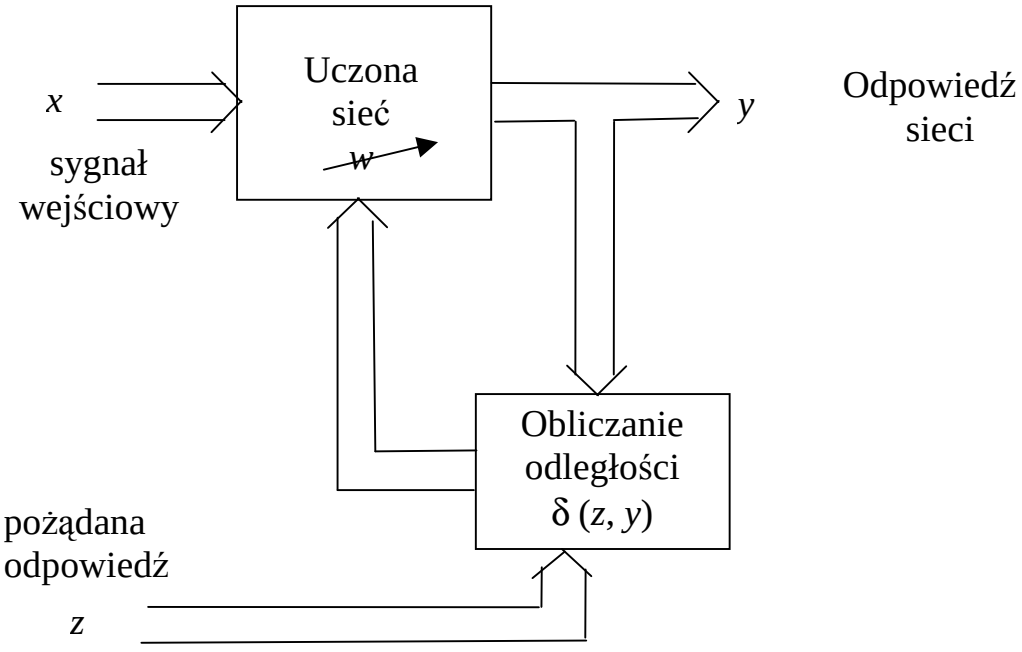
2.4. Algorytmy uczenia sieci neuronowych

Prosta struktura sieci jednokierunkowych sprawia, że są najchętniej stosowane. Ponadto metody uczenia ich należą również do popularnych i łatwych w realizacji.

Proces uczenia prowadzi do utrwalania określonych zachowań na bazie doświadczeń. Pod pojęciem uczenia sieci rozumie się wymuszenie na sieci określonego reagowania na zadane sygnały wejściowe. Efekty uczenia mogą być różne dlatego należy stosować proces weryfikacji „zdobytej" wiedzy. Dla sieci podobnie jak dla organizmów żywych stosuje się etap uczenia, testowania i aplikacji. Oczywiście pierwsze dwa mogą być stosowane wielokrotnie. Najczęściej uczymy sieć wykonywać obliczenia przez dostrajanie wartości wag w_{ij}., które odbywa się iteracyjnie. Można to zrobić dwoma sposobami: poprzez ***uczenie nadzorowane*** zwane inaczej ***uczenie z nauczycielem*** oraz poprzez ***uczenie bez nadzoru*** (***samouczenie***) [41], [60], [183], [216].

2.4.1. Uczenie z nauczycielem

Podczas **uczenia z nauczycielem** przy nowych danych wejściowych nauczyciel podpowiada pożądaną odpowiedź sieci *z*. Odległość $\delta(z,y)$ pomiędzy rzeczywistą *y,* a pożądaną odpowiedzią sieci *z* jest miarą błędu używaną do



Rys. 2.9. Schemat uczenia z nauczycielem

korekcji parametrów sieci (Rys. 2.9).
Zestaw obrazów wejściowych i wyjściowych użytych w czasie nauki nazywa się zbiorem uczącym.

W tym uczeniu wagi sieci są dobierane w sposób zapewniający, aby wyjścia z sieci zbliżały się do wyjść żądanych. Istnieje wiele sposobów uczenia nadzorowanego dlatego ważną sprawą jest wybranie skutecznej metody z wielu proponowanych. Największą popularność wśród tych metod uzyskała tzw. reguła **DELTY**.

2.4.1.1. Reguła DELTA

Opracowana przez Widrowa i Hoffa znalazła zastosowanie do uczenia elementów liniowych i nieliniowych. Reguła ta zakłada, że wraz z każdym wektorem wejściowym **X** do neuronu podawany jest sygnał **Z** (wymagana odpowiedź neuronu na sygnał **X**). Dla ciągu uczącego mającego postać:

$$\mathbf{U} = \left\langle \left\langle \mathbf{X}_1, \mathbf{Z}_1 \right\rangle, \left\langle \mathbf{X}_2, \mathbf{Z}_2 \right\rangle, \ldots, \left\langle \mathbf{X}_N, \mathbf{Z}_N \right\rangle \right\rangle.$$

w j – tym kroku procesu uczenia neuron odpowiada na sygnał $x^{(j)}$ sygnałem

$$y^{(j)} = \mathbf{W}^{(j)^{\mathrm{T}}} \cdot \mathbf{X}^{(j)} \tag{2.20}$$

oraz jest określany błąd:

$$\delta^{(j)} = z^{(j)} - y^{(j)}. \tag{2.21}$$

Na podstawie tego sygnału błędu $\delta^{(j)}$ oraz wektora wejściowego $\mathrm{X}^{(j)}$ możliwe jest takie skorygowanie wektora wag $\mathbf{W}^{(j)}$, by neuron generował sygnał bliższy

zadanemu. Nowy wektor wag $\mathbf{W}^{(j+1)}$ (w j+1-szym kroku) obliczany jest wg reguły DELTA:

$$\mathbf{W}^{(j+1)} = \mathbf{W}^{(j)} + \eta^{(j)} \cdot \delta^{(j)} \cdot \mathbf{X}^{(j)}, \tag{2.22}$$

gdzie: η - współczynnik uczenia.

Reguła ta daje się łatwo stosować pod warunkiem wprowadzenia początkowego wektora wag $\mathbf{W}^{(0)}$, dobranego losowo. Bezwarunkowo trzeba unikać przyjmowania jednakowych wartości dla różnych składowych wektora **W** na początku procesu uczenia. Nie dotrzymanie tego warunku prowadzi do braku postępu w początkowym etapie uczenia.

Celem procesu uczenia jest uzyskanie zgodności odpowiedzi neuronu $y^{(j)}$ z wymaganymi wartościami $z^{(j)}$, co daje się sprowadzić do minimalizacji pewnej funkcji kryterialnej Q:

$$\mathrm{Q} = \frac{1}{2} \sum_{j=1}^{N} (z^{(j)} - y^{(j)})^2 . \tag{2.23}$$

Przyjmując do rozważań algorytm spadku gradientu należy zmieniać każdą wagę w_i osobno o pewną wielkość Δw_i. Wzór gradientowego uczenia dla kroku j ma postać:

$$w_i^{(j+1)} - w_i^{(j)} = \Delta w_i^{(j)} = - \eta \frac{\partial \mathrm{Q}^{(j)}}{\partial w_i}, \tag{2.24}$$

gdzie: i – indeks oznacza konkretną wagę neuronu.

