2021

Metody Sztucznej Inteligencji. 6. Uczenie sieci neuronowych

6. UCZENIE SIECI NEURONOWYCH

WŁODZIMIERZ KASPRZAK

UCZENIE SIECI NEURONOWEJ, SIECI GŁĘBOKIE

Przedstawione zostaną: typowa architektura i metody uczenia sieci neuronowych (wsteczna propagacja błędu, uczenie przez współzawodnictwo, predykcja sekwencji czasowej).

Spis treści

1 Sztuczne sieci neuronowe

Sztuczne sieci neuronowe są dogodnym narzędziem stosowanym tam, gdzie podanie rozwiązania o analitycznej postaci funkcji jest trudne lub niemożliwe. Dzięki algorytmom uczenia sieci na podstawie zadanego zbioru próbek możemy wyznaczyć sieć neuronową dobrze *aproksymującą* nieznaną postać funkcji.

1.1 Sieć perceptronu

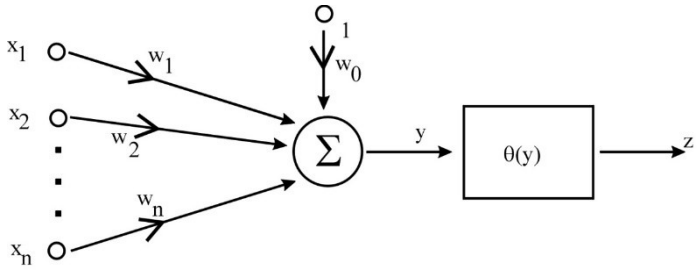
1.1.1 Model neuronu

Powszechnie stosowany model neuronu składa się z dwóch zasadniczych funkcji: liniowej funkcji pobudzenia i zwykle nieliniowej funkcji aktywacji (Rysunek 1). Oznaczmy wejścia jako (x_1,x_2,…,x_n) i wyjście *z*. Funkcja pobudzenia ma postać:

$$y = \sum_{i=1}^{n} (w_i \cdot x_i) - w_0 \cdot 1$$

gdzie w_0 jest wagą dodatkowego wejścia progowego. Funkcja aktywacji (wyjście neuronu):

$$z = \theta(y).$$



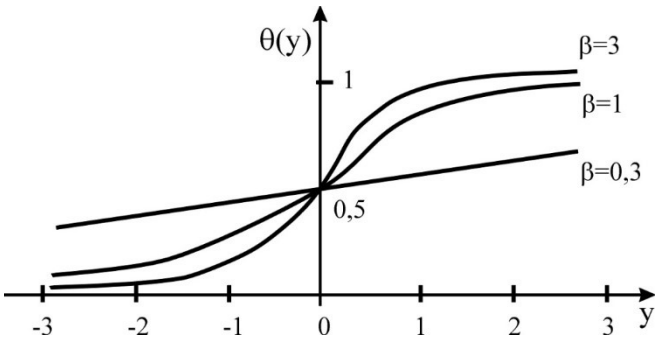
Rysunek 1. Model neuronu (Rosenblatt, 1957)

1.1.2 Funkcja aktywacji neuronu

Typowe funkcje aktywacji to:

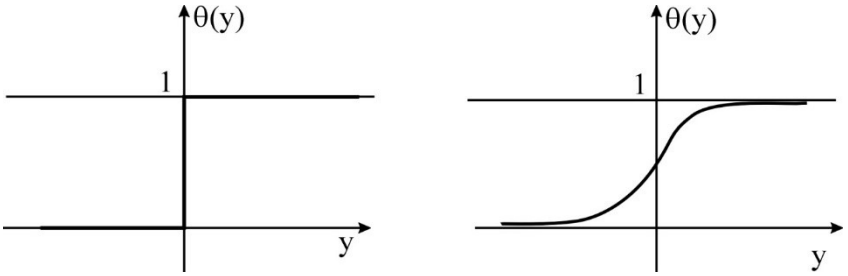
1. Funkcja liniowa, $z = k\,y$, gdzie *k* jest zadanym stałym współczynnikiem.
2. Funkcja skoku jednostkowego: $z = \begin{cases} 1 \text{ , gdy } y_i > y_T \\ 0 \text{ , przeciwnie} \end{cases}$, gdzie y_T jest zadanym progiem.
3. Funkcja *logistyczna* (*sigmoidalna unipolarna*): $\theta(y) = \frac{1}{1+exp(-\beta y)}$, gdzie β jest zadanym parametrem. Wyjście neuronu przyjmuje wartości z przedziału [0, 1].
4. Funkcja *tangens hiperboliczny* (*bipolarna*): $\theta(y) = tgh(\frac{\alpha\, y}{2})$, gdzie α jest zadanym parametrem. Wyjście przyjmuje wartości z przedziału [-1, 1].

