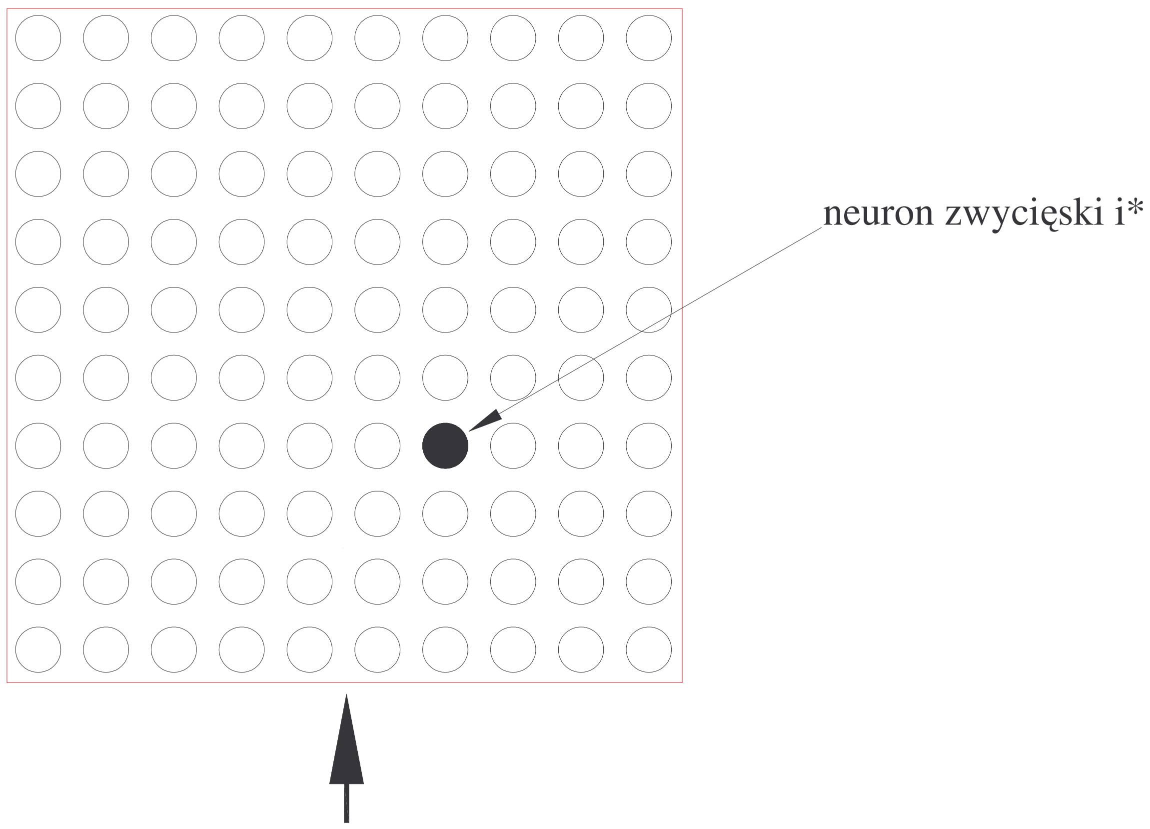
# Literatura

1. Tadeusiewicz R.: **Sieci neuronowe**. Akademicka Oficyna Wydawnicza, Warszawa 1993
2. Osowski S.: **Sieci neuronowe w ujęciu algorytmicznym**. WNT, Warszawa 1996
3. Hertz J., Krogh A., Palmer R. G.: **Wstęp do teorii obliczeń neuronowych**. WNT, Warszawa 1993
4. Masters T.: **Sieci neuronowe w praktyce. Programowanie w C++**. WNT, Warszawa 1996
5. Piłot T.,Gwiazda A.: **Zastosowania sztucznej inteligencji w inżynierii produkcji**. Pod Red. R. Knosali, WNT 2002