Uwzględniając, że Q jest zależne od y, a ponadto y jest funkcją wektora wag **W**, można obliczyć pochodną funkcji złożonej dla elementów nieliniowych (analogiczny jest sposób uczenia elementów liniowych

$$\frac{\partial Q^{(j)}}{\partial w_i} = \frac{\partial Q^{(j)}}{\partial y^{(j)}} \cdot \frac{\partial y^{(j)}}{\partial w_i} = \frac{\partial Q^{(j)}}{\partial y^{(j)}} \cdot \frac{dy^{(j)}}{de^{(j)}} \cdot \frac{\partial e^{(j)}}{\partial w_i} \tag{2.25}$$

gdzie:

$$\frac{\partial Q^{(j)}}{\partial y^{(j)}} = -(z^{(j)} - y^{(j)}) = -\delta^{(j)}, \tag{2.26}$$

$$\frac{\partial e^{(j)}}{\partial w_i} = x_i^{(j)}, \tag{2.27}$$

$$\frac{dy^{(j)}}{de^{(j)}} = \frac{d\varphi(e)}{de^{(j)}}. \tag{2.28}$$

Funkcję $\varphi(e)$ przyjmuje się zwykle w postaci funkcji logistycznej (wzór 2.5) ponieważ ma prostą pochodną:

$$\frac{d\varphi(e)}{de^{(j)}} = y^{(j)}(1 - y^{(j)}). \tag{2.29}$$

Formułę uczenia elementów nieliniowych zapisuje się:

$$\Delta w_i^{(j)} = \eta \delta^{(j)} \frac{d\varphi(e)}{de^{(j)}} \cdot x_i^{(j)}. \tag{2.30}$$

Opisany algorytm uczenia jest możliwy do bezpośredniego zastosowania jedynie w przypadku sieci jednowarstwowej. Dla sieci wielowarstwowych nie można go zastosować, ponieważ jeśli wektor $\mathbf{Y}^{(j)}$ sygnałów wyjściowych nie będzie odpowiadał wymaganiom (wystąpi błąd $\delta^{(j)}$), to nie będzie można w stanie ustalić, w jakim stopniu za pojawienie się tego błędu odpowiadają neurony warstwy wyjściowej, na których ten błąd się ujawnił, a w jakim stopniu błąd powstał w elementach wcześniejszej warstwy, których sygnały podawane były jako wejściowe do neuronów ocenianej warstwy.

2.4.1.2. Algorytm wstecznej propagacji błędów

Przez wiele lat nie znano metody skutecznego uczenia sieci wielowarstwowych a warstwy, dla których nie można było wyznaczyć sygnału błędu znane były pod nazwą warstw ukrytych. Dopiero w połowie lat 80-tych zaproponowany został

algorytm wstecznej propagacji błędów (**backpropagation**) polegający na tym, że mając wyznaczony błąd $\delta_m^{(j)}$ występujący podczas realizacji *j*-tego kroku procesu uczenia w neuronie o numerze *m* można podawać ten błąd wstecz (do kierunku przepływu informacji – stąd nazwa algorytmu) do wszystkich tych neuronów, których sygnały stanowiły wejścia dla *m*- tego neuronu [183]. Uczenie odbywa się przez minimalizację odpowiednio zdefiniowanej funkcji celu Q(**W**), przy czym wektor **W** reprezentuje wagi sieci poddawane optymalizacji. Najprostsza funkcja celu ma postać błędu średniokwadratowego (wzór 2.23). Zastosowanie różniczkowalnej funkcji aktywacji umożliwia minimalizację funkcji celu metodami gradientowymi. W metodach tych do aktualizowania wektora wag wykorzystuje się informację o gradiencie funkcji celu:

$$\nabla Q = \left[\frac{\partial Q}{\partial w_1}, \frac{\partial Q}{\partial w_2}, \ldots \frac{\partial Q}{\partial w_n}\right]^T, \qquad (2.31)$$

przy czym w każdym kroku uczenia wyznacza się tzw. kierunek minimalizacji **p**(**W***(k)*), gdzie *k* odpowiada określonemu cyklowi uczącemu. Najprostszą metodą wyboru kierunku minimalizacji **p**(**W**) jest wybór zgodny z kierunkiem ujemnego gradientu.

Jest to tzw. algorytm największego spadku, w którym:

$$\mathbf{p}(\mathbf{W}) = -\nabla Q\,(W). \qquad (2.32)$$

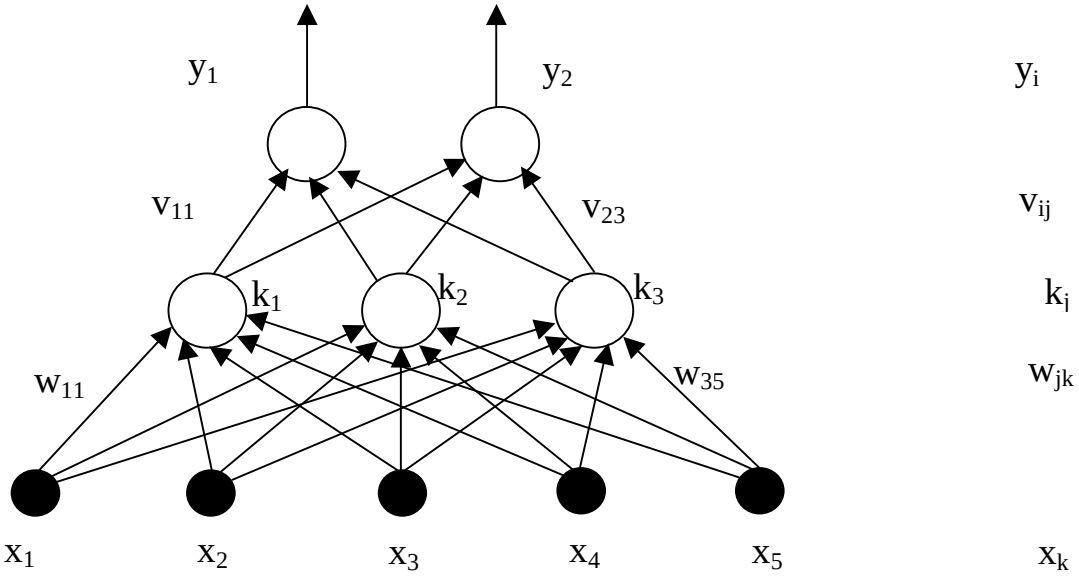
Algorytm ten umożliwia wyznaczenie minimum funkcji celu [160]. Uaktualnienie wektora wag odbywa się zgodnie ze wzorem:

$$\mathbf{W}^{(j+1)} = \mathbf{W}^{(j)} + \Delta\mathbf{W}, \qquad (2.32)$$

gdzie:

$$\Delta\mathbf{W} = \eta \cdot p(\mathbf{W}). \qquad (2.33)$$

Rozważmy jednokierunkową sieć dwuwarstwową (Rys. 2.10), w której sygnały wejściowe są stałe i równe określonym wartościom.



