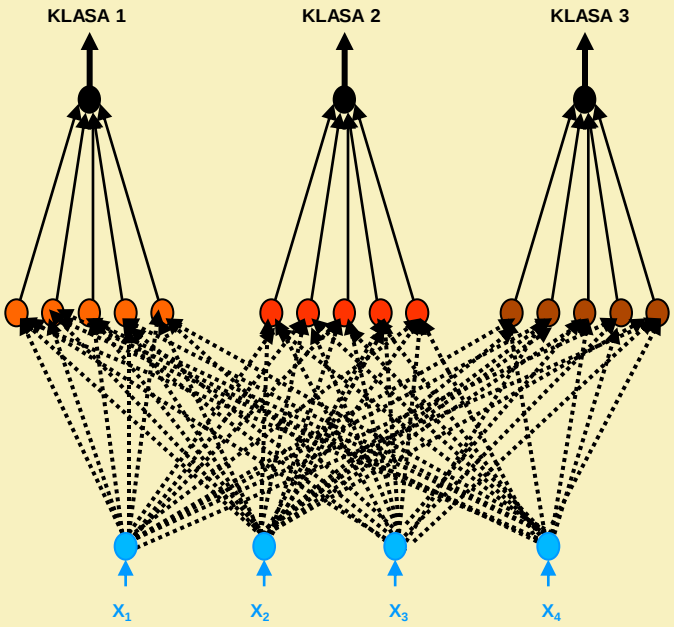
SIECI LVQ
Learning Vector Quantization

SIECI CP
CounterPropagation

Joanna Grabska- Chrząstowska

Learning Vector Quantization

SIECI LVQ



Joanna Grabska- Chrząstowska

REGUŁA WIDROW-HOFFA (DELTA)

$$w_{k\,i}^{(j+1)} = w_{k\,i}^{(j)} + \eta\ \delta\ x_i^{(j)}$$

$\delta = z - y$

METODA SAMOUCZENIA HEBBA

$$w_{k\,i}^{(j+1)} = w_{k\,i}^{(j)} + \eta\ y_k^{(j)}\ x_i^{(j)}$$

UCZENIE Z FORSOWANIEM

$$w_{k\,i}^{(j+1)} = w_{k\,i}^{(j)} + \eta\, z_k^{(j)}\, x_i^{(j)}$$

maksymalna liczba możliwych do zapamiętania wzorców w sieci o k neuronach:

$$N_{\max} \cong \frac{k}{2 \log k}$$

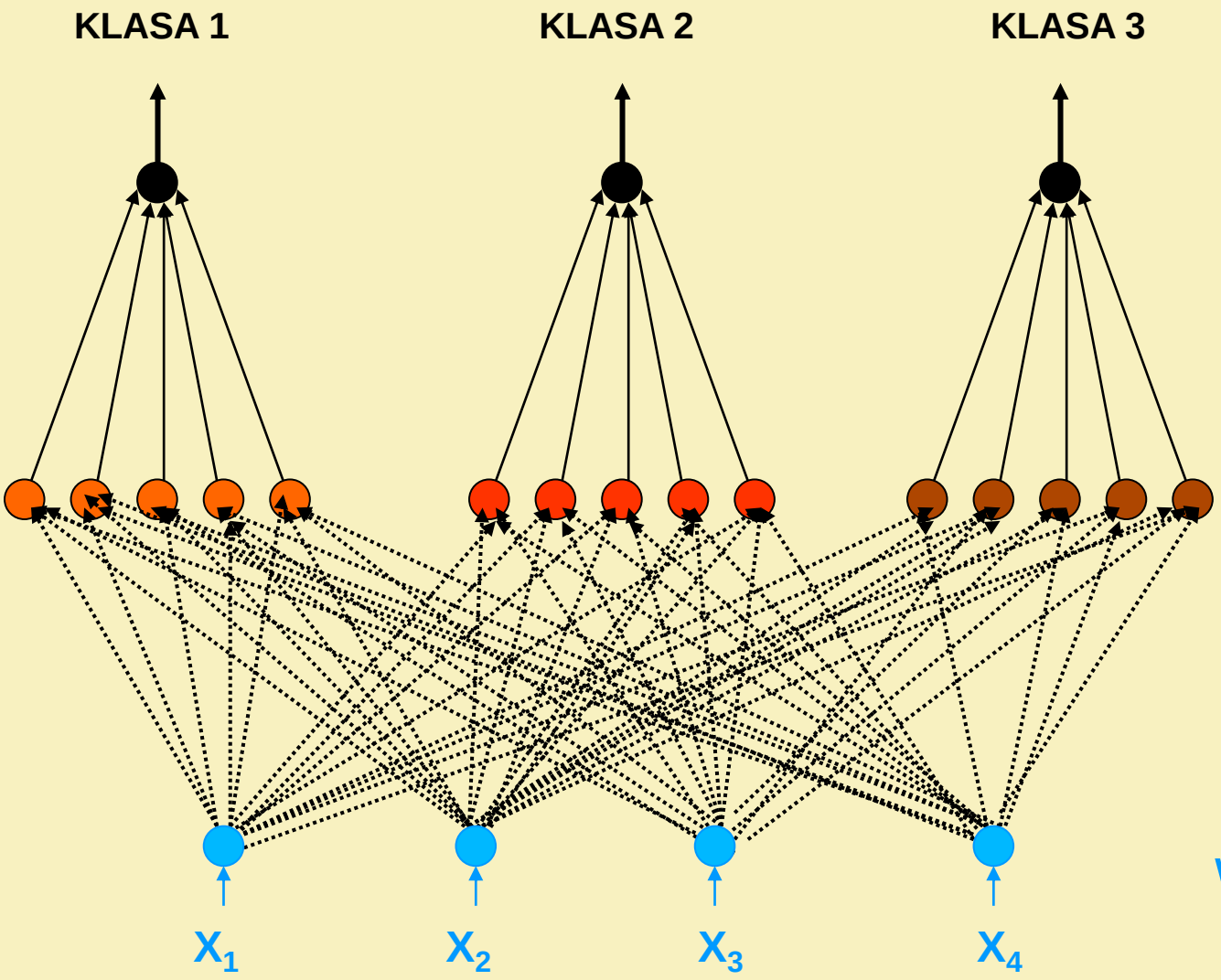
Efekty uczenia:

Przy założeniu, że wszystkie wektory wejściowe są ortonormalne ,

sieć:

- uczy się wiernie odtwarzać wymagane reakcje na wszystkie rozważane sygnały wejściowe (pamięć);
- potrafi uśredniać wejściowe sygnały i odtwarzać idealny wzorzec w serii przypadkowo zniekształconych obserwacji.

ARCHITEKTURA SIECI LVQ



Sieć LVQ (wprowadzona przez Kohonena) służy do klasyfikowania sygnały wejściowych i jest przykładem ***uczenia z forsowaniem***. Warstwa wyjściowa przypisuje wektory wyjściowe do jednej z kilku klas. Główna częścią sieci jest WARSTWA KOHONENA, która ucząc się dokonuje klasyfikacji.

LVQ dostarcza jednakową liczbę neuronów przypisanych do danej klasy. Podklasy w danej grupie nie muszą być podobne.

- Podczas uczenia obliczana jest **odległość** wektora wejściowego od wszystkich neuronów warstwy i wyłaniany jest **zwycięzca** jako ten leżący najbliżej.

- Jeżeli **wygrywający** neuron **należy** do klasy sygnału, który pojawił się na wejściu to jego wagi są modyfikowane tak aby zbliżyć się do prezentowanego sygnału.

- Jeżeli **wygrywający** neuron **nie należy** do tej klasy co sygnał wejściowy to jest od tego wektora odsuwany co określa się jako *odpychanie*.

- Podczas procesu uczenia **neuron** przypisany do danej klasy **wędruje** do obszaru związanego z tą kategorią.

- W trybie **testowania** (klasyfikacji) obliczana jest odległość prezentowanego wektora wejściowego do każdego neuronu i leżący najbliżej zostaje zwycięzca. Przynależność do klasy tego sygnału wskazuje ten zwycięski neuron.

W podstawowej wersji sieci LVQ obliczana jest odległość między wektorem wejściowym a wekto-rem wag i - tego neuronu dla każdego i = 1, ..., m

$$d_i = \| w_i - x \| = \sqrt{\sum_{j=1}^{N} (w_{ij} - x_j)^2}$$

Wagi zwycięskiego neuronu sa modyfikowane zgodnie z wzorem:

$$W' = \begin{cases} w + \alpha \ (x - w) & \text{jeśli neuron należy do właściwej klasy} \\ w - \gamma \ (x - w) & \text{jeśli neuron NIE należy do właściwej klasy} \end{cases}$$

WARIANTY LVQ - LVQ 1

Wprowadzenie pojęcia **„sumienia”**. Jeżeli neuron wygrywa zbyt często „oddaje’ zwycięstwo innemu neuronowi. Realizuje się to poprzez wprowadzenie odległości ***bias***.

$$d_i' = d_i + b_i$$

Wybiera się **globalnego** zwycięzcę obliczając odległość d_i i **lokalnego** zwycięzcę ale **z tej klasy** biorąc pod uwagę d_i'. Wagi lokalnego zwycięzcy są modyfikowane w następujący sposób:

$$w' = \begin{cases} w + \alpha(x - w) & \text{jeżeli lokalny zwycięzca jest równocześnie zwycięzcą globalnym} \\ w + \beta(x - w) & \text{jeżeli lokalny zwycięzca nie jest równocześnie zwycięzcą globalnym} \end{cases}$$

podczas gdy globalny zwycięzca jest odrzucany od wektora wejściowego zgodnie z wzorem:

$$w' = w - \gamma(x - w)$$ **jeżeli globalny zwycięzca nie jest we właściwej klasie**

WARIANTY LVQ - LVQ 2

Jeżeli w sieci jest neuron zwycięzca z wektorem wag w_1, który nie wskazuje na klasę sygnału wejściowego a drugi następny w kolejności o wagach w_2 właśnie z tej klasy to w takiej wersji sieci LVQ zwycięzca jest odpychany od sygnału wejściowego natomiast ten drugi neuron jest traktowany jak zwycięzca pod warunkiem, że odległość wektora wejściowego od obu wybranych neuronów jest podobna.

$w_1 = w_1 - \alpha \times (x - w_1)$

$w_2 = w_2 + \alpha \times (x - w_2)$

STRATEGIA UCZENIA SIECI LVQ

W typowych zastosowaniach powinno zacząć się od:

wariantu **LVQ 1**,

następnie przejść do

wersji podstawowej **LVQ** lub

LVQ bez odpychania,

a na koniec użyć

wariantu **LVQ 2**.

Basic LVQ learning proceeds by computing the Euclidean distances, *di*, between a training vector, *x*, and each PE's weight vector, *wi*, according to the standard formula:

$$di = ||wi - x|| = \left\{ \sum_{j=1}^{N} (w_{ij} - x_j)^2 \right\}^{1/2}$$

The winning PE's weight vector is adjusted according to the following

$$w' = \begin{cases} w + \alpha (x - w) & \text{if the winning PE is in the correct class} \\ w - \gamma (x - w) & \text{if the winning PE is **not** in the correct class} \end{cases}$$

LVQ1 (that is, LVQ with conscience) proceeds by adding a bias, *bi*, to the distance, *di*, to the PEs in the correct class, that is, the class of the training vector.

$$di' = di + bi$$

LVQ1 then determines both a global winner using the unbiased distances and an in-class winner using the biased distances.