Sieci neuronowe

# Sieć Kohonena

# Schemat wyjściowej warstwy sieci o algorytmie uczenia o konkurencji ostrej




[wektor cech wejściowych $x_j$]

# Algorytm uczenia Kohonena

Metoda uczenia konkurencyjnego

Modyfikacja wag połączeń sieci:

$$w_{i*j} = w_{i*j} + \eta(t)\,(x_j^{\mu} - w_{i*j}),$$

gdzie: $\eta(t)$ – współczynnik uczenia, t – numer iteracji, $x_j^{\mu}$ – wartość j-tej cechy $\mu$-tego wzorca wejściowego, $w_{ij}$ – wartość wagowa połączenia wejściowego węzła j z i-tym neuronem wyjściowym.

Wartość wyjściowa:

$$h_i^{\mu} = \sum_j w_{ij}\, x_j^{\mu} = \mathbf{w}_i \cdot \mathbf{x}^{\mu}$$

Dla miary Euklidesowej:

$$|\mathbf{w}_{i*} - \mathbf{x}^{\mu}| \leq \mathbf{w}_i \cdot \mathbf{x}^{\mu}$$

Dla innych miar:

$$d\,(\mathbf{x}^{\mu},\mathbf{w}_{i*}) = \min d(\mathbf{x}, \mathbf{w}_i), \quad 0 \leq i \leq n$$

# Algorytm uczenia Kohonena

Metoda uczenia konkurencyjnego

W przypadku odległości stosuje się różne miary:

- miara Euklidesa:

$$d(\mathbf{x}, \mathbf{w}_i) = \sqrt{\sum_{j=1}^{N} \left(x_j - w_{ij}\right)^2}$$

- iloczyn skalarny:

$$d(\mathbf{x}, \mathbf{w}_i) = \sqrt{\sum_{j=1}^{N} \left(x_j - w_{ij}\right)^2}$$

- miara według normy $L_1$ (Manhattan):

$$d(\mathbf{x}, \mathbf{w}_i) = \sqrt{\sum_{j=1}^{N} \left(x_j - w_{ij}\right)^2}$$

- miara według normy $L_\infty$:

$$d(\mathbf{x}, \mathbf{w}_i) = \max_{j}\left(x_j - w_{ij}\right)$$

# Algorytm uczenia Kohonena

Normalizacja wektora wejściowego i wagowego

Sposoby normalizacji można przedstawić następująco:

• redefinicja składowych wektora wejściowego:

$$x_j \leftarrow \frac{x_j}{\sqrt{\sum_{j=1}^{N} x_j^2}}$$

• zwiększenie wymiaru przestrzeni wejściowej o jeden,

tzn. z $R^N \to R^{N+1}$, gdzie:

$$\sum_{j=1}^{N+1} x_j^2 = 1$$

# Algorytm uczenia Kohonena

Metoda uczenia konkurencyjnego z funkcją sąsiedztwa

Modyfikacja wag połączeń sieci:

$$\mathrm{w}_{\mathrm{ij}}(t) = \mathrm{w}_{\mathrm{ij}}(t-1) + \eta(t)\Lambda\left(i, i^{*}\right)\left(\mathrm{x}_{\mathrm{j}} - \mathrm{w}_{\mathrm{ij}}(t-1)\right)$$

Określenie węzła o maksymalnej wartości wyjściowej, gdy $|\mathbf{w}_i| = 1$ :

$$|\mathbf{w}_{i*} - \mathbf{x}^{\mu}| \leq \mathbf{w}_{i} \cdot \mathbf{x}^{\mu}$$

Obliczenie wartości funkcji w otoczeniu węzła „zwycięskiego”:

$$\Lambda\left(i, i^{*}\right) = \exp\left(-\frac{\mathrm{d}^{2}\left(i, i^{*}\right)}{2\lambda(t)^{2}}\right)$$

Obliczenie nowych wartości funkcji promienia sąsiedztwa $\lambda(t)$ i współczynnika uczenia $\eta(t)$.

