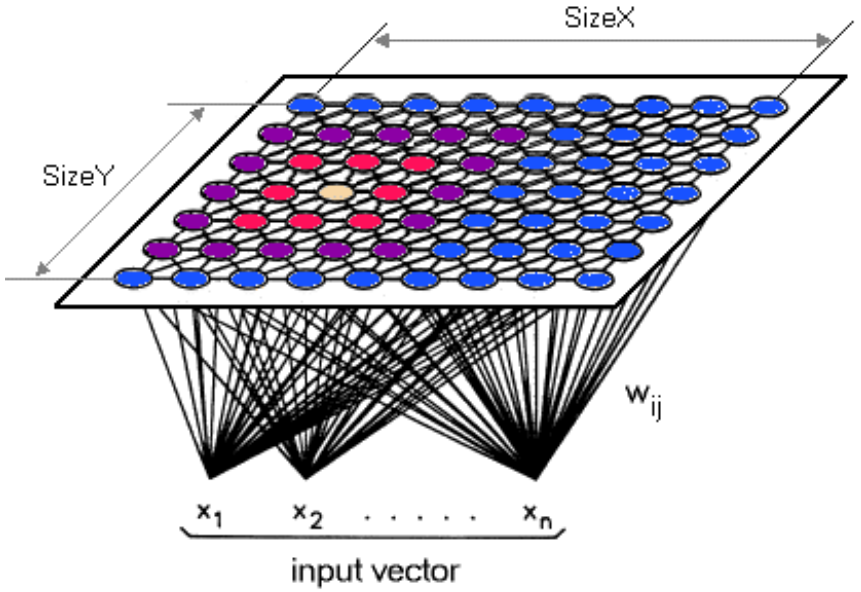
horzyk@agh.edu.pl, Google: Adrian Horzyk

ALGORYTM UCZENIA SIECI SOM

1. Zbuduj wyjściową mapę pseudoneuronów (zwykle 2D).
2. Inicjalizuj wagi każdego pseudoneuronu niewielkimi liczbami losowymi różnymi od 0.
3. Bierz kolejno albo losuj kolejne wektory $X_k = \{x_1, x_2, \ldots, x_n\}$ ze zbioru uczącego $X_1, \ldots, X_K$.
4. Oblicz wartość wyjściową wszystkich pseudoneuronów mapy sieci SOM wg poniższego wzoru, wyznaczającego odległość wzorca wejściowego do wektora wag.
5. Określ, który psedoneuron został najmocniej pobudzony dla $X_k = \{x_1, x_2, \ldots, x_n\}$.
6. Aktualizuj wagi pseudoneuronu zwycięskiego w taki sposób, żeby lepiej odwzorowywały wektor wejściowy.
7. Aktualizuj wagi sąsiednich pseudoneuronów w podobny sposób, lecz z malejącym współczynnikiem im dalej znajdują się od pseudoneuronu zwycięskiego.
8. Wróć do kroku 3 dopóki wszystkie wzorce uczące nie zostaną reprezentowane w sieci z wystarczającą dokładnością.

Zwycięski pseudoneuron określamy zwykle na podstawie jego najbliższej odległości Euklidesa od wektora wag:

$$d\left(X_k, W_{i,j}(t)\right) = \sqrt{\sum_{i=1}^{I}\sum_{j=1}^{J}\left(X_k - W_{i,j}(t)\right)^2}$$