Rys. 2.10. Dwuwarstwowa sieć jednokierunkowa

Różne wzorce oznaczono indeksem μ, a więc wejście *k*-te jest równe $x_k^{(\mu)}$, gdy jest prezentowany wzorzec μ-ty, (μ=1, 2, … , *m*). Dla danego μ-tego wzorca *j*-ta jednostka ukryta otrzymuje sygnał pobudzenia sieciowego w postaci :

$$e_j^{(\mu)} = \sum_k w_{jk} \cdot x_k^{(\mu)} \tag{2.34}$$

i wytwarza sygnał wyjściowy:

$$k_j^{(\mu)} = \varphi(e_j^{(\mu)}) = \varphi(\sum_k w_{jk} \cdot x_k^{(\mu)}) . \tag{2.35}$$

Jednostka wyjściowa *i*-ta otrzymuje zatem sygnał:

$$e_i^{(\mu)} = \sum_j v_{ij} k_j^{(\mu)} = \sum_j v_{ij} \varphi(\sum_k w_{jk} \cdot x_{kj}^{(\mu)}) \tag{2.36}$$

i ostatecznie wytwarza sygnał wyjściowy:

$$y_i^{(\mu)} = \varphi(e_i^{(\mu)}) = \varphi(\sum_j v_{ij} \cdot k_j^{(\mu)}) = \varphi(\sum_j v_{ij} \cdot \ \varphi(\sum_k w_{jk} \cdot x_k^{(\mu)})) . \tag{2.36}$$

Miara błędu uczenia:

$$Q(w) = \frac{1}{2}\sum_{i\mu}[z_i^{(\mu)} - y_i^{(\mu)}]^2 \ , \tag{2.37}$$

ma po podstawieniu postać:

$$Q(w) = \frac{1}{2}\sum_{i\mu}[z_i^{(\mu)} - \varphi(\sum_j v_{ij} \cdot \varphi(\sum_k w_{jk} \cdot x_k^{(\mu)}))]^2 . \tag{2.38}$$

Jest ona ciągłą i różniczkowalną funkcją wszystkich wag, można więc zastosować algorytm spadku gradientu do uczenia właściwych wag [72]. Dla połączeń między jednostkami ukrytymi a wyjściowymi reguła spadku gradientu daje:

$$\Delta v_{ij} = -\eta\frac{\partial Q}{\partial v_{ij}} = \eta\sum_{\mu}[z_i^{(\mu)} - y_i^{(\mu)}]\varphi'(e_i^{(\mu)}) \cdot k_j^{(\mu)} = \eta\sum_{\mu}\delta_i^{(\mu)}k_j^{(\mu)} \tag{2.39}$$

gdzie: φ’(*e*) – pochodna funkcji aktywacji:

$$\delta_i^{(\mu)} = \varphi'(e_i^{(\mu)})(z_i^{(\mu)} - y_i^{(\mu)}) . \tag{2.40}$$

Aby obliczyć zmianę Δw_{jk} połączeń między jednostkami wejściowymi a ukrytymi, należy wykonać różniczkowanie względem wag w_{jk}.

$$\Delta w_{jk} = -\eta\frac{\partial Q}{\partial w_{jk}} = -\eta\sum_{\mu}\frac{\partial Q}{\partial k_j^{(\mu)}}\frac{\partial k_j^{(\mu)}}{\partial w_{jk}} =$$

$$= \eta\sum_{\mu i}(z_i^{(\mu)} - y_i^{(\mu)})\varphi'(e_i^{(\mu)}) \cdot v_{ij} \cdot \varphi'(e_j^{(\mu)}) \cdot x_k^{(\mu)} =$$

$$= \eta\sum_{\mu i}\delta_i^{(\mu)}v_{ij} \cdot \varphi'(e_j^{(\mu)}) \cdot x_k^{(\mu)} = \eta\sum_{\mu}\delta_j^{(\mu)}x_k^{(\mu)} , \tag{2.41}$$

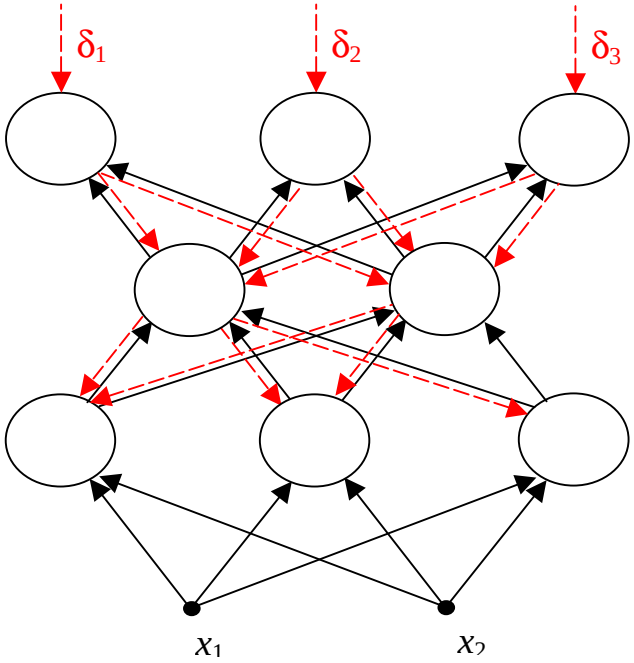
przy czym:

$$\delta_j^{(\mu)} = \varphi'(e_j^{(\mu)})\sum_i v_{ij}\delta_i^{(\mu)} . \tag{2.42}$$

W zasadzie dla dowolnej liczby warstw reguła modyfikacji metodą propagacji wstecznej ma zawsze postać:

$$\Delta w_{pq} = \eta \cdot \sum_{wzorce} \delta_{\text{wyjście}} \cdot x_{\text{wejście}} . \tag{2.43}$$

Znaczenie δ zależy od rozpatrywanej warstwy; dla ostatniej warstwy połączeń δ jest określone wzorem (2.40), a dla wszystkich pozostałych warstw jest dane przez równanie (2.42), które umożliwia wyznaczenie δ dla danej jednostki ukrytej k_j jako funkcji błędów jednostek y_i, które są przez nią pobudzone. „Przenoszone" wstecznie błędy mnożone są przez te same współczynniki w (wartości wag), przez które mnożone były przesyłane sygnały, tylko kierunek przesyłania informacji został odwrócony: od wyjścia do wejścia. Rys. 2.11 przestawia ideę algorytmu wstecznej propagacji, w którym linie przerywane reprezentują „przepływ" sygnałów błędów.



Rys. 2.11. Idea działania algorytmu propagacji wstecznej
w sieci trójwarstwowej

Uwzględniając bardziej szczegółowy schemat sieci z obliczanymi błędami poszczególnych warstw algorytm wstecznej propagacji błędu można zapisać jako następujące kroki:

1. przyjąć losowe wartości wektorów wag,
2. podać wybrany wzorzec na wejście sieci,

3. wyznaczyć sygnały wyjściowe wszystkich neuronów wyjściowych sieci zgodnie z (2.36)),
4. obliczyć błędy wszystkich neuronów warstwy wyjściowej (2.40),
5. obliczyć błędy w warstwach ukrytych, na podstawie znajomości błędów w warstwach, następnych zgodnie z (2.42),
6. Zmienić wartości wag; wzór (2.43),
7. wrócić do punktu 2.

Metodę propagacji błędów można uważać za metodę optymalizacji, minimalizującą zadane kryterium jakości za pomocą metody gradientowej ze stałym współczynnikiem korekcji. W praktyce okazuje się, że zastosowanie metody propagacji wstecznej może napotkać wiele trudności. Uczenie tą metodą jest skuteczne, ale bardzo powolne. Jego przyśpieszenie można uzyskać stosując różne techniki.

2.4.1.3. Metody przyśpieszające uczenie sieci

W metodzie propagacji wstecznej oblicza się gradient błędu i dokonuje się korekty wag każdorazowo po podaniu kolejnego wektora uczącego. Takie postępowanie nosi nazwę **przyrostowego uaktualniania wag**. Możliwe jest inne postępowanie, w którym oblicza się gradient błędu łącznego i dokonuje się korekty wag raz na jeden cykl, tzn. po podaniu wszystkich obrazów uczących. Wówczas jest to **kumulacyjne uaktualnianie wag**. Ponieważ gradient błędu łącznego jest sumą gradientów błędów dla poszczególnych obrazów, kumulacyjne uaktualnienie wag można zrealizować obliczając poprawki wag po każdym obrazie uczącym, ale bez dokonywania ich zmiany. Zmiana wag jest dokonywana dopiero po całym cyklu uczącym, czyli po podaniu wszystkich wzorców. Efektywność obu metod jest zależna od rozwiązywanego problemu, ale na ogół uaktualnianie przyrostowe jest bardziej efektywne od kumulacyjnego.