The in-class winner is then moved toward the training vector according to the following:

$$w' = \begin{cases} w + \alpha\,(x - w) & \text{if the in–class winner is also the global winner} \\ w + \beta\,(x - w) & \text{if the in–class winner is not the global winner} \end{cases}$$

while the global winner is moved away from the training vector according to the following:

$$w' = w - \gamma\,(x - w) \qquad \text{if the global winner is not in the correct class}$$

The bias distance, *bi*, is calculated by the following:

$$bi = \mu \times di_{max} \times (1 - Npi)$$

where

di_{max} is a maximum distance which is internally estimated

μ is a constant which decreases as training progresses

pi estimates the PE win frequency; is initialized at 1/N

N is the number of Kohonen PEs per class

φ is the constant for updating the in-class win frequencies. This is done according to the formula

$$pi = \begin{cases} (1 - \varphi)\, pi & \text{if } i \text{ is not the in-class winner} \\ (1 - \varphi)\, pi + \varphi & \text{if } i \text{ is the in-class winner} \end{cases}$$

LVQ2 has a winning PE, $w1$, which is not in the class of the training vector and a second place PE, $w2$, which is in the class of the training vector.

The winning PE is moved away from the training vector and the second place PE is moved toward the training vector according to the following:

$$w1 = w1 - \alpha \times (x - w1)$$

$$w2 = w2 + \alpha \times (x - w2)$$

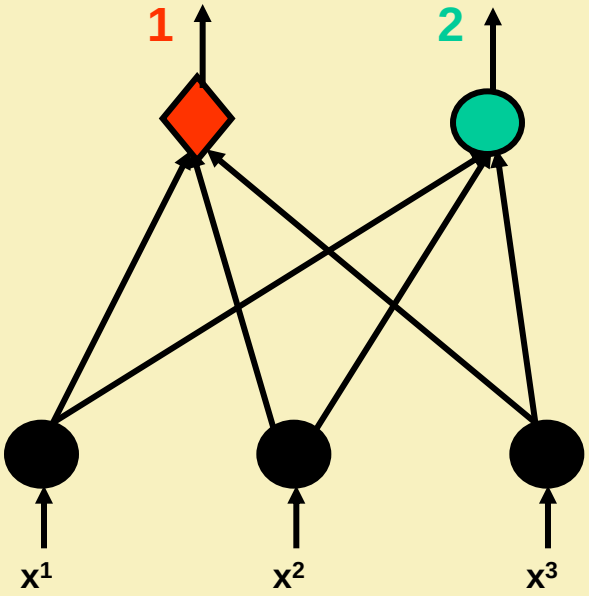
if x is near to $\dfrac{(w1 + w2)}{2}$

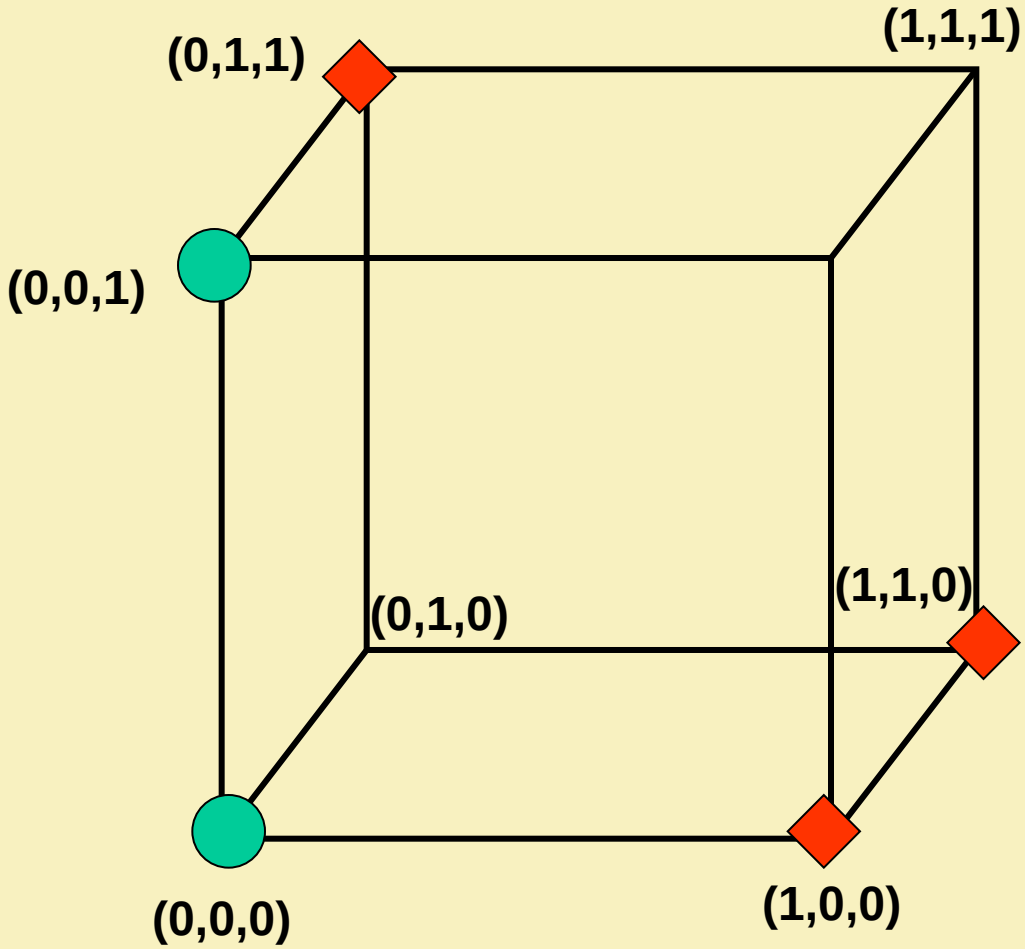
Nearness in this case requires that x lie between the two planes perpendicular to the line joining $w1$ to $w2$ and passing through the points:

$$\frac{(w1 + w2)}{2} + \omega \times (w2 - w1)$$

$$\frac{(w1 + w2)}{2} - \omega \times (w2 - w1)$$

PRZYKŁAD ZASTOSOWANIA SIECI LVQ





x_1 (1 , 1 , 0) klasa 1

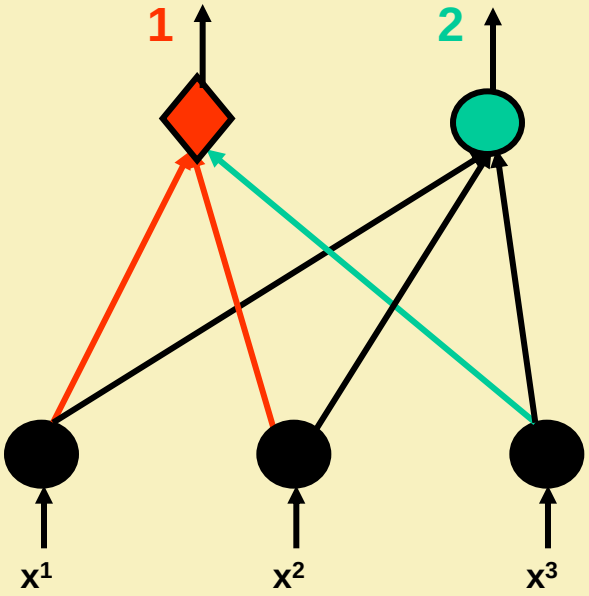
x_2 (0 , 0 , 0) klasa 2

x_3 (0 , 0 , 1) klasa 2

x_4 (1 , 0 , 0) klasa 1

x_5 (0 , 1 , 1) klasa 1

PRZYKŁAD ZASTOSOWANIA SIECI LVQ



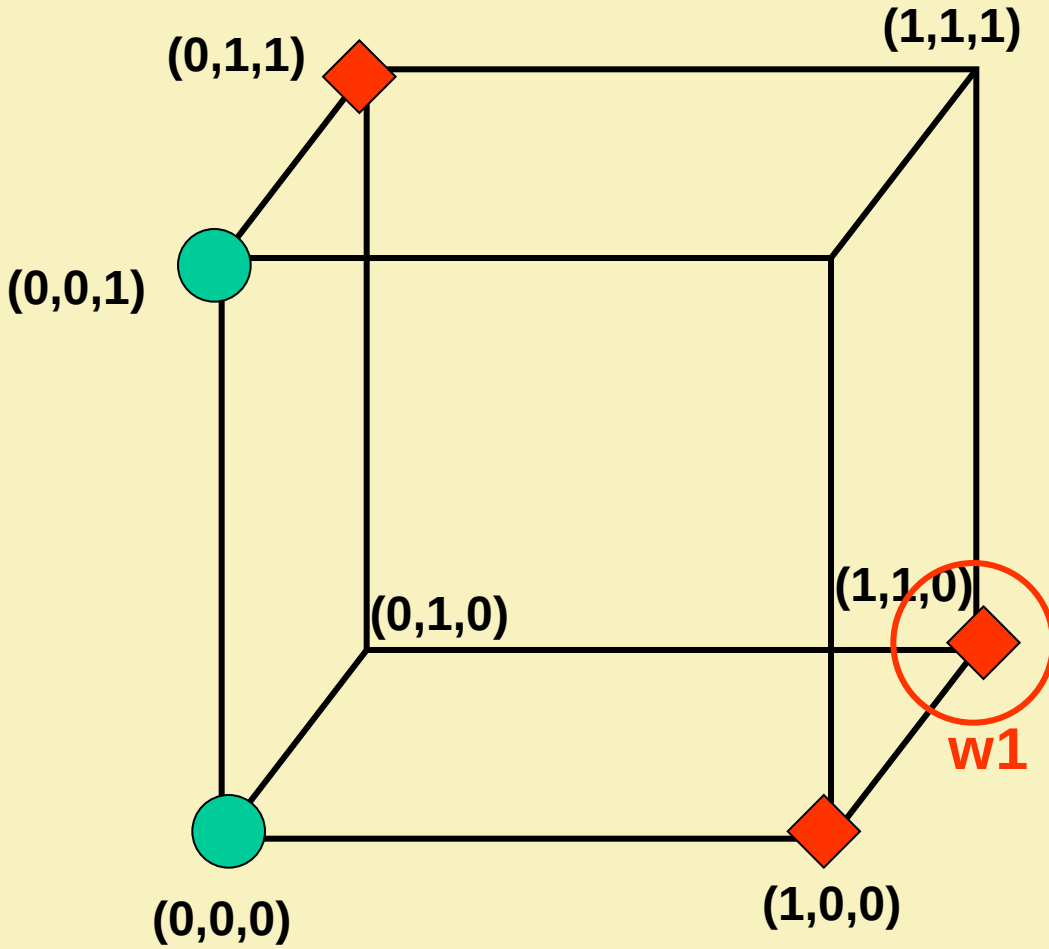
x_1 (1 , 1 , 0) klasa 1

x_2 (0 , 0 , 0) klasa 2

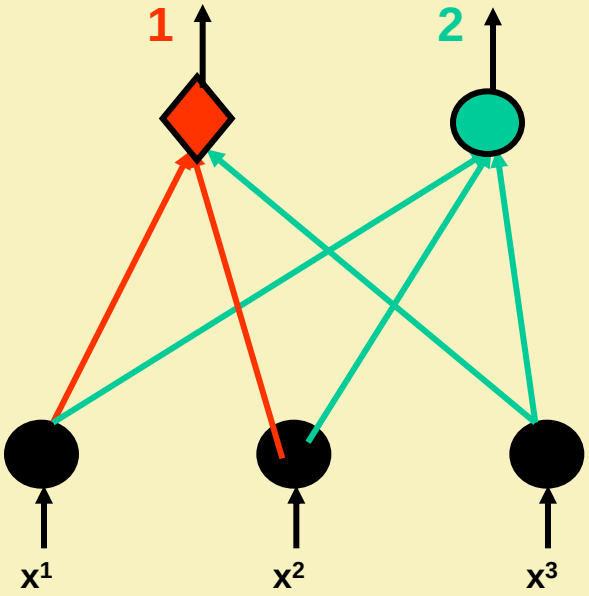
x_3 (0 , 0 , 1) klasa 2

x_4 (1 , 0 , 0) klasa 1

x_5 (0 , 1 , 1) klasa 1



PRZYKŁAD ZASTOSOWANIA SIECI LVQ



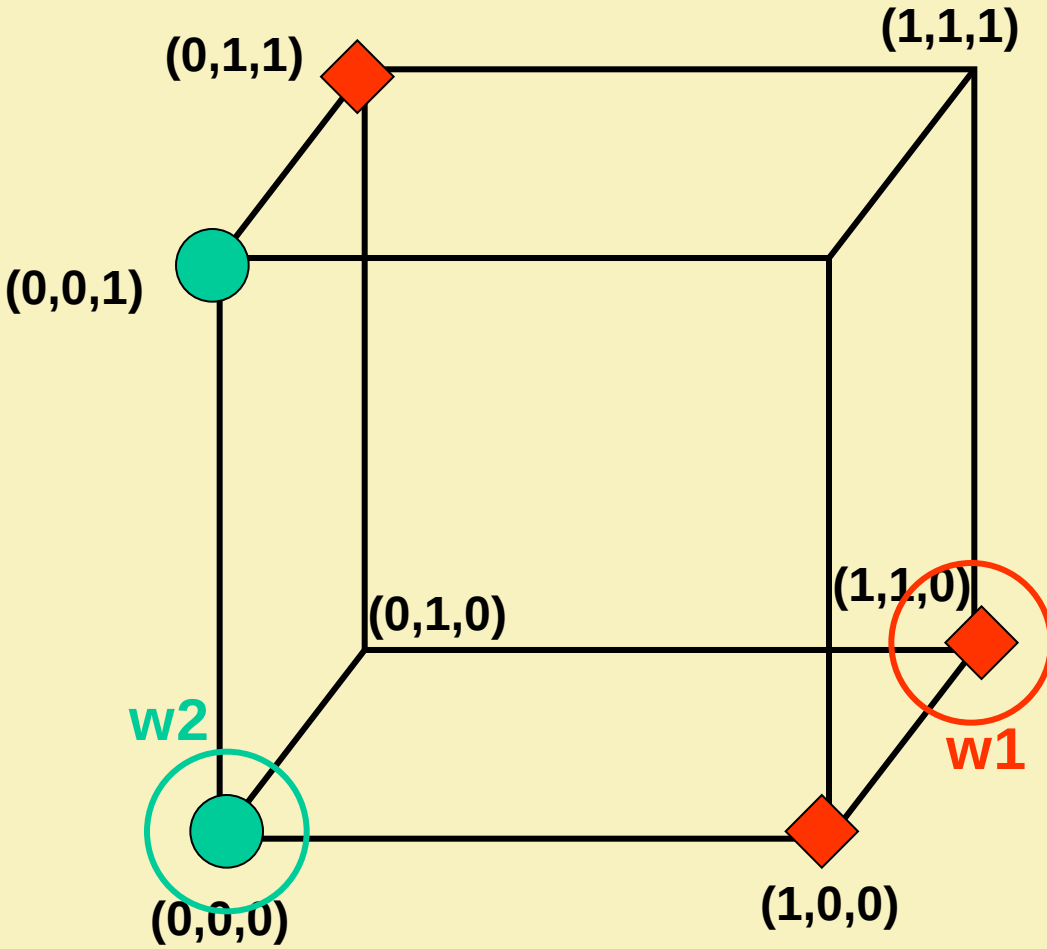
x_1 (1 , 1 , 0) klasa 1

x_2 (0 , 0 , 0) klasa 2

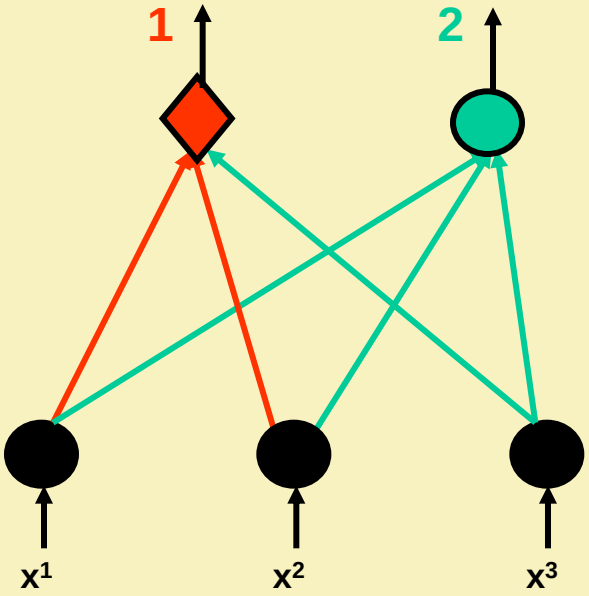
x_3 (0 , 0 , 1) klasa 2

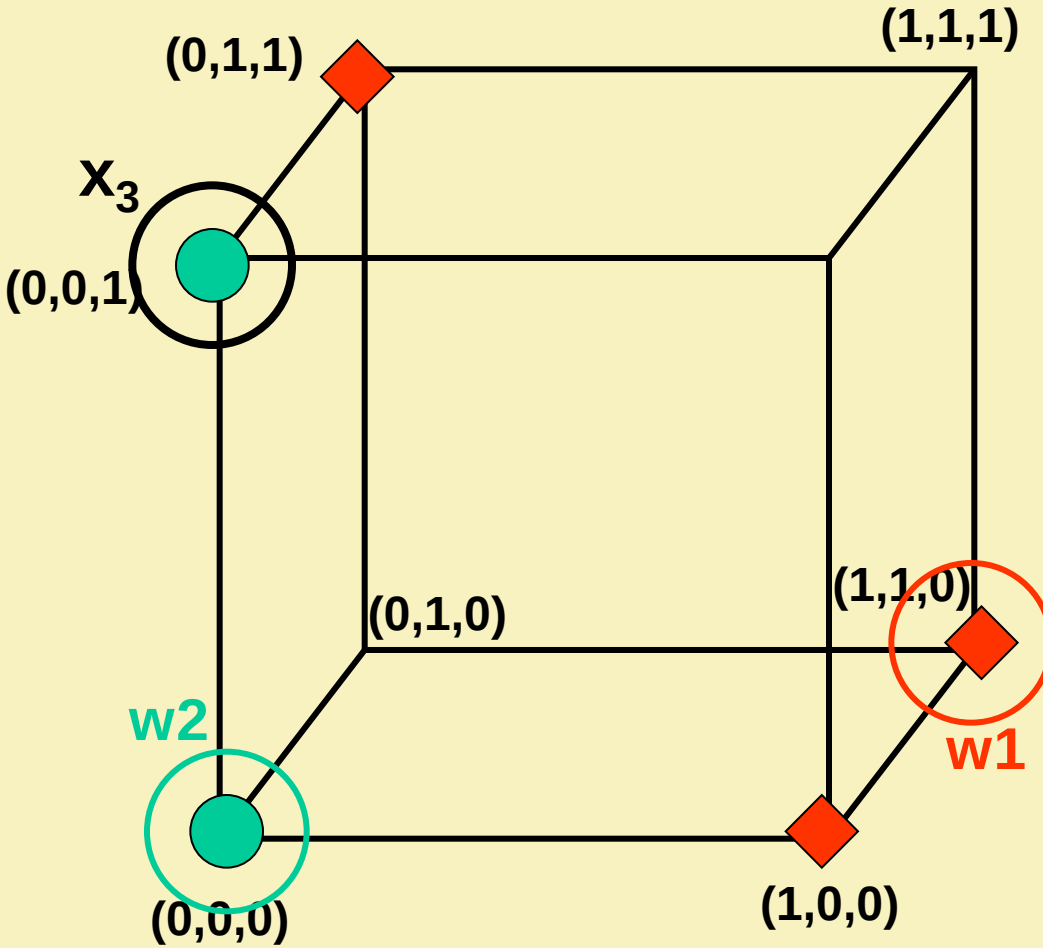
x_4 (1 , 0 , 0) klasa 1

x_5 (0 , 1 , 1) klasa 1



PRZYKŁAD ZASTOSOWANIA SIECI LVQ





x_1 (1 , 1 , 0) klasa 1