Wpływ wartości parametru β na kształt sigmoidalnej funkcji aktywacji ilustruje Rysunek 2.



Rysunek 2. Wpływ parametru β na kształt funkcji sigmoidalnej.

Najczęściej stosuje się funkcje aktywacji o ciągłej pochodnej w całej dziedzinie, np. zamiast funkcji *skoku jednostkowego* stosujemy funkcję *logistyczną* (*sigmoidalną unipolarną*) (Rysunek 3).

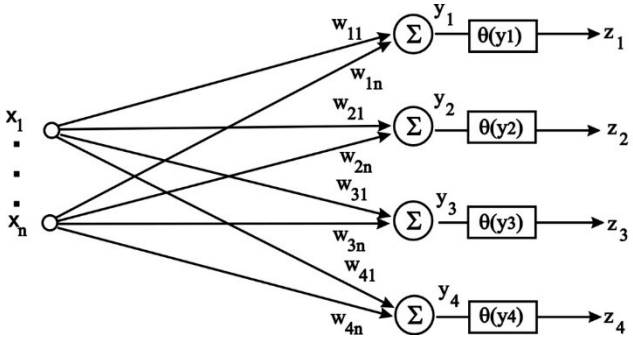


Rysunek 3. Funkcje aktywacji: (lewa strona) funkcja skoku jednostkowego, (prawa strona) funkcja sigmoidalna unipolarna.

Dla *logistycznej* (*sigmoidalne unipolarnejj*) funkcji aktywacji o postaci, $z = \theta(y) = \frac{1}{1+exp(-\beta y)}$, jej pochodna jest postaci: $\frac{\partial z}{\partial y} = \beta \cdot z(1-z)$

1.1.3 Warstwa neuronów

W sieci **jednowarstwowej** neurony ułożone są w jednej warstwie. W sieci typu **perceptron** (sieć **jednokierunkowa**, ang. *„feed-forward”*) sygnały wejściowe są przesyłane od wejścia do wyjścia poprzez połączenia **pobudzające**. Połączenie wektora wejściowego z neuronami warstwy wyjściowej jest zwykle pełne co reprezentuje macierz wag połączeń $\mathbf{W}$ (Rysunek 4), Funkcja pobudzenia neuronów warstwy wyjściowej (w zapisie macierzowym): $\boldsymbol{y} = \boldsymbol{W} \cdot \boldsymbol{x}$



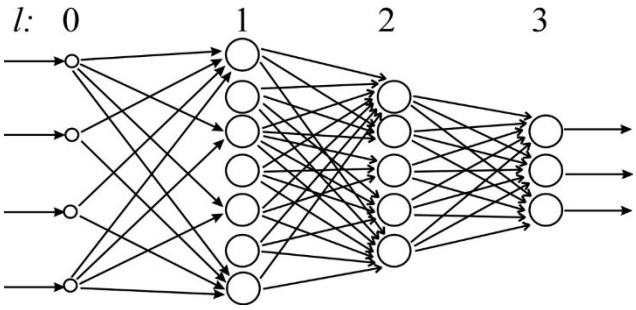
Rysunek 4. Warstwa jednokierunkowa.