Istotnym problemem jest występowanie w funkcji błędu minimów lokalnych. Typową modyfikacją algorytmu pozwalającą wyjść z minimum

lokalnego jest dodawanie do wag, po każdym kroku uczenia, małych, przypadkowych wartości. Modyfikacja ta jest pewnym wariantem **algorytmów szukania przypadkowego**. W metodzie tej do aktualnej wartości wektora wag tak długo dodawany jest losowy wektor, aż uzyskany nowy wektor wag generuje mniejszy błąd. Wtedy nowy wektor wag staje się aktualnym wektorem i proces się powtarza. Najprostszym i jednocześnie bardzo efektywnym sposobem zmniejszającym możliwość zatrzymania się w minimum lokalnym jest uaktualnianie przyrostowe wag z wzorcami podawanymi w losowej kolejności.

Prostym podejściem pozwalającym uniknąć zatrzymania się w lokalnym minimum jest kilkakrotne powtarzania procesów uczenia, rozpoczynając od różnych wartości początkowych wag. Nowy proces uczenia można rozpocząć, gdy wartości błędu bieżącego procesu przestają maleć. Powtarzanie uczenia wydłuża łączny jego czas, który dla dużej liczby wymiarów wektorów wag może stać się parametrem krytycznym. Wybór wielkości początkowych wag też jest istotny. Okazuje się, że zbyt duże wagi początkowe powodują nasycenie sigmoidalnych funkcji aktywacji i proces uczenia może być bardzo wolno zbieżny lub zatrzyma się w minimum lokalnym. Zalecanym postępowaniem ogólnym jest taki losowy wybór wag, aby pobudzenie łączne było nieco mniejsze od jedności.

Duży wpływ ma szybkość uczenia sieci ma wybór i kształt funkcji aktywacji sigmoidalnej, która charakteryzuje się współczynnikiem nachylenia λ. Przy ustalonym współczynniku korekcji η, wszystkie wagi korygowane są proporcjonalnie do współczynnika nachylenia λ. Wynika stąd, że wybór dużych wartości λ może prowadzić do podobnych rezultatów jak przyjęcie dużych wartości η. W związku z tym rozsądny jest wybór standardowej wartości λ, np. $\lambda=1$ i wpływanie na szybkość zbieżności wyłącznie przez współczynnik η. Współczynnik ten powinien być dobierany eksperymentalnie do każdego problemu. W grę wchodzi też stosowanie zmiennych współczynników η. Ogólna zasada polega na tym, aby

zwiększać η o stałą wartość, gdy błąd systematycznie maleje, oraz zmniejszać geometrycznie, gdy błąd wzrasta [216].

Przytoczone wyżej metody przyspieszania procesu uczenia zostały opracowane dość wcześnie. Są często stosowane ale ich możliwości są ograniczone. Dlatego w ostatnich latach prowadzone były badania nad nowymi metodami przyspieszającymi uczenie, opartymi na **algorytmach gradientowych.** Bazują one na rozwinięciu w szereg Taylora funkcji celu Q(**W**) w najbliższym sąsiedztwie znanego rozwiązania **W**. W przypadku funkcji wielu zmiennych rozwinięcie to można przedstawić w otoczeniu określonego wcześniej punktu na kierunku **p** (w praktyce wykorzystuje się trzy pierwsze składniki rozwinięcia Taylora):

$$\mathrm{Q}(\mathbf{W}+\mathbf{p}) = \mathrm{Q}(\mathbf{W}) + [\mathbf{g}(\mathbf{W})]^T \cdot \mathbf{p} + \frac{1}{2} p^{\mathrm{T}} \mathbf{H}(\mathbf{W}) \cdot \mathbf{p} + O(h^3), \qquad (2.44)$$

gdzie:

$$\mathbf{g}(\mathbf{W}) = \nabla \mathrm{Q} = \left[\frac{\partial \mathrm{Q}}{\partial w_1}, \quad \frac{\partial \mathrm{Q}}{\partial w_2}, \quad \ldots, \quad \frac{\partial \mathrm{Q}}{\partial w_n} \right]^{\mathrm{T}}, \qquad (2.45)$$

jest wektorem gradientu, natomiast:

$$\mathrm{H}(\mathrm{W}) = \begin{bmatrix} \dfrac{\partial \mathrm{Q}}{\partial \mathrm{W}_1 \partial \mathrm{W}_1} & \cdots & \dfrac{\partial \mathrm{Q}}{\partial \mathrm{W}_n \partial \mathrm{W}_1} \\ \\ \dfrac{\partial \mathrm{Q}}{\partial \mathrm{W}_1 \partial \mathrm{W}_n} & \cdots & \dfrac{\partial \mathrm{Q}}{\partial \mathrm{W}_n \partial \mathrm{W}_n} \end{bmatrix} \qquad (2.46)$$

jest symetryczną macierzą kwadratową drugich pochodnych (hesjan) w której: **p** – wektor kierunkowy, zależny od aktualnych wartości wektora **W**,
O (h^3) – błąd obcięcia szeregu.

Punkt rozwiązania $\mathbf{W}=\mathbf{W}^{(j)}$ (*j*- wskaźnik oznaczający wartość zmiennych uzyskanych w *j*-tym cyklu) jest optymalnym punktem funkcji Q(**W**), jeśli $\mathbf{g}(\mathbf{W}^{(j)})=0$, a hesjan $\mathbf{H}(\mathbf{W}^{(j)})$ jest dodatnio określony. Przy spełnieniu tych warunków

funkcja w dowolnym punkcie należącym do sąsiedztwa $\mathbf{W}^{(j)}$ ma wartość większą niż w punkcie $\mathbf{W}^{(j)}$, a zatem punkt ten jest rozwiązaniem odpowiadającym minimalnej wartości funkcji celu.
Kierunek poszukiwań **p** i krok η dobierane są tak, aby dla nowego punktu

$$\mathbf{W}^{(j+1)} = \mathbf{W}^{(j)} + \eta^{(j)}\,\mathbf{p}^{(j)}, \tag{2.47}$$

była spełniona zależność:

$$Q(\mathbf{W}^{(j+1)}) < Q(\mathbf{W}^{(j)}). \tag{2.48}$$

Poszukiwanie minimum trwa dotąd, aż warunki:

$$Q(\mathbf{W}^{(j-1)}) - Q(\mathbf{W}^{(j)}) \le \varepsilon\left(1 + \left|Q(\mathbf{W}^{(j)})\right|\right),$$

$$\left\|\mathbf{W}^{(j-1)} - \mathbf{W}^{(j)}\right\| \le \sqrt{\varepsilon}\,(1 + \left\|\mathbf{W}^{(j)}\right\|), \tag{2.49}$$

$$\left\|\mathbf{g}(\mathbf{W}^{(j)})\right\| \le \sqrt[3]{\varepsilon}\left(1 + \left|Q(\mathbf{W}^{(j)})\right|\right),$$

zostaną spełnione bądź też nie zostanie przekroczony określony czas obliczeń (ε - jest to element oznaczający dokładność obliczeń). Te wiadomości są podstawą zrozumienia metod opartych na algorytmach gradientowych.