# Algorytm uczenia Kohonena

## Funkcje sąsiedztwa – funkcje promienia sąsiedztwa

Jako funkcje promienia sąsiedztwa $\lambda(t)$ stosuje się zwykle następujące:

• funkcję liniową $\lambda_1(t)$:

$$\lambda_1(t) = \lambda_0 \left( 1 - \frac{t}{T_1} \right) + 1$$

gdzie: $T_1$ określa maksymalny okres,
t określa numer iteracji,
$\lambda_0$ wartość początkowa promienia sąsiedztwa.

• funckję wykładniczą:

$$\lambda_2(t) = \lambda_0 \left( \frac{\lambda_{\min}}{\lambda_0} \right)^{\frac{t}{T_{\max}}}$$

•gdzie: $T_{max}$ - parametr określający maksymalną wartość iteracji,
$\lambda_{min}$ - wartość przejściowa promienia sąsiedztwa, przy którym funkcja ta przechodzi przez punkt o współrzędnych ( $T_{max}$, $\lambda_{min}$ ).

# Algorytm uczenia Kohonena

Funkcja celu Rittera i Schultena

$$E\{\mathrm{w}_{\mathrm{ij}}\} = \frac{1}{2}\sum_{ijk} \mathrm{M}_{\mathrm{i}}^{\mu}\Lambda(i,k)(\mathrm{x}_{\mathrm{j}} - \mathrm{w}_{\mathrm{ij}})^2 = \frac{1}{2}\sum_{\mu}\Lambda(i,i^*)\left|\mathbf{x}^{\mu} - \mathbf{w}_{i^*}\right|$$

Funkcje współczynnika uczenia:

• funkcja liniowa

$$\eta(t) = \eta_0\left(1 - \frac{t}{T_1}\right)$$

• funkcja potęgowa

$$\eta(t) = \eta_0 \alpha t^{-\alpha}$$

• funkcja wykładnicza

$$\eta_2(\mathrm{t}) = \eta_0\left(\frac{\eta_{\min}}{\eta_0}\right)^{\frac{t}{T_{\max}}}$$

# Algorytm „gazu neuronowego” cz.I

1. Sortowanie węzłów wyjściowych ze względu na odległość wzorca wejściowego, w mierze Euklidesa, od macierzy wag, związanej z danym węzłem:

$$d_0 < d_1 < ... < d_{n-1}$$

gdzie:

n – liczba neuronów wyjściowych,

m=0,1,...,n-1 – określa pozycję węzła i-tego w wektorze odległości,

$d_m$ – oznacza odległość na m-tej pozycji i-tego węzła wyjściowego w uporządkowanym wektorze odległości, która wyraża się zależnością:

$$\mathrm{d_m} = \left| \mathbf{x} - \mathbf{w}_i \right|$$

gdzie:

**x -** macierz wzorca wejściowego,

$\mathbf{w}_\mathrm{i}$ **-** macierz wag połączeń i-tego węzła wyjściowego i macierzy wejściowej **x**.

# Algorytm „gazu neuronowego” cz.II

2. Określenie wartości funkcji sąsiedztwa na węzłach uporządkowanych wg odległości $d_m$:

$$\Lambda\left(i, i^*\right) = e^{\frac{\lambda(t)}{d_m^2}}$$

gdzie λ(t) jest funkcją promienia sąsiedztwa.

Funkcje celu stowarzyszona z funkcją sąsiedztwa charakteryzuje się, przy wartościach $\lambda \to 0$, przejściem w algorytm uczenia konkurencji ostrej. Postać funkcji przybiera kształt paraboli. W przypadku jednak, gdy $\lambda \to \infty$, wtedy przy wolnej zmianie λ dostajemy funkcję wielomodalną, o wielu minimach lokalnych. W tym momencie funkcja ta zapewnia dojście procesu uczenia do globalnego minimum. Wskutek tego algorytm ten jest określany jako najbardziej skuteczny wśród takich algorytmów jak mapa Kohonena czy k-uśrednień.