WYZNACZENIE ZWYCIĘZCY

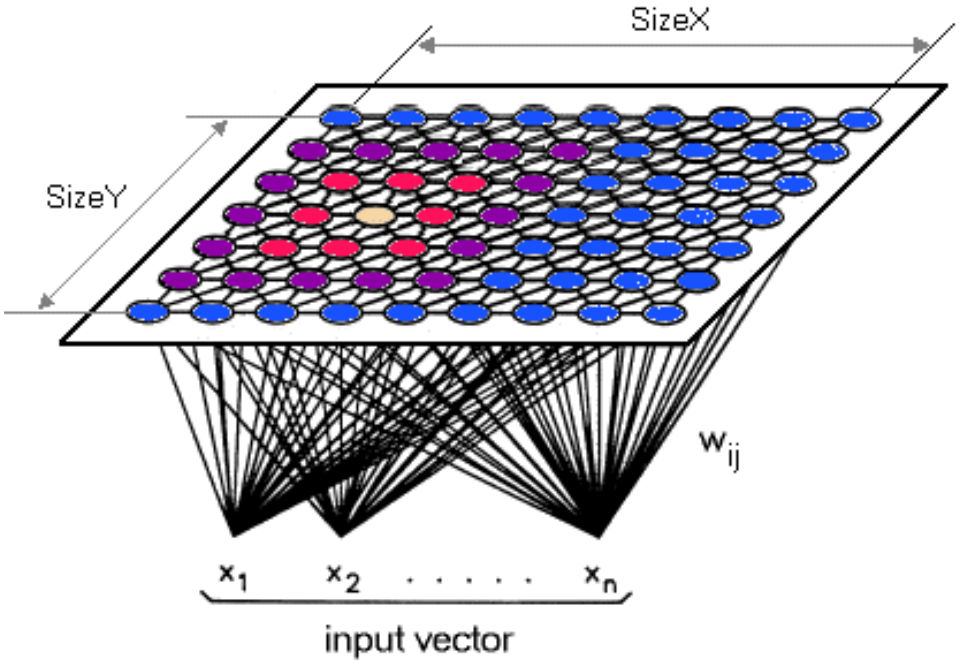
Wyznaczamy kolejno odległości Euklidesa poszczególnych wzorców uczących X_1, …, X_K względem wszystkich wektorów wag $W_{1,1}$, …, $W_{I,J}$, gdzie I x J to ilość węzłów w 2D sieci SOM:

$$d\left(X_k, W_{i,j}(t)\right) = \sqrt{\sum_{i=1}^{I}\sum_{j=1}^{J}\left(X_k - W_{i,j}(t)\right)^2}$$

W trakcie obliczania tych odległości wyznaczamy równocześnie argumenty (i, j) dla najbliższego wektora wag do każdego wzorca uczącego:

$$(a, b) = arg \min_{i,j} d\left(X_k, W_{i,j}(t)\right)$$

który będzie zwycięzcą aktualizującym swoje wagi najmocniej oraz względem którego będą wyznaczane odległości do sąsiednich pseudoneuronów, których wagi będą aktualizowane słabiej.