Algorytm największego spadku

Ograniczając się w rozwinięciu Taylora do liniowego przybliżenia funkcji $Q(\mathbf{W})$ w najbliższym sąsiedztwie znanego rozwiązania **W** mamy:

$$Q(\mathbf{W}^{(j)} + \pi^{(j)}) = Q(\mathbf{W}^{(j)}) + [\mathbf{g}(\mathbf{W}^{(j)})]^{\mathrm{T}} \cdot \mathbf{p}^{(j)} + O(h^2)\,. \qquad (2.50)$$

Aby spełnić zależność:

$$Q(\mathbf{W}^{(j+1)}) < Q(\mathbf{W}^{(j)}) \qquad (2.51)$$

wystarczy dobrać:

$$[\mathbf{g}(\mathbf{W}^{(j)})]^{\mathrm{T}} \cdot \mathbf{p}^{(j)} < 0\,, \qquad (2.52)$$

przy czym przyjęcie $\mathbf{p}^{(j)} = -\mathbf{g}(\mathbf{W}^{(j)})$ spełnia ten warunek.

Metoda ta jest mało efektywna, dlatego polecane jest stosowanie metody momentum.

Metoda momentum

Aktualizacja wag sieci:

$$\mathbf{W}^{(j+1)} = \mathbf{W}^{(j)} + \Delta\mathbf{W}^{(j)}, \qquad (2.53)$$

następuje wg zmienionej formuły określającej $\Delta\mathbf{W}^{(j)}$:

$$\Delta\mathbf{W}^{(j)} = \eta^{(j)} \cdot \mathbf{p}^{(j)} + \alpha(\mathbf{W}^{(j)} - \mathbf{W}^{(j-1)}), \qquad (2.54)$$

gdzie: α to współczynnik momentu z przedziału [0,1].

Pierwszy składnik wyrażenia odpowiada zwykłej metodzie uczenia, drugi uwzględnia ostatnią zmianę wag i jest niezależny od aktualnej wartości gradientu. Im większa jest wartość współczynnika α, tym składnik wynikający z momentu ma większy wpływ na dobór wag. Przy wartości α=0,9 zwiększa się 10-krotnie wartość współczynnika uczenia. Dobór wartości współczynnika momentu jest niełatwą sprawą i wymaga wielu eksperymentów, mających na celu dopasowanie jego wartości do specyfiki rozwiązywanego problemu [160]. Metoda ta jest dużo efektywniejsza od metody największego spadku.

Algorytm zmiennej metryki

W metodzie tej wykorzystuje się kwadratowe przybliżenie funkcji Q(**W**) w sąsiedztwie znanego rozwiązania $\mathbf{W}^{(j)}$:

$$Q(\mathbf{W}^{(j)} + \mathbf{p}^{(j)}) \approx Q(\mathbf{W}^{(j)}) + [\mathbf{g}(\mathbf{W}^{(j)})]^{\mathrm{T}} \cdot \mathbf{p}^{(j)} + \frac{1}{2}\mathbf{p}^{(j)^{\mathrm{T}}} \cdot \mathbf{H}(\mathbf{W}^{(j)}) \cdot \mathbf{p}^{(j)} + O(h^3).$$

Minimum tej funkcji wymaga, aby:

$$\frac{dQ(\mathbf{W}^{(j)} + \mathbf{p})}{d\mathbf{p}} = 0. \tag{2.55}$$

Dokonując operacji różniczkowania otrzymuje się:

$$\mathbf{g}(\mathbf{W}^{(j)}) + \mathbf{H}(\mathbf{W}^{(j)})\mathbf{p}^{(j)} = 0,$$

ostatecznie:

$$\mathbf{p}^{(j)} = -[\mathbf{H}(\mathbf{W}^{(j)})]^{-1} \cdot \mathbf{g}(\mathbf{W}^{(j)}). \tag{2.56}$$

Wzór ten wyznacza w sposób jednoznaczny taki kierunek $\mathbf{p}^{(j)}$, który zapewnia minimum funkcji celu. Aby wyznaczyć ten kierunek, należy w każdym cyklu określić wartość gradientu *g* oraz hesjanu **H** w punkcie $\mathbf{W}^{(j)}$. Hesjan w każdym kroku musi być dodatni, co jest trudne do spełnienia. Dlatego zamiast hesjanu $\mathbf{H}(\mathbf{W}^{(j)})$ stosuje się jego przybliżenie $\mathbf{G}(\mathbf{W}^{(j)})$.

W metodzie zmiennej metryki w każdym kroku modyfikuje się hesjan z kroku poprzedniego o pewną poprawkę. Poprawka musi być tak dobrana, aby aktualna wartość hesjanu $\mathbf{G}(\mathbf{W}^{(j)})$ przybliżała krzywiznę funkcji celu Q zgodnie z zależnością:

$$\mathbf{G}(\mathbf{W}^{(j)})(\mathbf{W}^{(j)} - \mathbf{W}^{(j-1)}) = \mathbf{g}(\mathbf{W}^{(j)}) - \mathbf{g}(\mathbf{W}^{(j-1)}). \tag{2.57}$$

Metoda ta jest obecnie uważana za jedną z najlepszych metod optymalizacji funkcji wielu zmiennych [160].

Algorytm Levenberga – Marquardta

W metodzie tej dokładną wartość hesjanu **H**(**W**) we wzorze (2.56) zastępuje się jego wartością aproksymowaną **G**(**W**), określaną na podstawie informacji zawartej w gradiencie z uwzględnieniem czynnika regularyzującego. Przy wprowadzeniu metody przyjęto definicję funkcji celu w postaci odpowiadającej istnieniu jednego wzorca uczącego:

$$Q(\mathbf{W}) = \frac{1}{2}\sum_{i=1}^{m}[e_i(W)]^2 , \tag{2.58}$$

przy czym:

$$e_i = [z_i - y_i(\mathbf{W})]. \tag{2.59}$$

Przy wprowadzonych oznaczeniach:

$$\mathbf{E}(\mathbf{W}) = \begin{bmatrix} e_1(w) \\ e_2(w) \\ \\ e_m(w) \end{bmatrix}, \qquad \mathbf{J}(\mathbf{W}) = \begin{bmatrix} \frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} & \cdots & \frac{\partial e_1}{\partial w_n} \\ \frac{\partial e_2}{\partial W_1} & \frac{\partial e_2}{\partial w_2} & \cdots & \frac{\partial e_2}{\partial w_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_m}{\partial w_1} & \frac{\partial e_n}{\partial w_2} & \cdots & \frac{\partial e_m}{\partial w_n} \end{bmatrix}, \tag{2.60}$$

wektor gradientu i aproksymowana macierz hesjanu odpowiadające funkcji celu (2.58) są określone w postaci:

$$\mathbf{g}(\mathbf{W}) = [\mathbf{J}(\mathbf{W})]^{\mathrm{T}} \cdot \mathbf{E}(\mathbf{W}) \tag{2.61}$$

$$\mathbf{G}(\mathbf{W}) = [\mathbf{J}(\mathbf{W})]^{\mathrm{T}} \cdot \mathbf{J}(\mathbf{W}) + \mathbf{R}(\mathbf{W}) \tag{2.62}$$

w której **R**(**W**) – oznacza składniki rozwinięcia hesjanu **H** (**W**) zawierającego wyższe pochodne względem **W**. Wobec trudności w określeniu wartości **R**(**W**) wprowadza się czynnik regulujący υ zwany parametrem **Levenberga** – **Marquardta**.