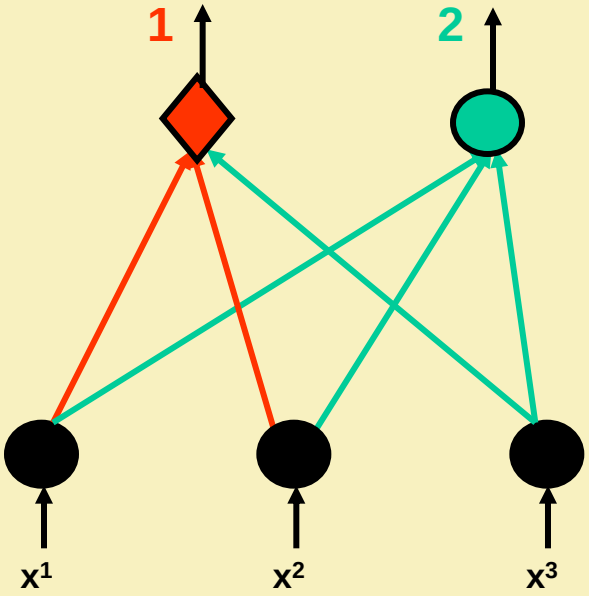
x_2 (0 , 0 , 0) klasa 2

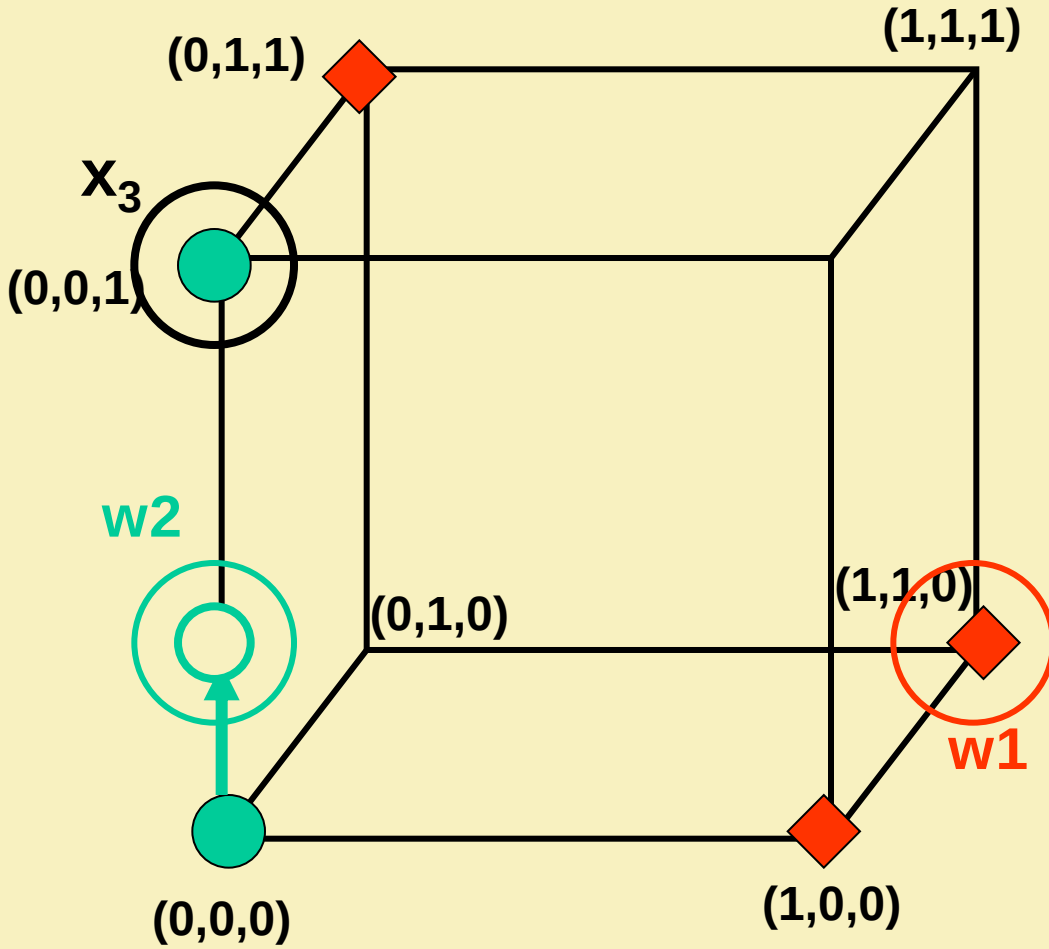
x_3 (0 , 0 , 1) klasa 2

x_4 (1 , 0 , 0) klasa 1

x_5 (0 , 1 , 1) klasa 1

PRZYKŁAD ZASTOSOWANIA SIECI LVQ





x_1 (1 , 1 , 0) klasa 1

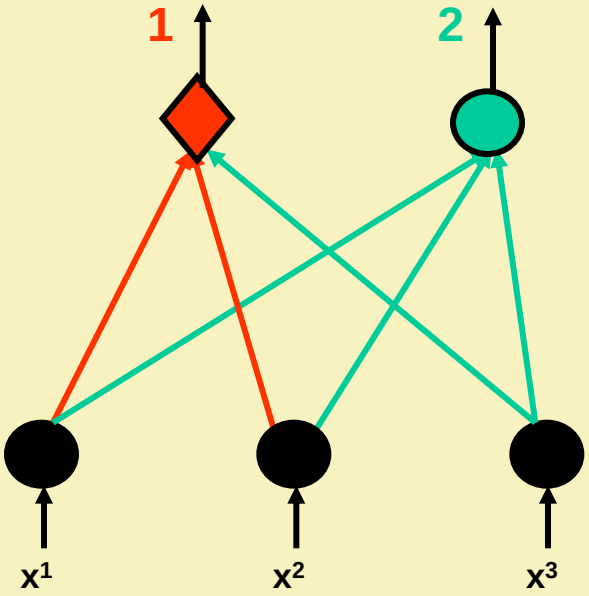
x_2 (0 , 0 , 0) klasa 2

x_3 (0 , 0 , 1) klasa 2

x_4 (1 , 0 , 0) klasa 1

x_5 (0 , 1 , 1) klasa 1

PRZYKŁAD ZASTOSOWANIA SIECI LVQ



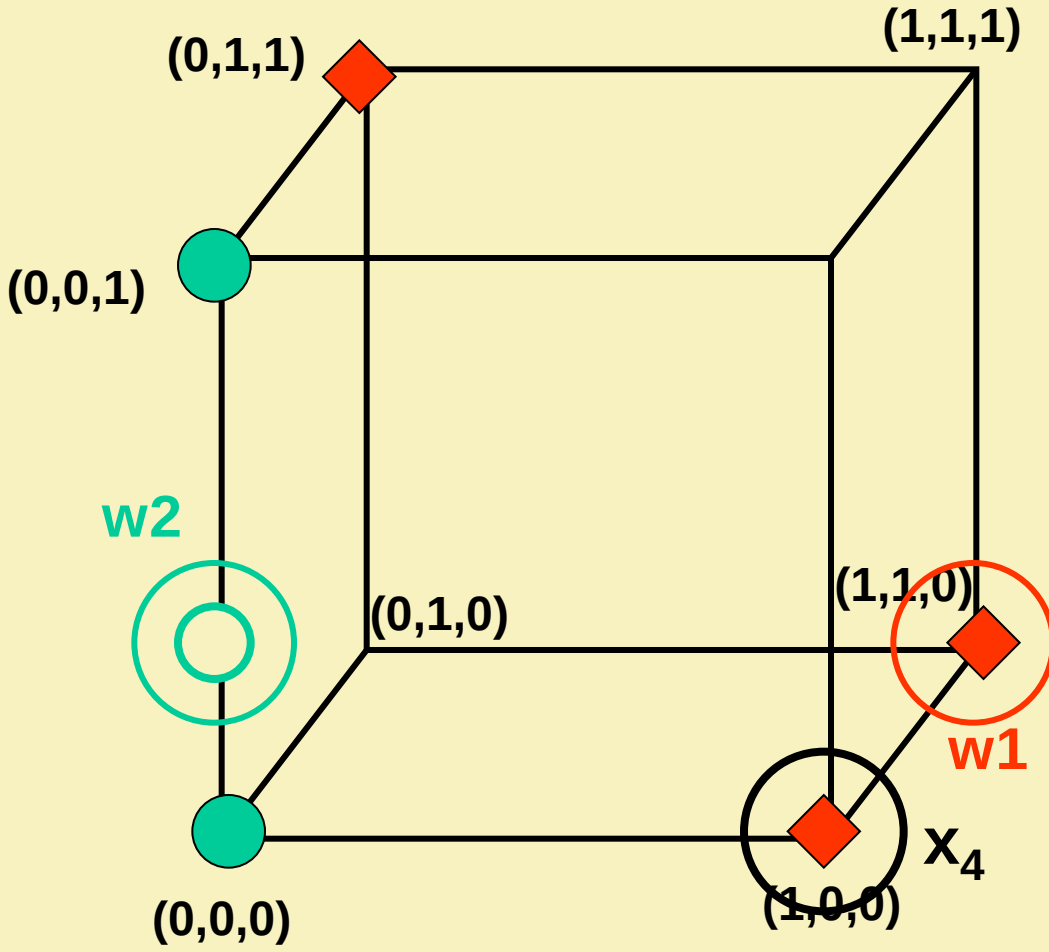
x_1 (1 , 1 , 0) klasa 1

x_2 (0 , 0 , 0) klasa 2

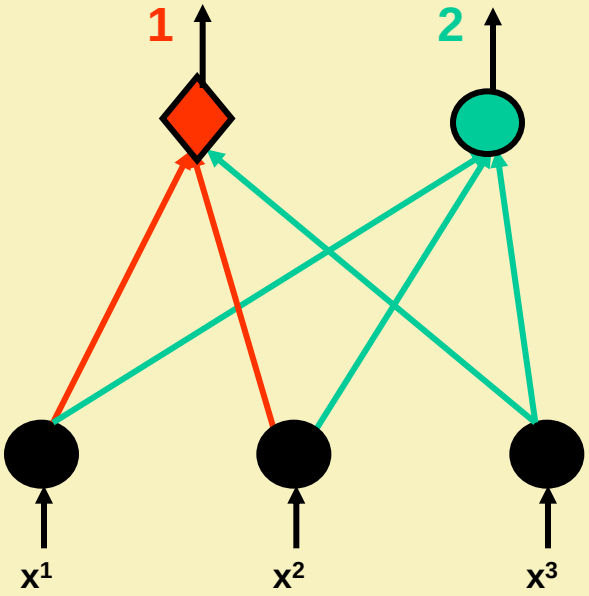
x_3 (0 , 0 , 1) klasa 2

x_4 (1 , 0 , 0) klasa 1

x_5 (0 , 1 , 1) klasa 1



PRZYKŁAD ZASTOSOWANIA SIECI LVQ



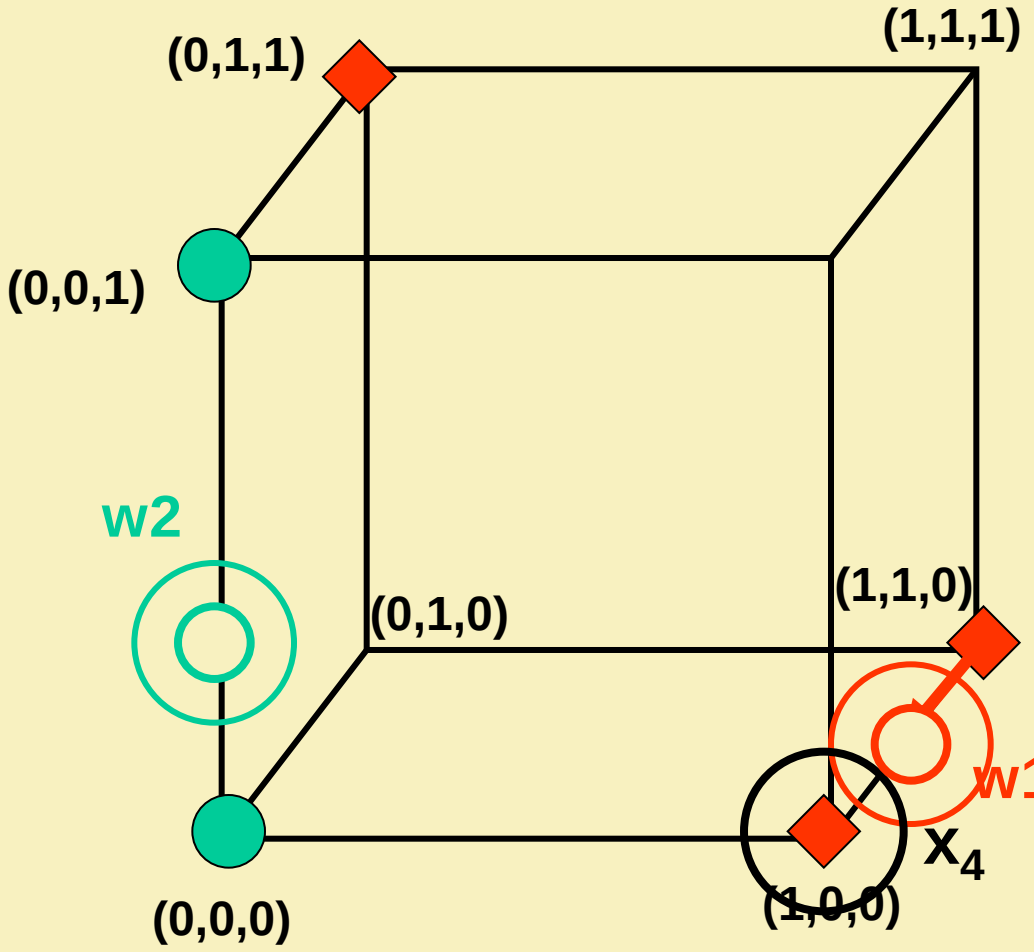
x_1 (1 , 1 , 0) klasa 1

x_2 (0 , 0 , 0) klasa 2

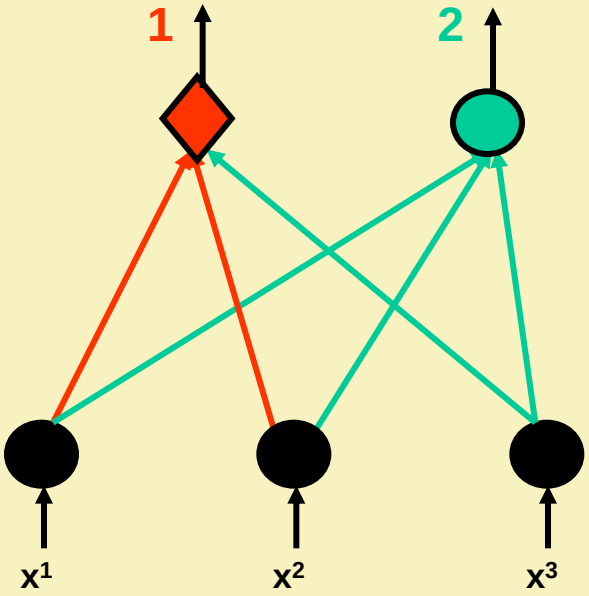
x_3 (0 , 0 , 1) klasa 2

x_4 (1 , 0 , 0) klasa 1

x_5 (0 , 1 , 1) klasa 1