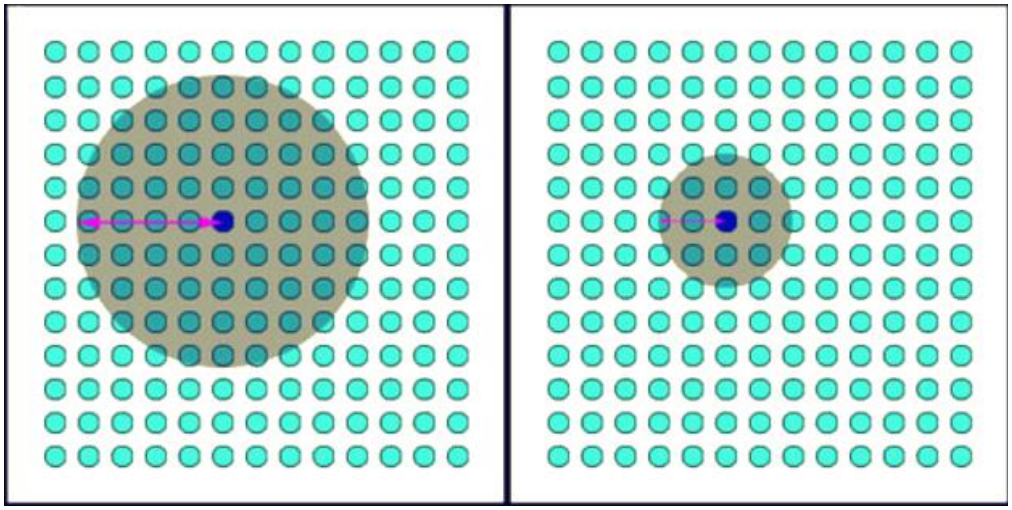
PRZEBIEG UCZENIA SIECI SOM

W trakcie uczenia sieci SOM zakres aktualizowanych sąsiadów stopniowo maleje. Na początku zasięg aktualizacji sąsiadów jest duży i stopniowo zawęża się, co można matematycznie wyrazić wzorem określającym zmniejszający się promień odległości pseudoneuronów (względem pseudoneuronu zwycięskiego) ulegających aktualizacji:

$$\sigma(t) = \sigma_0 \cdot e^{-\frac{t}{\alpha}}$$

gdzie σ_0 określa pewien promień początkowy, który na początku może obejmować nawet całą siatkę neuronów, czyli np. $\sigma_0 = max\{I, J\}$.

w zależności od czasu, jaki upłynął od początku nauki t_0 i pewnej stałej zawężania α, np. α = 1000.



Podobnie wyznaczamy współczynnik siły adaptacji wag:

$$\gamma(t) = \gamma_0 \cdot e^{-\frac{t}{\alpha}}$$ np. γ_0 = 1

Wyznaczając wagi według zależności:

$$W_{i,j}(t+1) = W_{i,j}(t) + \delta(t) \cdot \gamma(t) \cdot \left(X_k - W_{i,j}(t)\right)$$

gdzie $\delta(t)$ jest współczynnikiem zależnym od odległości pseudoneuronu względem zwycięzcy: $\delta(t) = e^{-\frac{d\left(N_{i,j}(t), N_{a,b}(t)\right)^2}{2 \cdot \sigma^2(t)}}$ gdzie $N_{i,j}(t)$ to neuron umieszczony na pozycji (i,j)

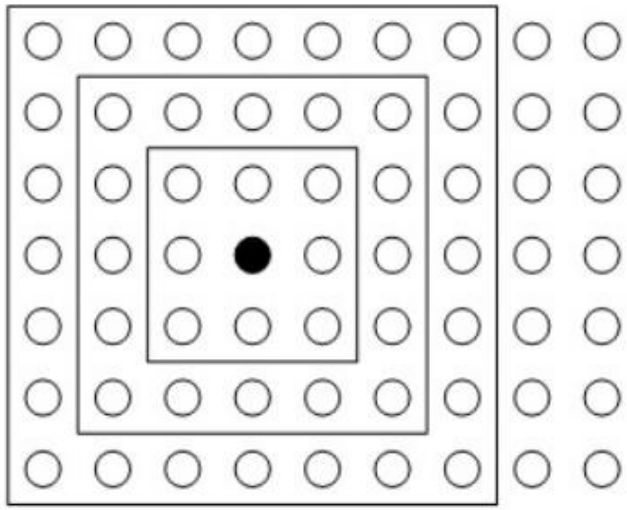
PRZEBIEG UCZENIA SIECI SOM

Odległość pseudoneuronów (węzłów) względem zwycięzcy liczymy w zależności od przyjętej siatki, oraz miary odległości (Euklidesowej lub innej metryki) np.:

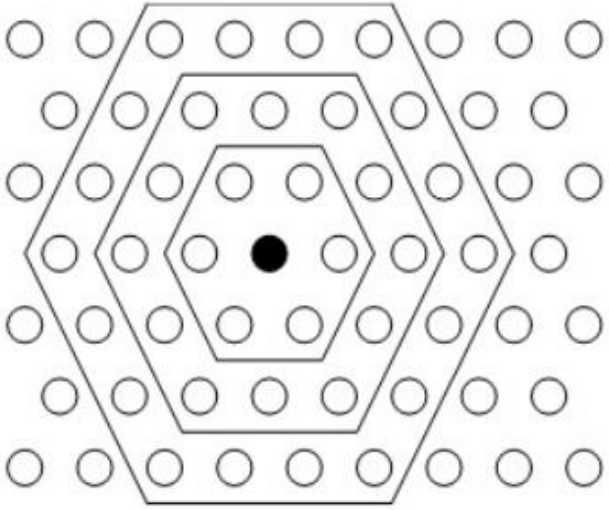
$$d\left(N_{i,j}(t), N_{a,b}(t)\right) = |i-a| + |j-b|$$

$$d\left(N_{i,j}(t), N_{a,b}(t)\right) = \sqrt{(i-a)^2 + (j-b)^2}$$

Możemy przyjąć różne rodzaje siatek węzłów oraz w różny sposób określać te najbliższe:



Siatka prostokątna



Siatka heksagonalna

POPRAWA JAKOŚCI UCZENIA SOM

Jakie parametry są istotne z punktu widzenia adaptacji sieci SOM:

- zakres zmienności początkowo wylosowanych niewielkich dodatnich wag różnych od zera, przy czym zakres losowania tych wartości powinien być z zakresu od setnych do dziesiętnych zakresu zmienności dla danego atrybutu, np. jeśli $x_i^{min} \leq x_i \leq x_i^{max}$, wtedy wagi powinny być losowane z zakresu, np.:

$$\frac{x_i^{max} - x_i^{min}}{100} \leq w_i \leq \frac{x_i^{max} - x_i^{min}}{10}$$

- rozmiar danych wejściowych n,
- niezależność atrybutów danych wejściowych $X_k = \{x_1, x_2, \ldots, x_n\}$ i ich ilość,
- szybkość adaptacji i dobór współczynników, tj.: α, przy czym w trakcie nauki $t \leq \alpha$,
- w, co umożliwia określenie ilości stopni swobody w trakcie adaptacji modelu, czyli potencjalnie określenie ilości najbliższych sąsiadów o podobnym stopniu tej bliskości.
- ilości pseudoneuronów (węzłów wyjściowych) siatki, których ilość powinna być znacząco mniejsza niż ilość wzorców uczących po to, żeby „zmusić” je do reprezentacji pewnego podzbioru wzorców, a nie tylko jednego z nich. Ilość tych pseudoneuronów nie może też być mniejsza niż ilość spodziewanych klas. Jeśli znana jest liczba pożądanych klas/grup c, wtedy początkowa ilość neuronów może być określona jako c^m tworząc m-wymiarową hiperkostkę węzłów o wymiarze c.

STRUKTURA I WYMIAR SIATKI

Jak dobrać odpowiednią siatkę węzłów i jej wymiar:

- **Warto określić ilość atrybutów niezależnych m w wektorach/macierzach danych wejściowych, przy czym $1 \leq m \leq n$.**

- **Żeby umożliwić sieci SOM nie tylko określenie grup wzorców, lecz również prawidłowe odwzorowanie odległości pomiędzy grupami, należy odpowiednio określić wymiar przestrzeni siatki v, na jaką będą rzutowane wzorce wejściowe. Ten wymiar na pewno nie może być mniejszy niż ilość atrybutów niezależnych, czyli $m \leq v \leq n$. Wobec tego naukę można rozpocząć od wymiaru m, w razie potrzeby stopniowo go zwiększając, lecz nie więcej niż do wymiaru n.**

Jak rozpoznać, że wymiar siatki węzłów jest odpowiedni czy nieodpowiedni:

- **Rozmiar siatki jest nieodpowiedni, jeśli w wyniku wielu adaptacji rozpoczynających się od różnie wylosowanych wag początkowych odległość neuronów zwycięskich reprezentujących najmocniejsze klasy znacznie się różni. Neurony te nie muszą być w tych samych miejscach, lecz powinny być mniej więcej równo odległe.**

STRUKTURA I WYMIAR SIATKI

Jak rozpoznać, iż ilość neuronów siatki jest niewystarczająca:

- **Jeśli w wyniku nauki dla wielu adaptacji rozpoczynających się od różnie wylosowanych wag początkowych powstają różni lub różnoliczni zwycięzcy w stosunku do innych wyników, oznacza to trudność, którą można rozwiązać stopniowo zwiększając ilość węzłów, lecz nie więcej niż ilość wzorców / 2, a dla bardzo licznych zbiorów wzorców nawet ich pierwiastek, czyli: $\sqrt{ilość\ wzorców}$**

Sposób dobierania ilości neuronów i wymiaru siatki:

1. **Rozpoczynamy od siatki o wymiarze m równym ilości atrybutów niezależnych, a jeśli jest on nieznany wtedy od wartości 1, 2 lub $\sqrt{ilość\ atrybutów}$.**
2. **Początkową ilość neuronów dla danego wymiaru c dobieramy według ilości spodziewanych/pożądanych klas/grup, a więc dla całej siatki c^m.**
3. **Następnie przechodzimy do procesu nauki, badając i porównując ilości reprezentantów dla poszczególnych neuronów po pewnej ilości kroków, np. 1000, rozpoczynając naukę sieci SOM wielokrotnie (np. 10x) od różnie wylosowanych wag początkowych.**
4. **Jeśli otrzymujemy znacznie różniącą się ilość reprezentantów dla poszczególnych pseudoneuronów (węzłów wyjściowych), należy zwiększyć ilość neuronów w którymś lub kilku wymiarach np. o 1.**

STRUKTURA I WYMIAR SIATKI

5. Gdy ilość reprezentantów (wzorców) poszczególnych pseudoneuronów (czyli ilości wzorców reprezentowanych przez poszczególne węzły się w miarę ustabilizuje, wtedy warto policzyć odległości zwycięzców, jako najmniejszą ilość przejść (krawędzi) oddzielających te neurony. Przy czym warto porównywać tych zwycięzców, którzy reprezentują podobne zbiory wzorców wejściowych o podobnej ilości.
6. Jeśli te odległości znacznie się różnią, oznacza to, iż wymiar hiperprzestrzeni m do której zrzutowaliśmy zbiór wzorców uczących jest niewystarczający do poprawnego odwzorowania relacji pomiędzy grupami/klasami.
7. Należy więc zwiększyć wymiar hiperprzestrzeni na m+1 i równocześnie zmniejszyć ilość neuronów dla wszystkich wymiarów i rozpocząć od punktu 2.
8. Gdy odległości pomiędzy zwycięzcami reprezentującymi największą ilość wzorców się ustabilizują dla kilku procesów uczenia rozpoczętych od różnych wag losowych, wtedy najprawdopodobniej udało nam się osiągnąć właściwy wymiar podprzestrzeni siatki SOM oraz prawidłową ilość neuronów.
9. Ponadto można eksperymentować jeszcze z określeniem sąsiedztwa w takiej siatce, czy tylko rozważamy sąsiedztwo ortogonalne czy również po przekątnej. Generalnie to po przekątnej powinno dawać lepsze wyniki i nie powodować konieczności zbyt dużego zwiększania wymiaru siatki SOM, przyspieszając naukę.

Taki proces adaptacji może wymagać kilkudziesięciu a nawet kilkuset adaptacji sieci SOM rozpoczynanej od różnych wag w celu określenia poprawnego (optymalnego) modelu.

JAK ZACZĄĆ?