Aproksymowana macierz hesjanu w *j*-tym kroku algorytmu przyjmuje postać:

$$\mathbf{G}(\mathbf{W}^{(j)}) = [\mathbf{J}(\mathbf{W}^{(j)})]^{\mathbf{T}} \mathbf{J}(\mathbf{W}^{(j)}) + \upsilon^{(j)}\mathbf{1}. \tag{2.63}$$

Na początku procesu uczenia przyjmuje się wartość parametru $\upsilon^{(j)}$ bardzo dużą. W takim przypadku:

$$\mathbf{G}(\mathbf{W}^{(j)}) \approx \upsilon^{(j)}1 \tag{2.64}$$

i poszukiwanie kierunku odbywa się zgodnie z metodą największego spadku (2.56)

$$\mathbf{p}^{(j)} = -\frac{\mathbf{g}(\mathbf{W}^{(j)})}{\upsilon^{(j)}} \tag{2.65}$$

W miarę zmniejszania się błędu i zbliżania się do rozwiązania, parametr $\upsilon^{(j)}$ jest zmniejszany. O skuteczności działania algorytmu decyduje odpowiedni dobór $\upsilon^{(j)}$. Duża wartość początkowa $\upsilon^{(j)}$ w miarę postępów optymalizacji musi ulec redukcji aż do wartości zerowej przy rozwiązaniu bliskim optymalnemu [160].

Metoda gradientów sprzężonych

W metodzie tej rezygnuje się z informacji o hesjanie. Kierunek poszukiwań $\mathbf{p}^{(j)}$ jest konstruowany w taki sposób, by był sprzężony ze wszystkimi poprzednimi kierunkami $\mathbf{p}^{(0)}, \mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(j-1)}$.

Zbiór wektorów $\mathbf{p}_i$; $i=0,1,\ldots,j$ jest wzajemnie sprzężony względem macierzy $\mathbf{G}$, jeżeli:

$$\mathbf{p}_i^T \cdot \mathbf{G} \cdot \mathbf{p}_m = 0 \qquad i \neq m. \tag{2.66}$$

Wektor $p^{(j)}$, który spełnia powyższe założenia, ma postać:

$$\mathbf{p}^{(j)} = -\mathbf{g}^{(j)} + \sum_{m=0}^{j-1} \beta_{j\,m} \mathbf{p}_m \tag{2.67}$$

przy czym:

$\mathbf{g}^{(j)} = \mathbf{g}(\mathbf{W}^{(j)})$ – aktualna wartość wektora gradientu,

$\beta^{(j)}$ – współczynnik sprzężenia.

Sumowanie w drugiej części wzoru dotyczy wszystkich poprzednich kierunków minimalizacyjnych. Metodę tą stosuje się jako jedyny skuteczny algorytm optymalizacji przy bardzo dużej liczbie zmiennych sięgających nawet kilkudziesięciu tysięcy. Dzięki małym wymaganiom co do pamięci i stosunkowo niewielkiej złożoności obliczeniowej metoda ta umożliwia efektywne rozwiązanie bardzo dużych problemów optymalizacyjnych [160].

Omówione algorytmy określają jedynie kierunek, wzdłuż którego funkcja maleje, nie podając wielkości kroku, dla którego funkcja ta osiąga wartość minimalną na danym kierunku. Po określeniu właściwego kierunku $\mathbf{p}^{(j)}$ i wyborze

na nim nowego punktu rozwiązania $\mathbf{W}^{(j+1)}$, dla którego jest spełniony warunek (wzór 2.51) należy tak dobrać $\eta^{(j)}$, aby nowy punkt rozwiązania:

$$\mathbf{W}^{(j+1)} = \mathbf{W}^{(j)} + \eta^{(j)}\, \mathrm{p}^{(j)}, \tag{2.68}$$

leżał możliwie blisko minimum funkcji Q($\mathbf{W}$) na kierunku $\mathbf{p}^{(j)}$.

Właściwy dobór współczynnika $\eta^{(j)}$ ma ogromny wpływ na zbieżność algorytmu optymalizacji do minimum funkcji celu. Przyjęcie zbyt małej wartości η powoduje niewykorzystanie możliwości zminimalizowania wartości funkcji celu w danym kroku i konieczność jego powtórzenia w następnym. Zbyt duży krok powoduje przeskoczenie minimum funkcji i podobny efekt jak poprzednio.

Istnieje wiele sposobów doboru współczynnika uczenia η. Najprostszy polega na przyjęciu stałej wartości w całym procesie optymalizacyjnym. Jest to sposób najmniej efektywny, wymaga długiego czasu uczenia i ma skłonność do częstego utykania w minimum lokalnym.

Inną, bardziej skuteczną metodą jest przyjęcie adaptacyjnych zmian współczynnika uczenia η dopasowujących się do aktualnych zmian wartości funkcji celu w czasie uczenia. W metodzie tej na podstawie porównania sumacyjnego błędu:

$$\varepsilon = \sqrt{\sum_{j=1}^{M} (z_j - y_j)^2}\ , \tag{2.69}$$

w *i*-tej iteracji z jej poprzednią wartością określa się strategię zmian wartości η. W celu przyspieszenia procesu uczenia proponuje się ciągłe zwiększanie η sprawdzając jednocześnie, czy błąd ε nie zacznie wzrastać w porównaniu z błędem obliczanym przy poprzedniej wartości η.

Jednak najefektywniejszym sposobem doboru η jest minimalizacja kierunkowa funkcji celu na wyznaczonym wcześniej kierunku $\mathbf{p}^{(j)}$. Celem jest takie dobranie wartości $\eta^{(j)}$, aby nowy punkt $\mathbf{W}^{(j+1)}$ (wzór 2.67) odpowiadał minimum

funkcji celu na danym kierunku $\mathbf{p}^{(j)}$. Jeżeli $\eta^{(j)}$ odpowiada dokładnie minimum funkcji celu na danym kierunku $\mathbf{p}^{(j)}$, to pochodna kierunkowa w punkcie $\mathbf{W}^{(j+1)}$ musi być równa zeru. Postępowanie minimalizacyjne przeprowadza się dopóki są spełnione warunki:

$$[\mathbf{g}(\mathbf{W}^{(j)} + \eta^{(j)}\mathbf{p}^{(j)})]^{\mathrm{T}} \cdot \mathbf{p}^{(j+1)} \geq \gamma_2 [\mathbf{g}(\mathbf{W}^{(j)})]^{\mathrm{T}} \cdot \mathbf{p}^{(j)}, \qquad (2.70)$$

$$Q(\mathbf{W}^{(j)} + \eta^{(j)}\mathbf{p}^{(j)}) - Q(\mathbf{W}^{(j)}) \geq \gamma_1 \eta^{(j)} [\mathbf{g}(\mathbf{W}^{(j)})]^{\mathrm{T}} \cdot \mathbf{p}_k . \qquad (2.71)$$

Dogodne jest przyjęcie $0 \leq \gamma_1 \leq \gamma_2 < 1$.