PRZYKŁAD ZASTOSOWANIA SIECI LVQ



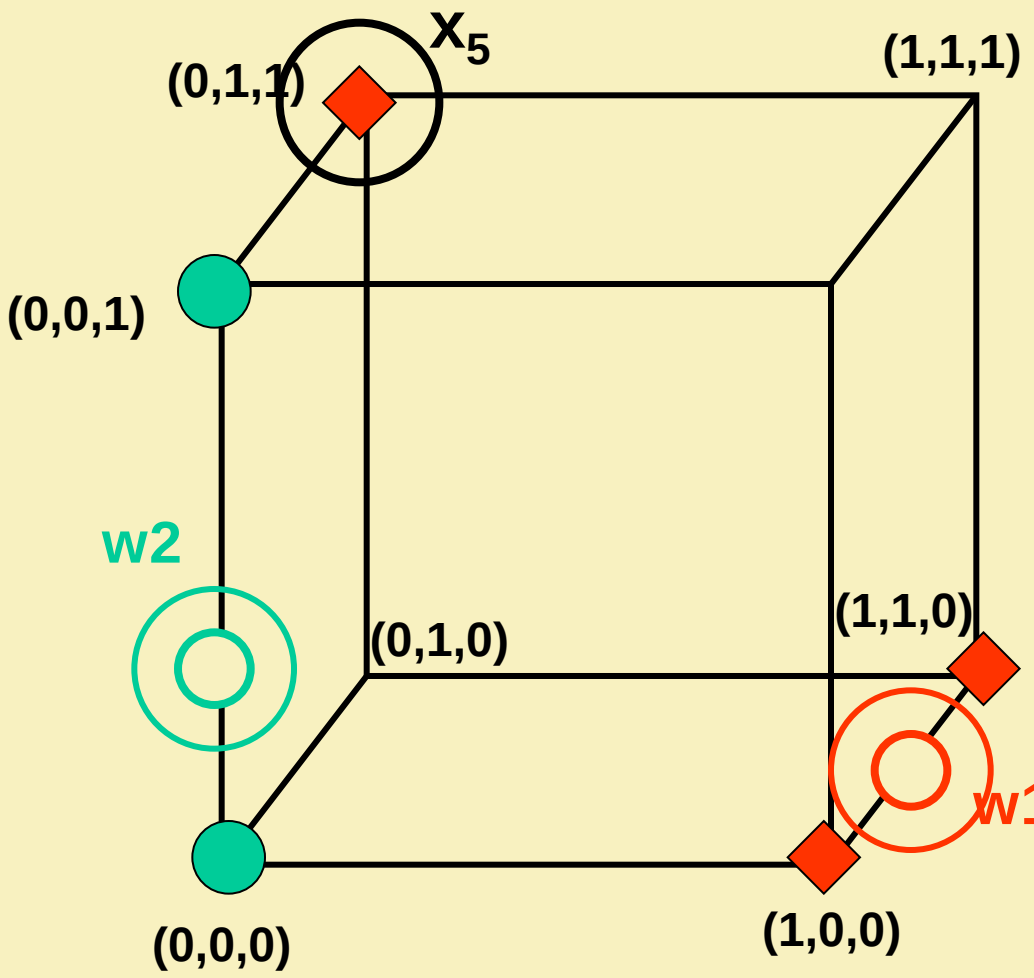
x_1 (1 , 1 , 0) klasa 1

x_2 (0 , 0 , 0) klasa 2

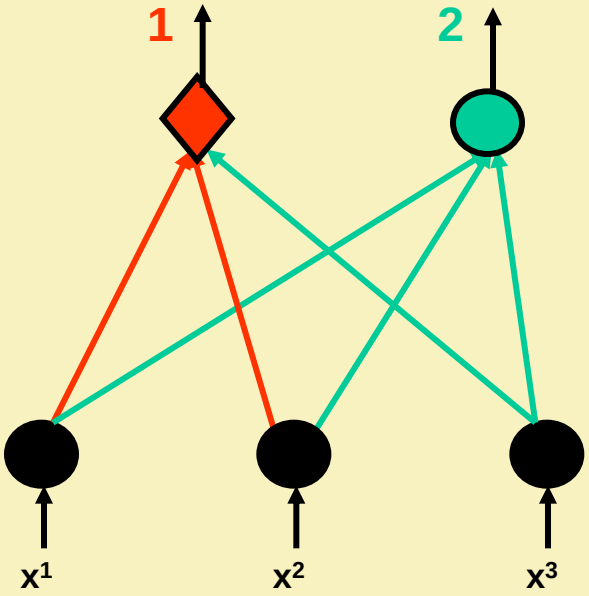
x_3 (0 , 0 , 1) klasa 2

x_4 (1 , 0 , 0) klasa 1

x_5 (0 , 1 , 1) klasa 1



PRZYKŁAD ZASTOSOWANIA SIECI LVQ



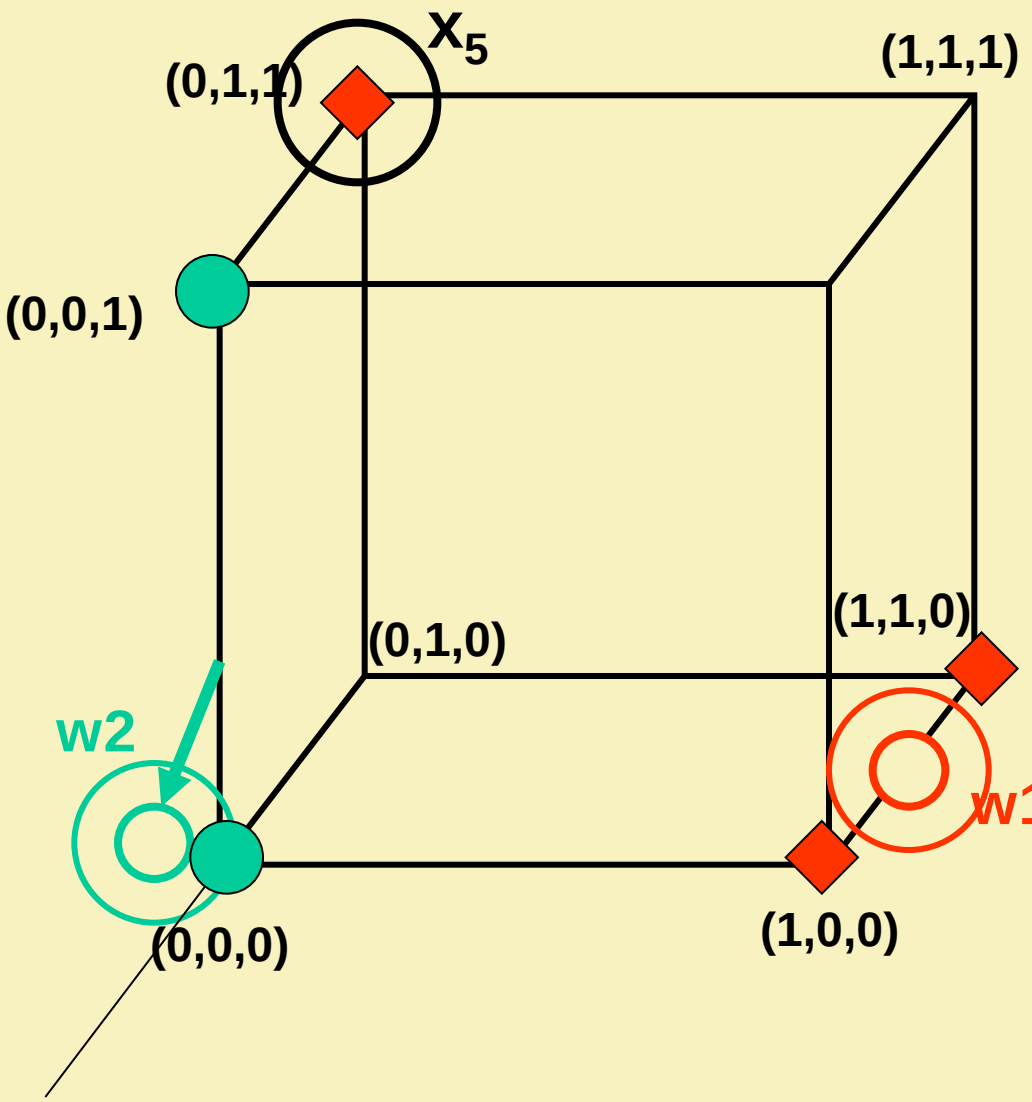
x_1 (1 , 1 , 0) klasa 1

x_2 (0 , 0 , 0) klasa 2

x_3 (0 , 0 , 1) klasa 2

x_4 (1 , 0 , 0) klasa 1

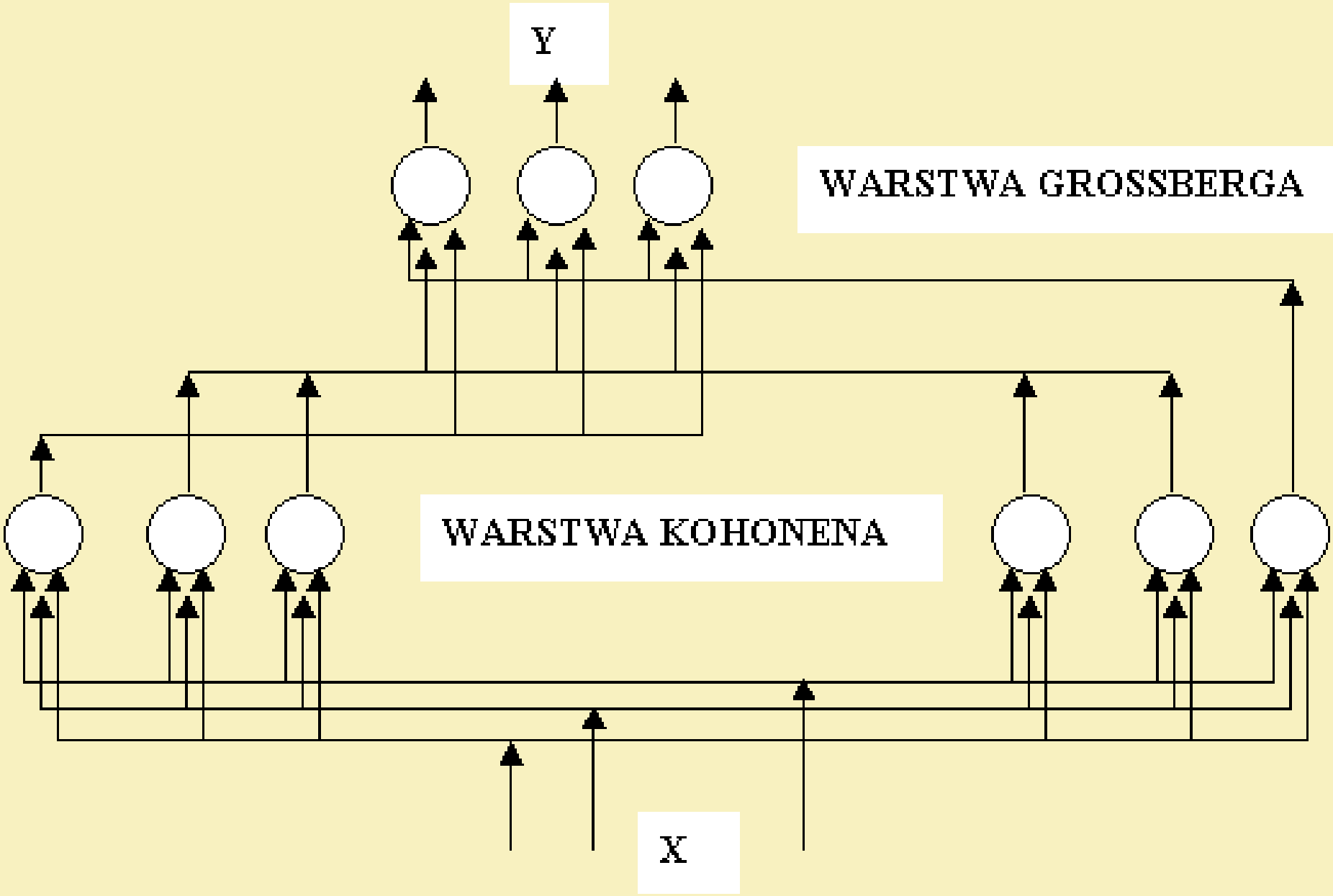
x_5 (0 , 1 , 1) klasa 1



SIECI CP
CounterPropagation

Joanna Grabska- Chrząstowska

ARCHITEKTURA SIECI CP



Sieci CounterPropagation (CP)

uczona z nauczycielem !!!

zaproponowane przez Roberta Hecht-Nielsens są kompilacją sieci Kohonena i sieci Grosberga. Wprowadzają nową jakość, czyli zwiększoną szybkość uczenia. Jest odpowiedzią na wady sieci ze wsteczną propagacją, w której uczenie jest powolne i pracochłonne.