1.1.4 Struktura sieci MLP

Wielowarstwowy perceptron (ang. *multilayer perceptron*, MLP) posiada warstwę wejściową, warstwę ukrytą (jedną lub więcej) i warstwę wyjściową (Rysunek 5). Funkcja aktywacji każdej warstwy o indeksie $l(=0,1,2,\ldots)$ **wynosi**:

$$\boldsymbol{z}^{(l)} = \theta(\boldsymbol{W}^{(l)}\boldsymbol{z}^{(l-1)} - \boldsymbol{w}_0{}^{(l)})$$

przy czym $\mathbf{z}(0) = \mathbf{x}$, $\theta(\mathbf{y})$ jest najczęściej nieliniową funkcją aktywacji, a $\mathbf{w}_0(l)$ to wektor wag dodatkowych wejść progowych.



Rysunek 5. Struktura sieci MLP o czterech warstwach

Można pokazać (Cybenko, 1989), że już 3-warstwowy perceptron o sigmoidalnej funkcji aktywacji i n_{hidden} neuronach w warstwie ukrytej, przy $n_{\text{hidden}} \rightarrow \infty$ (dowolnie duża liczba neuronów) może aproksymować dowolne zbiory w przestrzeni $\Re^n$, albo dowolną funkcję ciągłą zdefiniowaną w tej przestrzeni.

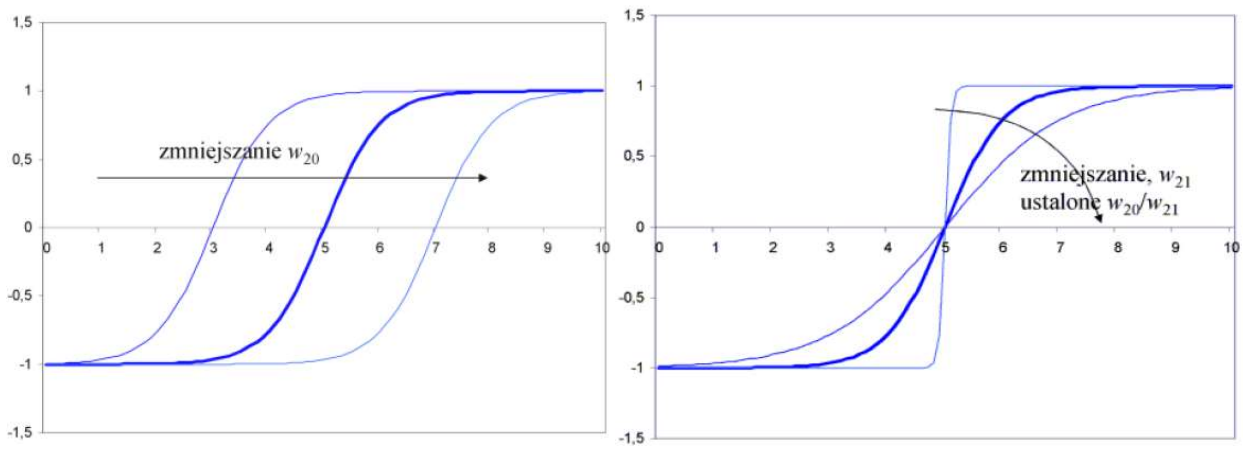
1.1.5 Parametry sieci MLP

Rozważmy sieć o jednym wejściu *x*, dwóch neuronach ukrytych n_1 i n_2 oraz jednym neuronie wyjściowym n_3 o liniowym wyjściu y_3. Nieliniowa funkcja aktywacji $\theta(y)$ jest postaci tangensa hiperbolicznego. Czyli aproksymujemy funkcję f: R→ R. Opisuje ją wzór:

$$z_3 = y_3 = w_{30} + w_{31} \cdot \theta(w_{10} + w_{11}x) + w_{32} \cdot \theta(w_{20} + w_{21}x)$$

Znaczenie parametrów pojedynczego neuronu warstwy ukrytej jest następujące (Rysunek 6):