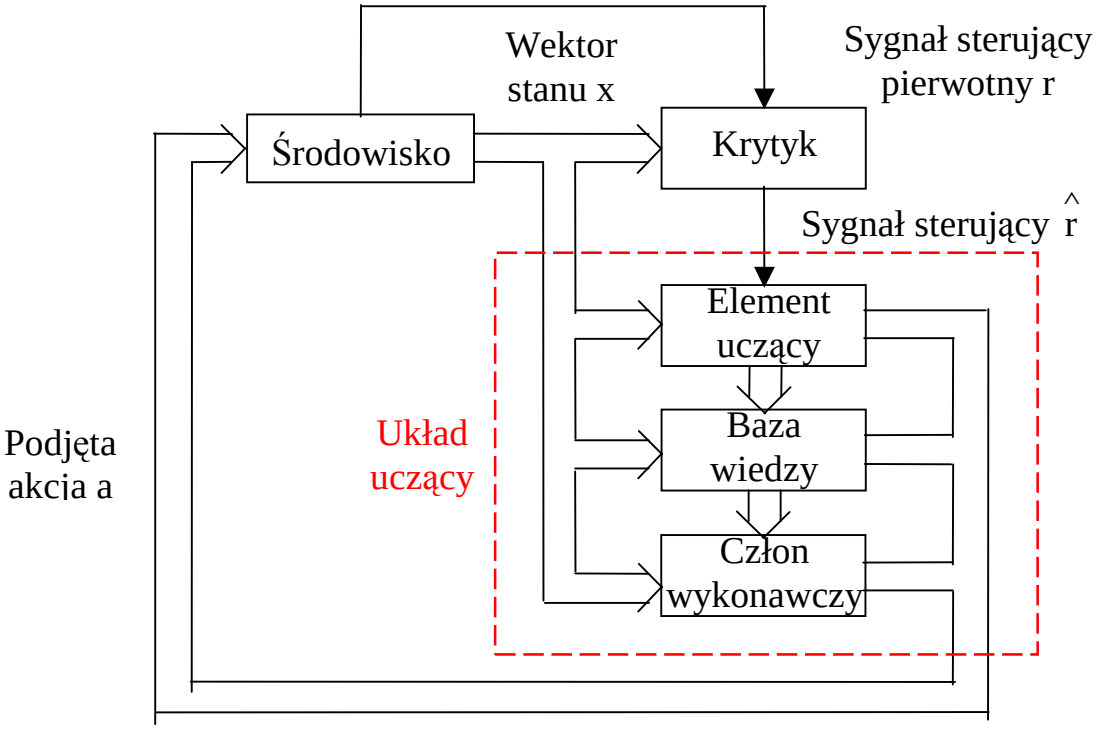
Wszystkie przedstawione metody są metodami lokalnymi, prowadzącymi do minimum lokalnego funkcji celu. W przypadku znajomości minimum globalnego można ocenić, czy osiągnięte minimum lokalne leży blisko rozwiązania idealnego. Jeśli rozwiązanie lokalne jest niezadowalające, to można powtórzyć proces uczenia przy innych wartościach startowych wag i innych parametrach procesu uczenia. Można przy tym pominąć osiągnięte rozwiązanie i rozpocząć proces uczenia od nowych, losowo wybranych wartości wag, albo też dodać wartości losowe do aktualnie uzyskanych rozwiązań i kontynuować proces uczenia. Ta ostatnia metoda (**jog of weights**) jest godna polecenia, gdyż nie zatraca uzyskanych już rezultatów uczenia.

2.4.1.4. Uczenie ze wzmocnieniem (z krytykiem)

W rozpatrywanym dotąd uczeniu pod nadzorem zakładano, że dla każdego wzorca wejściowego są znane prawidłowe wartości wyjściowe. Ale w pewnych sytuacjach tak szczegółowe informacje nie są dostępne. Wiadome jest tylko, czy wyjście jest dobre, czy złe. W tym przypadku stosuje się procedurę **uczenia ze wzmocnieniem**. Uczenie to jest jedną z postaci uczenia pod nadzorem ponieważ,

mimo wszystko, sieć otrzymuje sygnał zwrotny ze środowiska. Ale to sprzężenie zwrotne – jeden sygnał wzmocnienia tak/nie – jest tylko oceną, a nie instrukcją. Uczenie ze wzmocnieniem jest niekiedy nazywane **uczeniem z krytykiem** (**reinforcement learning**), w przeciwieństwie do uczenia z nauczycielem. W problemach uczenia ze wzmocnieniem przyjmuje się na ogół, że sieć działa w pewnym środowisku. Środowisko dostarcza sygnały na wejścia sieci, odbiera sygnały wyjściowe i na tej podstawie wytwarza sygnały wzmocnienia [72].

Jeśli działanie podjęte przez układ uczący daje wynik pozytywny, to następuje wzmocnienie tendencji do właściwego zachowania się systemu w podobnych sytuacjach w przyszłości. W przeciwnym przypadku, jeśli wynik jest negatywny, to następuje osłabienie tendencji takiego działania systemu. Typowy schemat blokowy układu z krytykiem przedstawiony jest na Rys.2.12.



Rys.2.12. Struktura sieci neuronowej z krytykiem [160]

Układ uczący współpracuje ze środowiskiem za pośrednictwem krytyka, który na podstawie aktualnego stanu środowiska i wiadomości co do jego

przyszłych zmian wypracowanej na podstawie aktualnej wiedzy przekazuje sygnał sterujący $\hat{r}$ umożliwiający podjęcie odpowiedniej akcji **a**, wpływającej na stan środowiska ***x***.

System należy do kategorii układów z opóźnieniem, gdyż sygnał sterujący r w chwili *k* jest określany na podstawie stanu środowiska w chwilach poprzednich i sekwencji tych zmian .

Uczenie z krytykiem, w odróżnieniu od uczenia pod nadzorem, ocenia skutki podjętej akcji i w zależności od tego oraz aktualnej bazy danych podejmuje decyzję co do dalszej akcji. Jest znacznie bardziej uniwersalne w zastosowaniu, gdyż nie wymaga obecności sygnałów żądanych na wyjściu systemu. Jednocześnie jego realizacją praktyczna jest bardziej skomplikowana.

2.4.2. Uczenie bez nauczyciela (samouczenie)

Metody samouczenia polegają na podawaniu na wejście sieci wyłącznie szeregu przykładowych danych wejściowych bez jakiejkolwiek informacji dotyczącej pożądanych czy chociażby tylko oczekiwanych sygnałów wyjściowych. Okazuje się, że odpowiednio zaprojektowana sieć neuronowa potrafi wykorzystać same tylko obserwacje wejściowych sygnałów i zbudować na ich podstawie sensowny algorytm swojego działania – najczęściej polegający na tym, że automatycznie wykrywane są klasy powtarzających się (być może z pewnymi odmianami) sygnałów wejściowych i sieć uczy się zupełnie spontanicznie, rozpoznawać te typowe wzorce sygnałów.

Samouczenie jest bardzo interesujące z punktu widzenia analogii, jakie istnieją między działaniem sieci i ludzkiego umysłu – człowiek też ma zdolność do spontanicznego klasyfikowania napotykanych obiektów i zjawisk, a po dokonaniu stosownej klasyfikacji rozpoznaje kolejne obiekty jako należące do jednej z tych wcześniej poznanych klas. Samouczenie jest także interesujące z punktu widzenia

zastosowań, gdyż nie wymaga żadnej jawnie podawanej do sieci neuronowej zewnętrznej wiedzy – która może być niedostępna – a sieć zgromadzi wszystkie potrzebne informacje i wiadomości zupełnie sama.