Tworzymy 2D tablicę (najlepiej kwadratową, aczkolwiek może być prostokątna) węzłów SOM (neuronów). Każdy neuron zawiera tablicę wag oraz zmienną przechowującą obliczoną odległość Euklidesa wektora wag do ostatnio zaprezentowanego sieci wektora wartości wejściowych. Ilość węzłów/neuronów tablicy powinna być znaczenie mniejsza niż ilość danych uczących. Dla 150 wzorców Irys np. 4x4 lub 5x5, tak żeby każdy węzeł/pseudoneuron reprezentował przynajmniej kilka podobnych wzorców jednej klasy.

Jest to metoda uczenia nienadzorowanego, więc nie podajemy na wejściu informacji o klasie. Nazwy klas wykorzystujemy tylko do sprawdzenia, jak uporządkowały nam się w strukturze tabelarycznej wzorce uczące.

W programie tworzymy 2 zagnieżdżone pętle obliczeniowe przechodzące po wszystkich węzłach/pseudoneuronach sieci SOM. Przechodzimy po tych neuronach 2x dla każdego wektora uczącego:

1. Obliczając dla każdego pseudoneuronu jego odległość Euklidesa do wejścia oraz równocześnie wyznaczając, który pseudoneuron uzyskał maksymalne pobudzenie.
2. Wyznaczamy zwycięzcę i obliczamy zmiany wektorów wag dla każdego neuronu sieci.

Tą procedurę powtarzamy dla wszystkich wzorców uczących tak długo dopóki wartości wag zmieniają się znacząco, tj. wielkość ich zmian jest większe niż pewna stała (np. 0.001). Jeśli zmiany wag są znikome, wtedy przerywamy proces uczenia.

Wzorce na początku najlepiej wybierać losowo ze zbioru uczącego lub jeśli znamy klasy, to wybierać je z różnych klas w celu przyspieszenia dopasowania się sieci SOM. W odwrotnym przypadku pseudoneurony/węzły zostaną wykorzystane przez wzorce jednej klasy i będą musiały się przeuczyć.

Wizualne sprawdzanie wyników dla Iris

W celu wizualnego sprawdzenia pośrednich i końcowych wyników uczenia się sieci SOM (animacja 2D) proszę każdej z klas zbioru przyporządkować jeden z kolorów podstawowych, np.:

- Setosa – Czerwony (Red)
- Versicolor – Zielony (Green)
- Virginica – Niebieski (Blue)

W trakcie nauki sieci SOM w siatce neuronów 2D dochodzi do tego, iż każdy z pseudoneuronów najmocniej (spośród pozostałych) reaguje na pewne wzorce uczące.

Proszę dla każdego takiego pseudoneuronu w każdym cyklu uczenia (obejmującego prezentację wszystkich 150 wzorców uczących) stworzyć 3-elementową tabelę pomocniczą, w której zliczane będą te „zwycięstwa”. Ze względu na to, iż znamy klasę wzorca, jaki prezentujemy sieci SOM w trakcie uczenia nienadzorowanego, możemy w tych tablicach zliczać ilości wzorców poszczególnych klas, jakie dany pseudoneuron reprezentuje w sieci SOM.
Pseudoneuronów jest znacznie mniej niż wzorców uczących, każdy taki pseudoneuron powinien średnio reprezentować pewną ilość tych wzorców, które potencjalnie mogą należeć do różnych klas. W tych pomocniczych tabelach dla każdego neuronu zliczamy właśnie te ilości, np. neuron X zwyciężył 7x dla wzorców Setosa, 2x dla wzorców Versicolor i 1x dla wzorców Virginica, więc nasza tablica składa się 3 wartości (liczników zwycięstw dla poszczególnych klas): [7, 2, 1], a więc w sumie 10x. W celu wizualizacji graficznej tego neuronu w siatce neuronów wyznaczamy jego składowe koloru następująco:
R = 255 * (7 / 10), G = 255 * (2 / 10), B = 255 * (1 / 10),
uzyskując kolor RGB, którym wypełniamy jego miejsce w siatce.