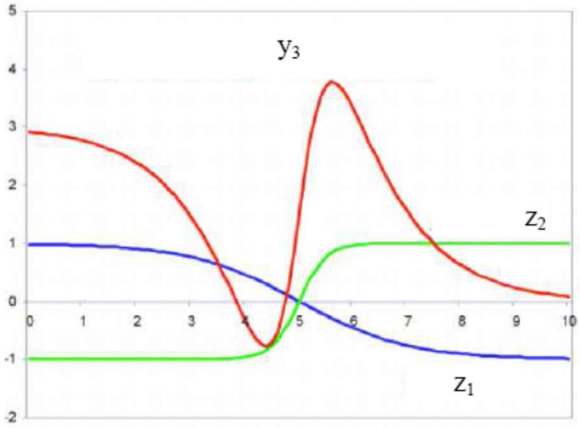
- wagi w_{10}/w_{11}, w_{20}/w_{21} służą do przesuwania wykresu funkcji $\theta(.)$ wzdłuż osi rzędnych
- wagi w_{11}, w_{21} wpływają na „stromość" wykresu funkcji $\theta(y_1)$, $\theta(y_2)$.



Rysunek 6. Wpływ wartości wag warstwy ukrytej na zachowanie się aproksymowanej funkcji

Parametry warstwy wyjściowej

Parametry warstwy wyjściowej służą określeniu stopnia wymieszania wyjść neuronów ukrytych. Im większa wartość wagi w_{31}, wzgl. w_{32} tym większy mnożnik użyty do wykresu wyjścia neuronu 1-szego wzgl. 2-ego (rys. 14.7).



Rysunek 7. Ilustracja wpływu wag warstwy wyjściowej

1.2 Uczenie sieci MLP

1.2.1 Uczenie z nadzorem

Podstawowa reguła uczenia z nadzorem to **reguła Widrow-Hoffa** (reguła „**delta**"). Według niej waga połączenia *j*-tego wejścia z *i*-tym (neuronem) wyjściowym jest wzmacniana proporcjonalnie do różnicy pożądanej i rzeczywistej aktywacji:

$$\Delta w_{ij} = \mu(s_i - z_i)x_j$$

gdzie s_i to pożądana aktywacja i-tego wyjścia.

Niech dla wektora wejść $\mathbf{x}_t$ oczekiwany (prawidłowy) wektor wyjściowy perceptronu wynosi $\mathbf{s}_t$. Miarą jakości sieci jest suma kwadratów błędów wyjść, $V(\mathbf{W})$ (gdzie *błędem* jest różnica między rzeczywistą a żądaną wartością), uśredniona dla zbioru treningowego:

$$V(\mathbf{W}) = \frac{1}{N_\mathrm{T}} \sum_{t \in \mathrm{T}} \sum_i {e_{t,i}}^2 = -\frac{1}{N_\mathrm{T}} \sum_{t \in \mathrm{T}} \sum_i (s_{t,i} - z_{t,i}(y_i(\mathbf{x}_t)))^2$$

1.2.2 Reguła modyfikacji wag

Uczenie perceptronu polega na optymalizacji wartości funkcji błędu V, czyli na poszukiwaniu jej minimalnej wartości metodą *spadku w kierunku gradientu* $\nabla V(\mathbf{W})$ (ang. *steepest gradient descent*). Pozwala to w iteracyjny sposób wyznaczyć wartości wag sieci odpowiadające minimum (globalnemu w przypadku sieci liniowej lub najczęściej lokalnemu – dla sieci nieliniowej). Dla pojedynczej wagi w_{ij} modyfikacja w *t*-tej iteracji jest postaci:

$$w_{ij}(t+1) = w_{ij}(t) - \eta(t) \frac{\partial V(\boldsymbol{W}, t)}{\partial w_{ij}}$$

gdzie η oznacza parametr zwany *współczynnikiem uczenia*.