Przy pomocy CP można szybko weryfikować hipotezy robocze.

DZIAŁANIE PIERWSZEJ WARSTWY

Założenie: **normalizacja wektorów wejściowych**

$$x'_i = \frac{x_i}{\sqrt{\sum_{j=1}^{n} x_j^2}} \quad \Rightarrow \quad \|x\| = 1$$

Pierwsza warstwa realizuje algorytm Kohonena:

$$e_j = W_j^T X$$

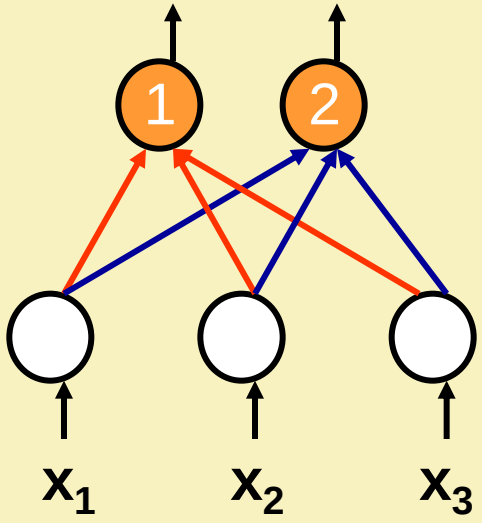
$$k_j = \begin{cases} 1 & \text{gdy } \forall_{i \neq j} \; e_j > e_i \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

GDY BRAK NORMALIZACJI

Przykład:

$$\mathbf{w}_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \qquad \mathbf{w}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$



$e_1 = 14$ $e_2 = 2$ ⇒ neuron nr 1 zostaje zwycięzcą

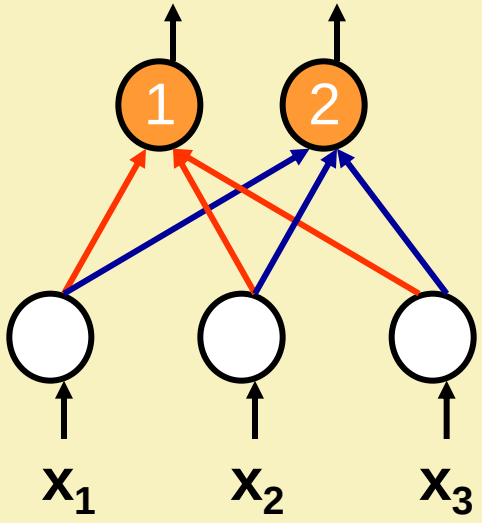
PRAWIDŁOWO

GDY BRAK NORMALIZACJI

Przykład:

$$\mathbf{w}_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \qquad \mathbf{w}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$



$\mathbf{e}_1 = 2$ $\mathbf{e}_2 = 1$ ⇒ neuron nr 1 zostaje zwycięzcą

BŁĄD

DZIAŁANIE DRUGIEJ WARSTWY

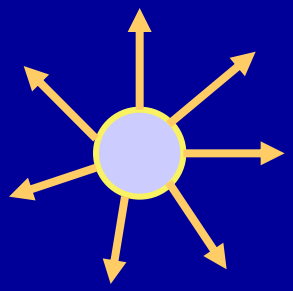
Druga warstwa realizuje algorytm Outstar Grossberga

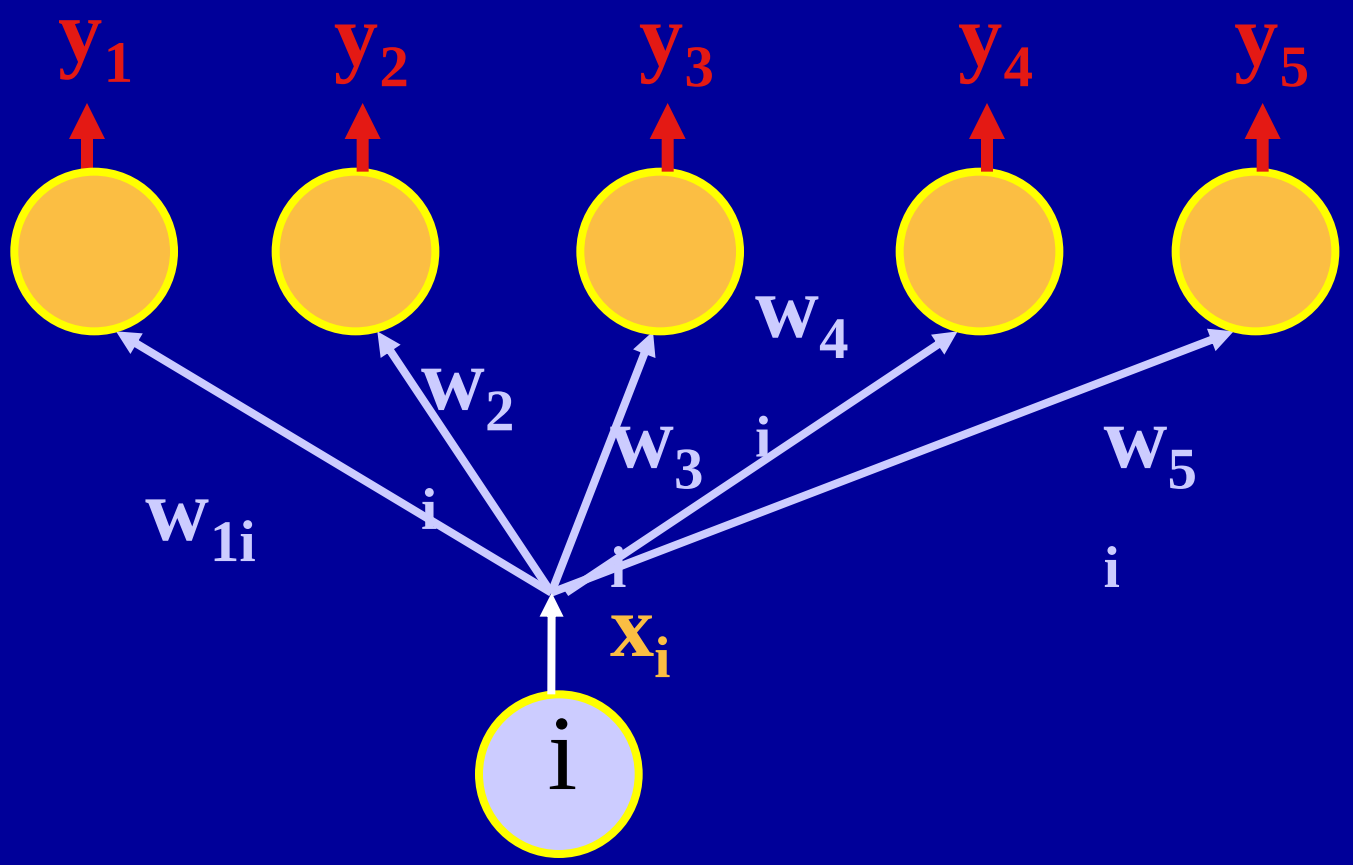
METODA OUTSTAR LEARNING
"GWIAZDY WYJŚĆ"

$$w_{ki}^{(j+1)} = w_{ki}^{(j)} + \eta^{(j)} \left[y_k^{(j)} - w_{ki}^{(j)} \right]$$

$$\eta^{(j)} = 0{,}1 - \lambda * j$$

i – ustalone
k - zmienne

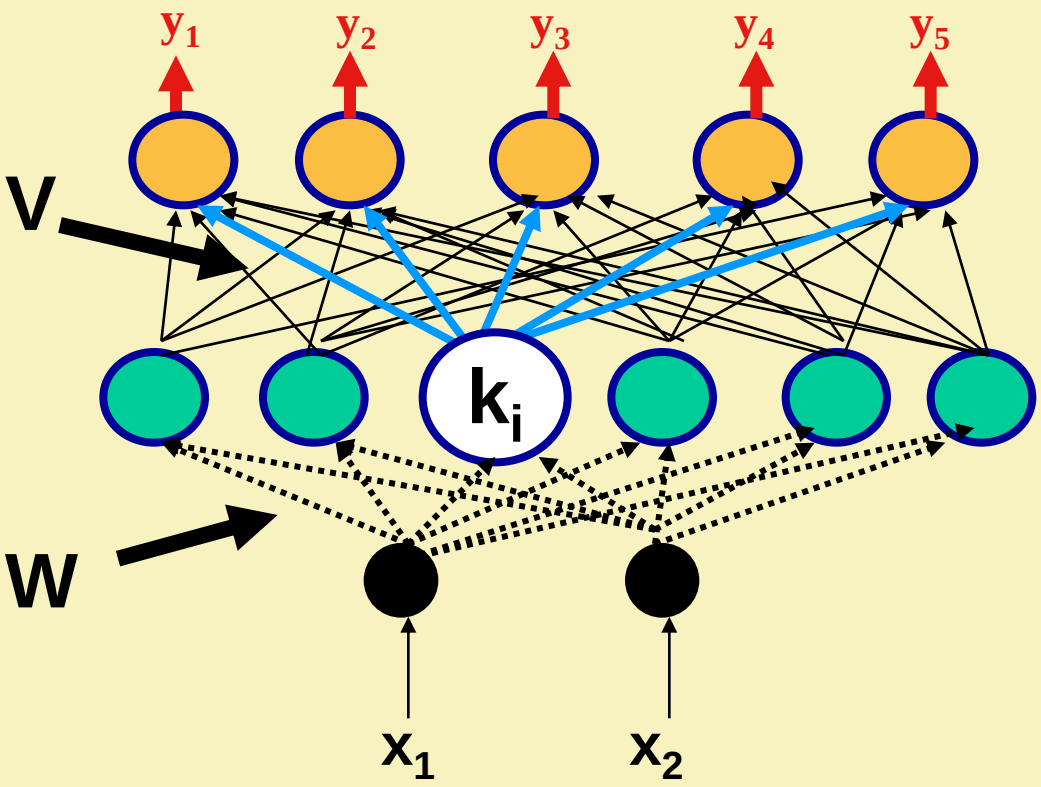




DZIAŁANIE DRUGIEJ WARSTWY

Druga warstwa realizuje algorytm Outstar Grossberga

$$Y = V\,K \quad \text{czyli} \quad y_s = \sum_{j=1}^{m} v_{sj}\, k_j$$



UCZENIE PIERWSZEJ WARSTWY

W danym kroku uczenia
korekcie wag podlega tylko zwycięzca

$$\Delta W = \eta_1 (X - W)$$

początkowo $w_{ij} = \sqrt{1/n}$

zamiast x podaje się na wejście x':

$$x_i^{(k)} ' = \eta_2(k)\, x_i^{(k)} + [1 - \eta_2 (k)] \sqrt{1/n}$$

UCZENIE PIERWSZEJ WARSTWY

W danym kroku uczenia
korekcie wag podlega tylko zwycięzca

$$\Delta W = \eta_1 (X - W)$$

początkowo $w_{ij} = \sqrt{1/n}$

bliskie 1

bliskie 0

zamiast x podaje się na wejście x':

$$x_i^{(k)} \,' = \eta_2(k)\, x_i^{(k)} + [1 - \eta_2 (k)] \sqrt{1/n}$$

dla małych $\eta_2(k)$ - małe

UCZENIE PIERWSZEJ WARSTWY

W danym kroku uczenia
korekcie wag podlega tylko zwycięzca

$$\Delta W = \eta_1 (X - W)$$

początkowo $w_{ij} = \sqrt{1/n}$

bliskie 0

zamiast x podaje się na wejście x':

$$x_i^{(k)} ' = \eta_2(k)\, x_i^{(k)} + [1 - \eta_2 (k)] \sqrt{1/n}$$

$\eta_2(k)$ - rośnie do 1

UCZENIE DRUGIEJ WARSTWY

Warstwę Grossberga uczymy według reguły Widrow - Hoffa

$$v_{ij}^{(k+1)} = v_{ij}^{(k)} + \eta_3 (z_i - y_i) k_j$$

Proces uczenia warstwy Grossberga polega na wpisywaniu do tablicy „look up table” właściwych wartości, które mają być odpowiednią reakcją na pewną grupę sygnałów pojawiających się na wejściu sieci, a którą identyfikuje pewien neuron z warstwy Kohonena.