Na początku procesu samouczenia wszystkie neurony otrzymują przypadkowe wartości współczynników wagowych. Do takiej przypadkowo zainicjowanej sieci zaczynają napływać sygnały wejściowe. Otrzymawszy te sygnały – wszystkie neurony określają na podstawie składowych tych sygnałów wejściowych oraz swoich wag, swoje sygnały wyjściowe. Sygnały te mogą być dodatnie albo ujemne. Na podstawie sygnałów wejściowych i ustalonych wyjść wszystkie neurony samouczącej się sieci korygują swoje wagi. Podczas korekty wag zachowanie każdego neuronu zależy od tego, jaka była wartość jego sygnału wyjściowego, którym odpowiedział on na pobudzenie. Jeśli sygnał wyjściowy neuronu był silnie pozytywny – wagi zmieniają się w taki sposób, że neuron zbliża się do obiektu, który mu się „podobał”. Oznacza to, że jeśli ponownie zostanie pokazany ten sam sygnał (obiekt) – neuron odnotuje go jeszcze bardziej entuzjastycznie (sygnał wyjściowy będzie miał jeszcze większą dodatnią wartość). Z kolei neurony, które wykazywały w stosunku do aktualnie pokazywanego obiektu stosunek negatywny – będą od niego odpychane w wyniku czego ich wagi skłonią je w przyszłości do jeszcze bardziej negatywnego reagowania na ten typ wejścia. W kolejnym kroku nowe wagi neuronów staje się starymi i po następnym sygnale cały cykl zaczyna się od początku. Jeśli ten proces będzie się dostatecznie długo powtarzał – powstanie skupisko neuronów, które będą wyspecjalizowane w rozpoznawaniu typowego obiektu należącego do tej właśnie grupy. W ten sposób sieć – zupełnie sama – wyspecjalizuje się w rozpoznawaniu wszystkich napotkanych rodzajów obiektów.

Opisany powyżej proces samouczenia jest ogólnym schematem uczenia. Dlatego chcąc się oprzeć na wzorach korzystamy z reguły Hebba.

2.4.2.1. Uczenie wg Hebba

Reguła uczenia Hebba określająca zmiany wartości jednej z wag w kolejnych j krokach uczenia:

$$w_i^{(j+1)} = w_i^{(j)} + \Delta w_i^{(j)}, \tag{2.72}$$

może być napisana jako:

$$\Delta w_i^{(j)} = \eta x_i^{(j)} y_i^{(j)}. \tag{2.73}$$

Wadą tej reguły jest wykładniczy wzrost wag przy wielokrotnej prezentacji takiego samego wymuszenia. Efektem tego jest nasycenie neuronu. Dla uniknięcia takiej sytuacji modyfikuje tę regułę przez wprowadzenie współczynnika zapominania γ.

$$\Delta w_i^{(j)} = \eta x_i^{(j)} y_i^{(j)} - \gamma w_i^{(j)} y_i^{(j)} \tag{2.74}$$

Odpowiedni dobór wartości γ umożliwia powstrzymanie niekontrolowanego wzrostu wag. Lepsze rezultaty uzyskuje się przyjmując modyfikację Oji, zgodnie z którą:

$$\Delta w_i^{(j)} = \eta y_i^{(j)} [x_i^{(j)} - y_i^{(j)} w_i^{(j)}]. \tag{2.75}$$

W ogólności algorytmy uczące Hebba można zaliczyć do uczenia typu korelacyjnego, w którym siła połączenia międzyneuronowego wzrasta, gdy sygnały są w stanie korelacji. Przeciwnym typem uczenia jest uczenie dekorelacyjne **antyhebbowskie**, w którym siła połączenia międzyneuronowego wzrasta wówczas, gdy sygnały są zdekorelowane (jeden sygnał w stanie

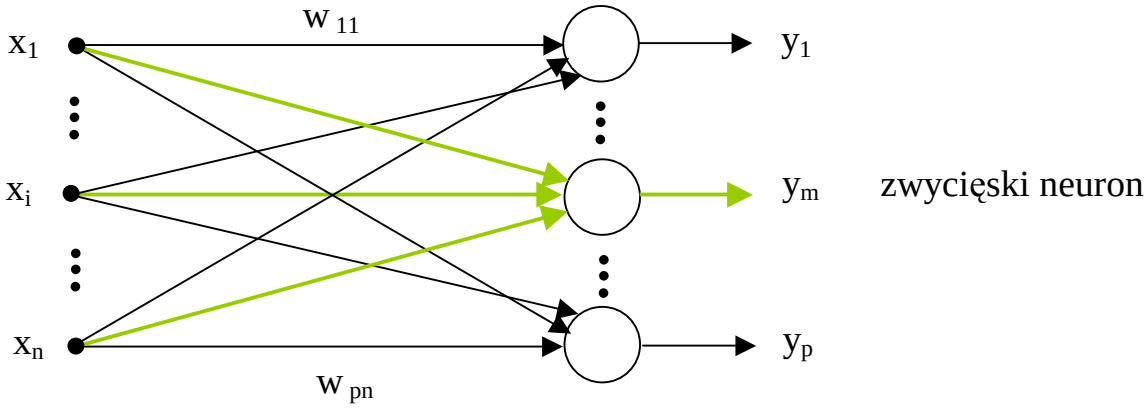
pobudzenia a drugi w stanie zgaszenia). Uczenie antyhebbowskie wyróżnia przeciwny znak uwzględniony przy $\Delta w_i^{(j)}$:

$$\Delta w_i^{(j)} = -\eta x_i^{(j)} y_i^{(j)}. \tag{2.76}$$

W przeciwieństwie do reguły Hebba uczenie antyhebbowskie nigdy nie wprowadza nieograniczonego wzrostu wag i jest stabilne bez żadnych dodatkowych modyfikacji reguły.

2.4.2.2. Uczenie konkurencyjne

Innym podejściem do samouczenia jest tak zwane **uczenie konkurencyjne** (inaczej uczenie **WTA – Winner Takes All**). W uczeniu tego typu neurony współzawodniczą ze sobą, aby stać się aktywnymi (pobudzonymi). W odróżnieniu od uczenia Hebba, gdzie dowolna liczba neuronów mogła być pobudzona, w uczeniu konkurencyjnym tylko jeden neuron może być aktywny, a pozostałe są w stanie spoczynkowym. Grupa neuronów współzawodniczących otrzymuje te same sygnały wejściowe x_i. W zależności od aktualnych wartości wag sygnały wyjściowe neuronów $y_m = \sum_i w_{mi} \cdot x_i$ różnią się między sobą. W wyniku porównania tych sygnałów zwycięża neuron, którego wartość y_m jest największa. Neuron zwycięzca przyjmuje na swoim wyjściu stan 1, a pozostałe (przegrywające) stan 0.



Rys.2.13. Model sieci do uczenia konkurencyjnego [143]

Uczenie to odbywa się z zastosowaniem znormalizowanych wektorów

$$\|\mathbf{W}\| = \|\mathbf{X}\| = \mathbf{1}. \tag{2.77}$$

Aktualizacja wag neuronu zwycięzcy w *j*-tym kroku uczenia odbywa się według reguły Kohonena, która przyjmuje postać:

$$w_{m\,i}^{(j+1)} = w_{m\,i}^{(j)} + \eta[x_j - w_{m\,i}^{(j)}]. \tag{2.78}$$

W efekcie współzawodnictwa neuronów następuje samoorganizacja procesu uczenia. Neurony dopasowują swoje wagi w ten sposób, że przy prezentacji grup wektorów wejściowych zbliżonych do siebie zwycięża zawsze ten sam neuron. W trybie odtworzeniowym, w którym przy ustalonych wartościach wag podaje się na wejście sieci sygnały testujące, neuron poprzez zwycięstwo we współzawodnictwie rozpoznaje swoją kategorię. Układy tego typu są stosowane najczęściej do klasyfikacji wektorów.

Odmianą uczenia konkurencyjnego jest uczenie typu **WTM** (**Winner Takes Most**), w którym neuron wygrywający konkurencję uaktywnia się w sposób maksymalny przyjmując wartość sygnału wyjściowego y_m=1

i umożliwiając częściowe uaktywnienie innych neuronów z sąsiedztwa. Stopień uaktywnienia neuronów z sąsiedztwa zależy od odległości ich wektorów wagowych od wag neuronu wygrywającego.