1.2.3 Warstwa wyjściowa

W kolejnej iteracji t wyznaczana jest chwilowa wartość błędu, $V_t = \left(z_i^{(L)} - s_i\right)^2$ i wyznaczane są gradienty tej funkcji względem aktualnych wartości wag. Niech L będzie indeksem warstwy wyjściowej. W tej warstwie gradient aktualnego błędu V w iteracji t może być wyrażony bezpośrednio jako funkcja korekty wyjścia, pochodnej funkcji aktywacji i wartości wejść tej warstwy:

$$\frac{\partial V(\boldsymbol{W}_t)}{\partial {w_{ij}}^{(L)}} = \frac{\partial V(\boldsymbol{W}_t)}{\partial z_i^{(L)}} \cdot \frac{\partial z_i^{(L)}}{\partial y_i^{(L)}} \cdot \frac{\partial y_i^{(L)}}{\partial {w_{ij}}^{(L)}} =$$
$$= 2 \cdot \left(z_i^{(L)} - s_i\right) \cdot {z'}_i^{(L)}\left(y_i^{(L)}\right) \cdot {x_j}^{(L)}$$

gdzie $z'_i(y_i) = \frac{dz_i}{dy_i}$ oznacza pochodną funkcji aktywacji neuronu, a ${x_j}^{(L)}$ oznacza *j*-te wejście dla tej warstwy (czyli wyjście warstwy poprzedniej ${z_j}^{(L-1)}$). Stąd wynika reguła modyfikacji wagi w warstwie wyjściowej L sieci :

$${w_{ij}}^{(L)}(t+1) = {w_{ij}}^{(L)}(t) - \eta \cdot \left(z_i^{(L)} - s_i\right) \cdot {z'}_i^{(L)}\left(y_i^{(L)}\right) \cdot {x_j}^{(L)}$$

Oznaczmy **korektę wartości pobudzenia** neuronu wyjściowego przez

$$\delta_i^{(L)} = -\left(z_i^{(L)} - s_i\right) \cdot {z'}_i^{(L)}\left(y_i^{(L)}\right),$$

tzn. że aktualny błąd dla *i*-tego wyjścia przeskalowany zostaje przez ujemną pochodną funkcji aktywacji (indeks iteracji pominęliśmy). Reguła **modyfikacji wagi** przyjmuje wtedy postać:

$${w_{ij}}^{(L)}(t+1) = {w_{ij}}^{(L)}(t) + \eta \delta_i^{(L)} {x_j}^{(L)}$$

1.2.4 Warstwy ukryte

J-te wyjście z poprzedniej warstwy, ${z_j}^{(L-1)} = {x_j}^{(L)}$, jest połączone wektorem wag $\mathbf{w}_j^{(\mathrm{L})}$ i funkcją błędu aktywacji z wyjściem warstwy wyjściowej ${z_j}^{(L)}$. Należy to uwzględnić przy określaniu pochodnej błędu względem wag warstwy $(L\text{-}1)$. Korekta pobudzenia neuronu warstwy ukrytej wynosi:

$$\delta_j^{(L-1)} = \sum_i w_{ij}^{(L)} \delta_i^{(L)} \cdot {z_j^{(L-1)}}'(y_j)$$

Zauważmy, że dla obliczenia gradientu funkcji aktualnego błędu $V(\mathbf{W}_t)$ względem wag warstwy ukrytej sieci wystarczy informacja o korekcie pobudzenia neuronów warstwy, bezpośrednio po niej następującej. Reguła modyfikacji wag warstwy ukrytej o indeksie, $l = 1,...,L\text{-}1$, jest postaci:

$$w_{ij}^{(l)}(t+1) = w_{ij}^{(l)}(t) + \eta \cdot \delta_i^{(l)} \cdot x_j^{(l)}$$