PRZYKŁAD ZADANIA DLA SIECI CP

Jak spędzać sobotni wieczór?

Sieć ma dwa wejścia:

x_1 - określające ilość pilnej pracy, jaka jest do wykonania

x_2 - podające "temperaturę" uczuć rodzinnych.

Sieć ma pięć wyjść:

y_1 oznacza czytanie książki,

y_2 - wspólne zakupy,

y_3 - wspólny spacer w parku,

y_4 - zabranie żony na wystawną kolację do restauracji,

y_5 - pozostanie w pracy.

Sieć uczono za pomocą następujących przykładów:

$x_1 = 0;\quad x_2 = 0;\quad y_1 = 1$ (gdy nie ma co robić a brak sympatii to najlepiej sobie poczytać);

$x_1 = 0,5;\quad x_2 = 0;\quad y_1 = 1$ (za mało pilnej pracy, żeby sobie psuć sobotnie popołudnie);

$x_1 = 0;\quad x_2 = 0,5;\quad y_2 = 1$ (uczuć rodzinnych wystarcza akurat na wspólne sobotnie zakupy);

$x_1 = 1;\quad x_2 = 1;\quad y_3 = 1$ (mimo nawału pracy da się wygospodarować czas na wspólny spacer w parku);

$x_1 = 0,5;\quad x_2 = 1;\quad y_4 = 1$ (najważniejsze są ucucia i romantyczna kolacja we dwoje);

$x_1 = 1;\quad x_2 = 0,5;\quad y_5 = 1$ (tę pilną pracę trzeba koniecznie wykonać).

Po treningu sieci przedstawiono zadanie

$x_1 = 0; \quad x_2 = 1$

Sieć podała odpowiedź

$y_4 = 1$

PEŁNA SIEĆ Z KONTRPROPAGACJĄ
FULL COUNTERPROPAGATION

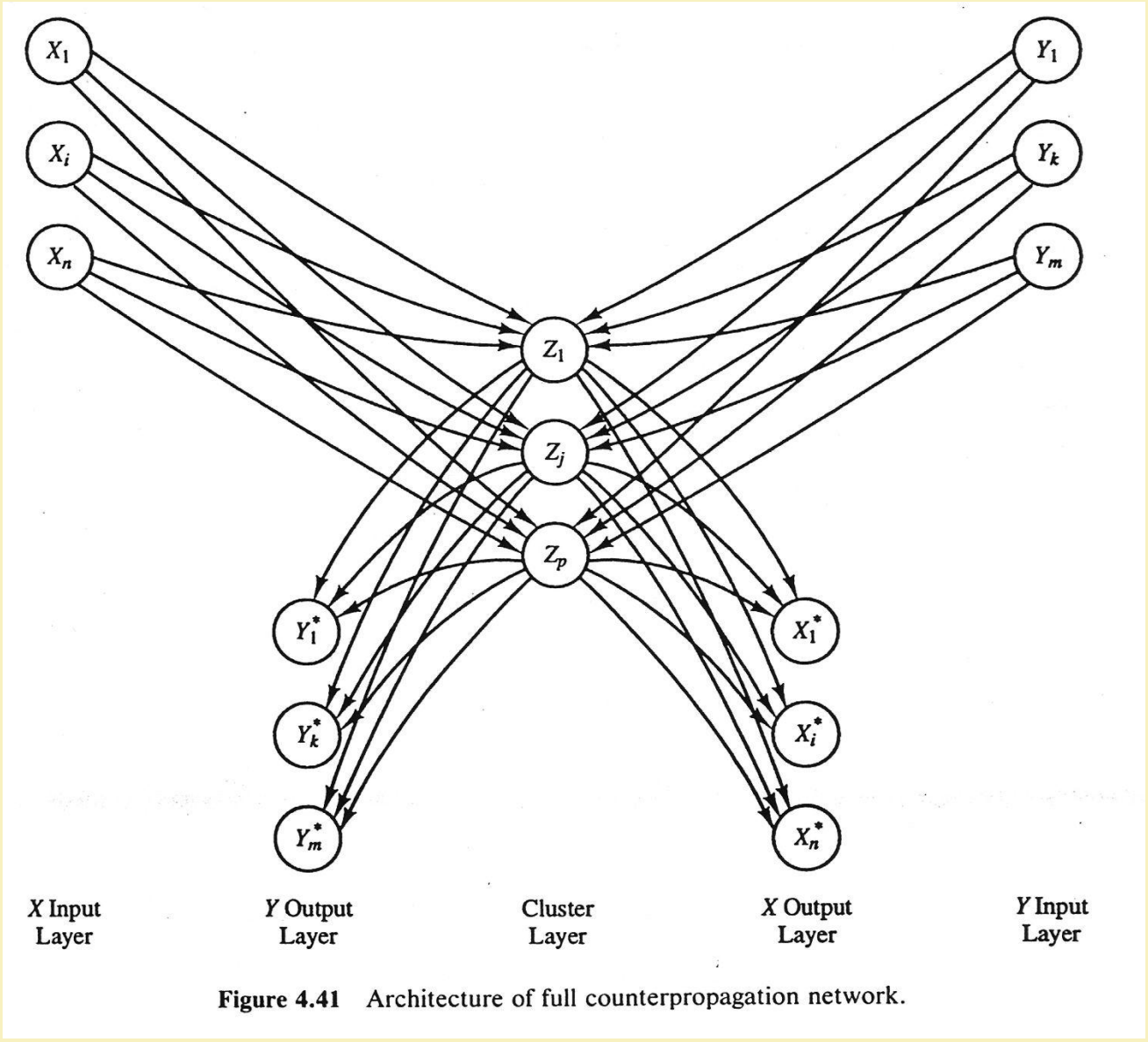
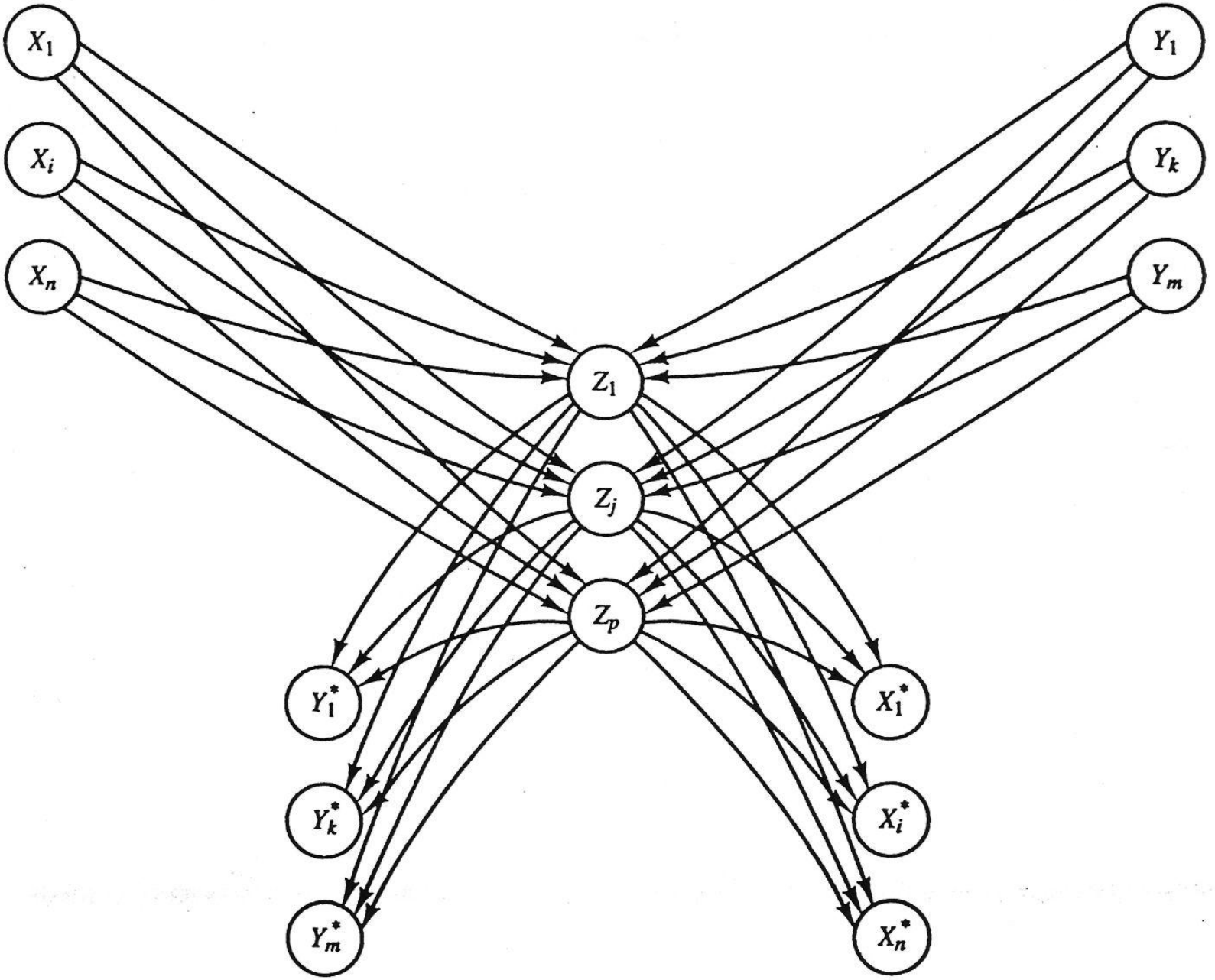


Figure 4.41 Architecture of full counterpropagation network.


X_1
X_i
X_n
Y_1
Y_k
Y_m
Z_1
Z_j
Z_p
Y_1^*
Y_k^*
Y_m^*
X_1^*
X_i^*
X_n^*
X Input Layer
Y Output Layer
Cluster Layer
X Output Layer
Y Input Layer

PEŁNA SIEĆ Z KONTRPROPAGACJĄ
Pierwsza faza uczenia

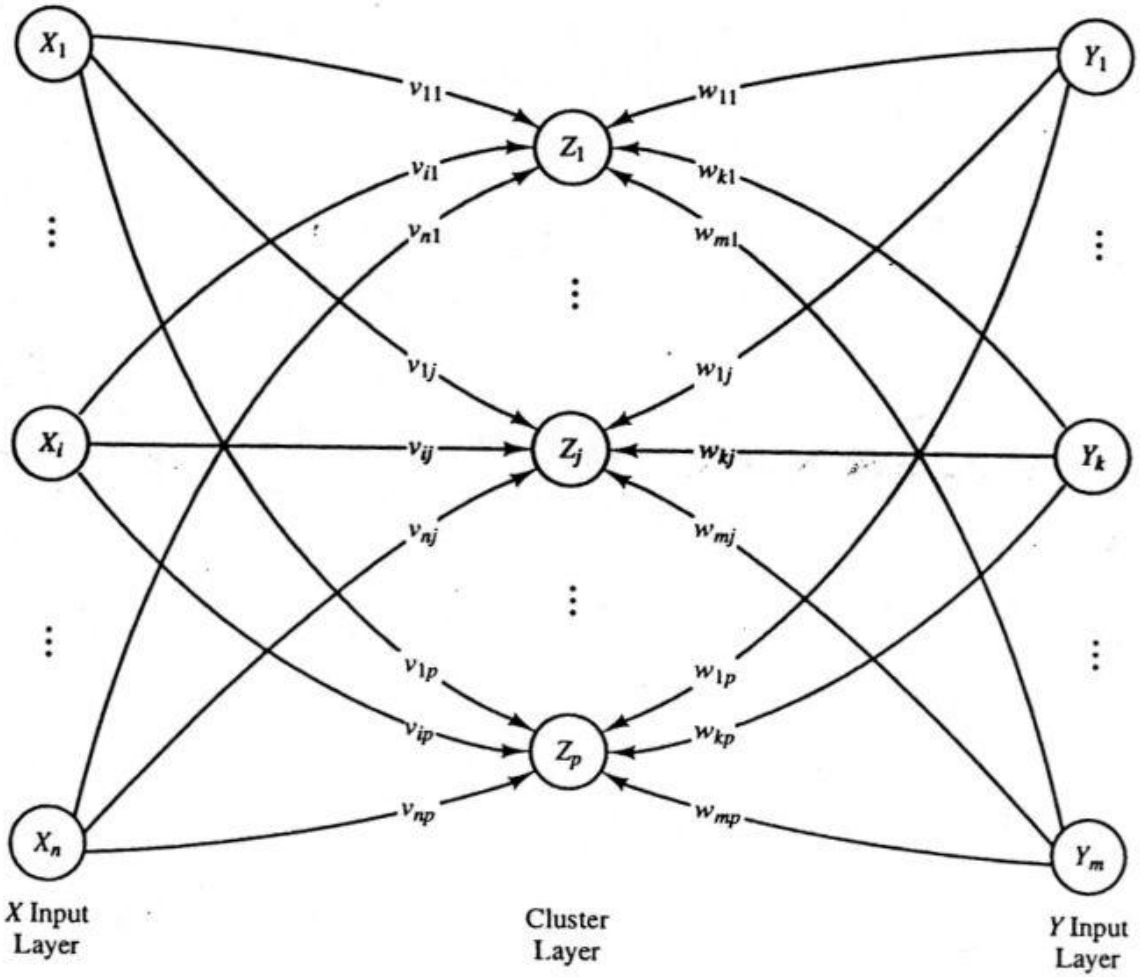


Figure 4.42 Active units during the first phase of counterpropagation training.

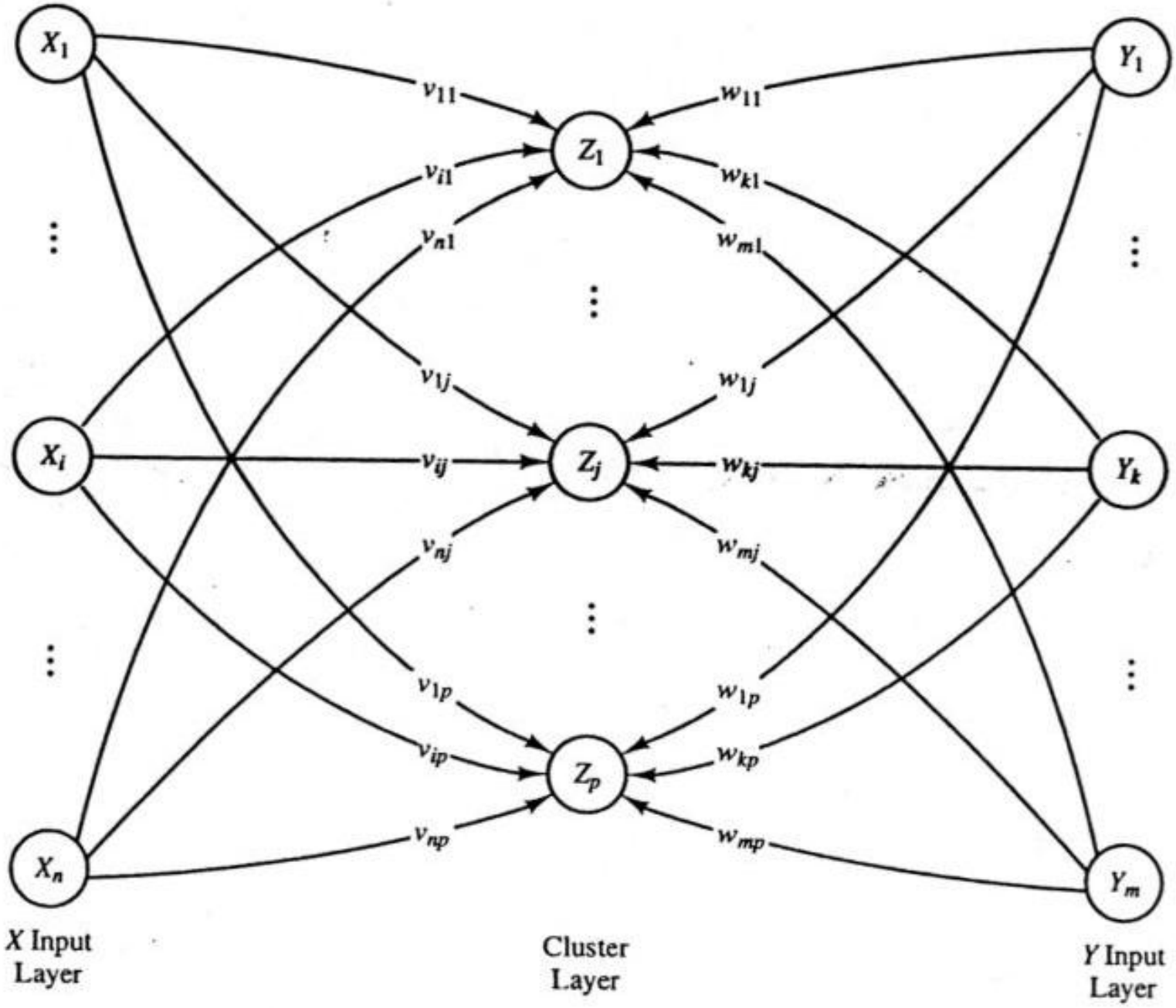


Figure 4.42 Active units during the first phase of counterpropagation training.

PEŁNA SIEĆ Z KONTRPROPAGACJĄ
Druga faza uczenia

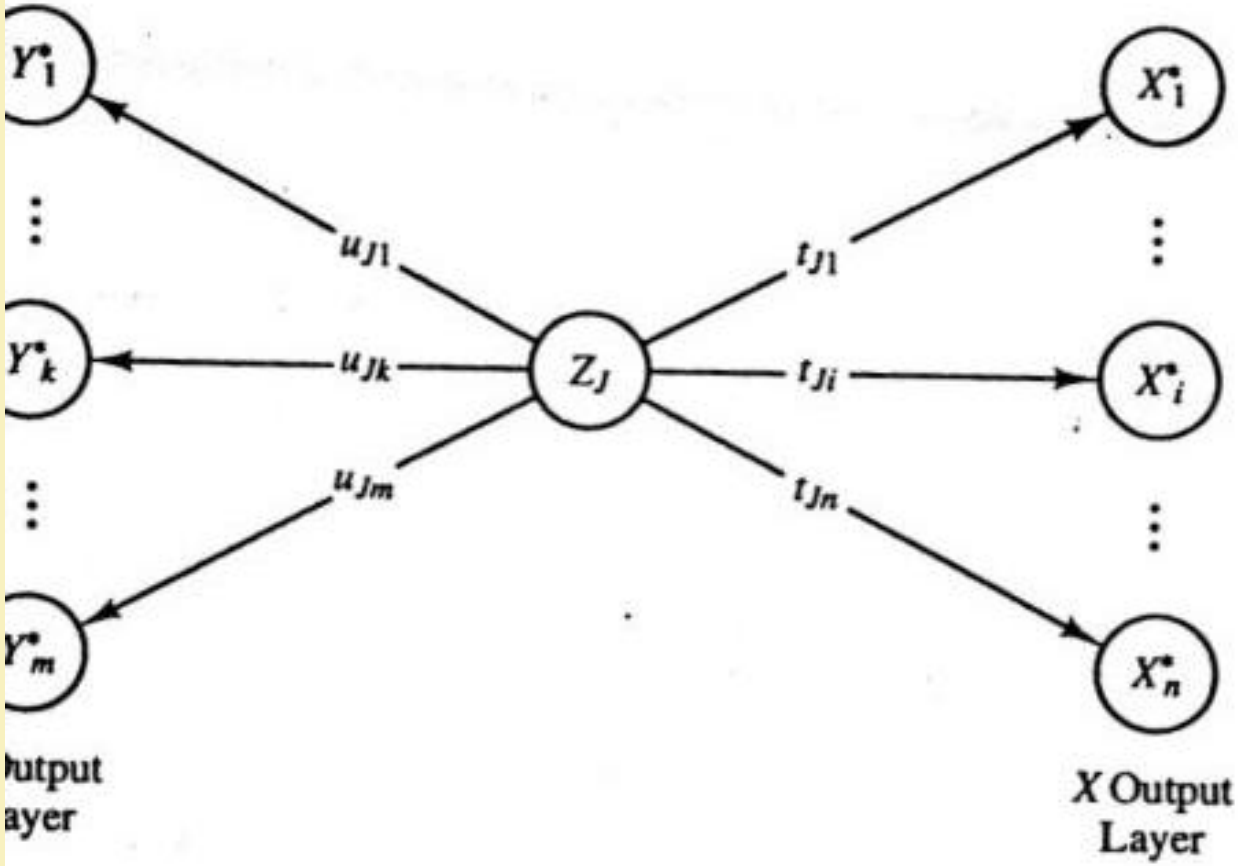
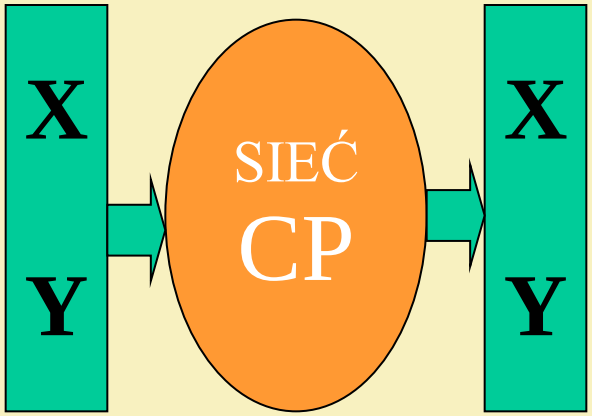


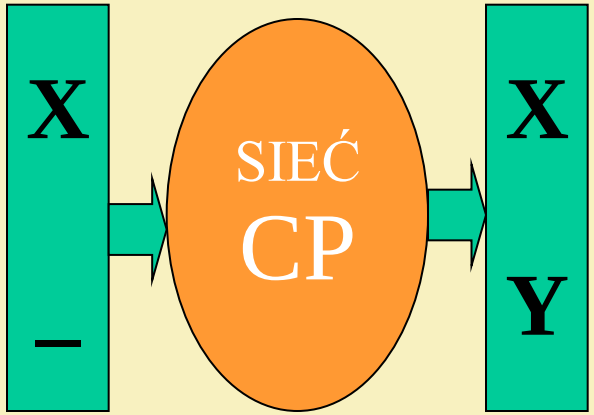
Figure 4.43 Second phase of counterpropagation training.

AUTOASOCJACJA

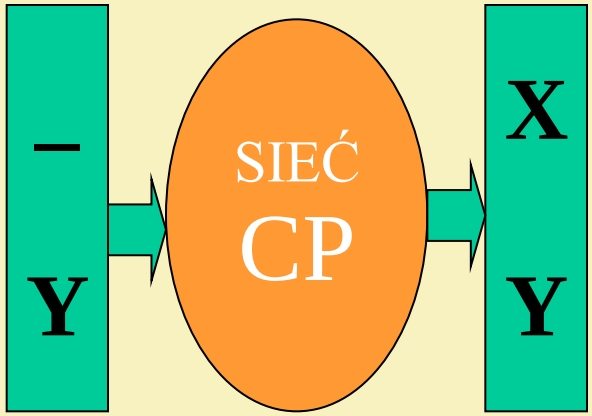
UCZENIE



EKSPLOATACJA



ALBO



LITERATURA

Tadeusiewicz Ryszard, *Sieci neuronowe*. W-wa 1993

Fausett Laurene, *Fundamental of Neural Networks Architectures, Algorithms, and Applications*. Prentice-Hall, Inc., 1994