Z tej przyczyny podstawowa zasada modyfikacji wag sieci MLP w procesie uczenia sieci jest nazywana **regułą wstecznej propagacji błędu** (ang. *error backpropagation rule*) – błąd propaguje się w kierunku od wyjścia do wejścia sieci. Ogólna postać reguły wstecznej propagacji błędu dla macierzy wag warstwy l wynosi:

$$\Delta \boldsymbol{W}^{(l)} = \eta \cdot \Delta^{(l)} \cdot \left[x^{(l)}\right]^T, \qquad x^{(1)} = x$$

dla $l = 1, \ldots, L$; gdzie $\Delta^{(l)}$ jest wektorem korekcji pobudzenia neuronów l-tej warstwy.
Obliczenie korekty wag rozpoczyna się od ostatniej warstwy $(l = L)$ i przemieszcza się wstecz warstwa po warstwie, aż zakończy się na warstwie o indeksie $l=1$. Podczas kroku modyfikacji wag propagowane są „wstecz" wartości korekty, obliczone początkowo dla najwyższej warstwy L, a następnie dla kolejnych od końca warstw ukrytych:

$$\Delta^{(l-1)} = (\mathbf{W}^{(l)})^T \Delta^{(l)} .* \left[\frac{\partial \theta}{\partial y}\right]$$

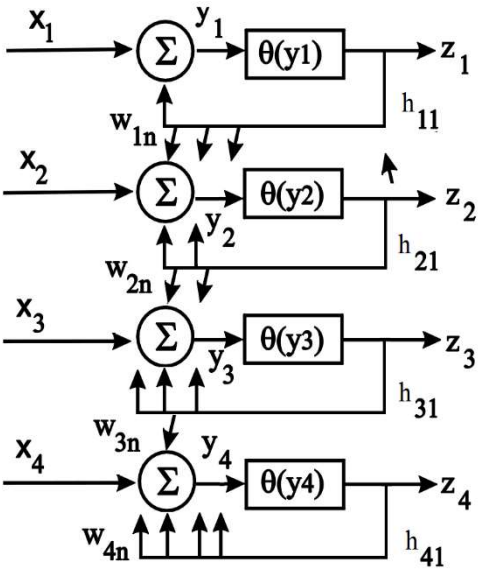
Uwaga: symbol **.*** we wzorach macierzowych na korektę wag oznacza mnożenie elementu pierwszego wektora z odpowiednim elementem drugiego wektora (element-po-elemencie).
W szczególności przy logistycznej funkcji aktywacji wartości korekty, dla $l = L,...,1$, obliczamy jako:

$$\Delta^{(L)} = (\mathbf{s} - \mathbf{z}^{(L)}) .* \left[1 - \mathbf{z}^{(L)}\right] .* \mathbf{z}^{(L)}$$

$$\Delta^{(l-1)} = (\mathbf{W}^{(l)})^T \Delta^{(l)} .* \left[1 - \mathbf{z}^{(l-1)}\right] .* \mathbf{z}^{(l-1)}$$

1.3 Sieci rekurencyjne

W **sieciach rekurencyjnych** istnieją połączenia wyjść z wejściami neuronów – są to tzw. połączenia **hamujące** (ang. *inhibitory links*). Aktywacje w sieciach rekurencyjnych ustalają się w wyniku relaksacji (proces dynamicznych zmian). Ponieważ połączenia hamujące realizują sprzężenie zwrotne pomiędzy wyjściem a wejściem sieci rekurencyjnej, więc po jej pobudzeniu stan sieci może oscylować aż do chwili osiągnięcia stanu stabilnego (rys. 14.8).



Rysunek 8. Przykład struktury sieci rekurencyjnej

1.3.1 Liniowa sieć rekurencyjne

Specyficznym przypadkiem jest liniowa **sieć rekurencyjna. Zapiszmy** wagi połączeń **hamujących** między neuronami warstwy w postaci macierzy wag **H**. Funkcja pobudzenia neuronów takiej sieci ma postać:

$$\mathbf{y} = \mathbf{x} - \mathbf{H} \cdot \mathbf{y}$$

W stanie stabilnym liniowa sieć rekurencyjna realizuje funkcję równoważną funkcji **liniowej sieci jednokierunkowej** o postaci:

$$\mathbf{y} = (\mathbf{I} + \mathbf{H})^{-1} \cdot \mathbf{x}$$

1.3.2 Przykłady sieci rekurencyjnych

- Sieć jednowarstwowa Hopfielda – pamięć asocjacyjna. Odtwarza ona pełny „najlepszy" wzorzec przy jej pobudzeniu niepełnym wzorcem.
- Maszyna wielowarstwowa Boltzmana – o binarnych wyjściach i ze stochastyczną regułą aktywacji tzw. „symulowanego wychładzania" (ang. *simulated annealing*) zależną od energii własnej i temperatury sieci. Rozwiązuje ona problemy globalnej optymalizacji.
- Samą warstwę sieci rekurencyjnej możemy połączyć z warstwą jednokierunkową dla wyznaczenia:
  1. wejścia o największej wartości (ang. WTA - *Winner Takes All*);
  2. K wejść o największej wartości (ang. kWTA)

1.4 Sieci typu WTA

1.4.1 WTA

Sieć WTA („zwycięzca bierze wszystko”) jest to jednowarstwowa sieć rekurencyjna, która:
1. Posiada nieliniową aktywację w postaci funkcji RELU;
2. Jej wagi hamujące są postaci:

$$h_{im} = \begin{cases} -\varepsilon & gdy \quad i \neq m \\ +1 & gdy \quad i = m \end{cases}$$

co oznacza, że wyjście neuronu dodaje się do sumy na jego wejściu a odejmuje się wartości wyjść wszystkich pozostałych neuronów przemnożone przez parametr ε . W dynamicznym procesie oscylacji po kolei wszystkie wyjścia o mniejszych wartościach zostają zmniejszone do wartości zero i pozostaje jedno wyjście niezerowe odpowiadające wejściu o największej wartości.
Uwaga: wszystkie wartości wejściowe powinny być nieujemne.

1.4.2 Sieć kWTA

Sieć WTA może być też zastosowana do znalezienia k wejść o największych wartościach (problem kWTA, „k zwycięzców”) spośród N wejść, gdzie $k = 1, 2, \ldots, N-1$. W tym przypadku potrzebna jest druga warstwa, która zlicza liczbę wyjść o zerowej wartości. W momencie, gdy liczba zerowych wyjść osiągnie N-k, pozostaje k wyjść o dodatnich wartościach i proces oscylacji należy przerwać (Tabela 1).

Tabela 1. Implementacja programowa sieci kWTA

Program w Matlabie

```
function [z, iternum] = kWTA( u, maxiter, k )
% Argumenty: u -  wektor „n” liczb (sygnały wejściowe dla sieci),
%                 maxiter – maksymalna liczba iteracji,
%                 k – liczba szukanych zwycięzców.
% Zwracany wynik: z – wektor „n” liczb (sygnały wyjściowe) –
% niezerowa wartość na i-tym wyjściu wskazuje, że odpowiednie i-te
% wejście należy do zbioru „k zwycięzców”.
                [c, n] = size(u);
                if (k> (n-1)) % błędna wartość argumentu k
                   iternum = 0;    z = u;
                  return;
                end
                eps = 1.0/(n+k);
% Inicjalizacja
z = u;   newz = z;
iternum = maxiter; % maksymalna liczba iteracji
for i=1 : maxiter %
   sumuvec = sum(z);     zeronum = 0;
```

```
  for j=1:n
    newz(j) = u(j) + z(j) - eps *(sumuvec - z(j)); % reguła modyfikacji
    if newz(j) < 0  % nieliniowość RELU
      newz(j) = 0;        zeronum = zeronum +1;
    end
  end
  z = newz; % synchroniczna modyfikacja wszystkich wyjść
  % Sprawdź aktualną liczbę niezerowych wyjść u(j):
  if zeronum >= (n-k)
    iternum = i;
    break; % zakończ
  end
end
```

1.5 Neuronowa klasteryzacja

Sieć Kohonena jest siecią dwuwarstwową, w której wagi pierwszej warstwy uczone są metodą „w warunkach konkurencji" (ang. *competitive learning*). Pierwsza warstwa jest liniowa i jednokierunkowa (ang. *feed-forward*) a jej wagi docelowo wyznaczają środki klastrów danych wejściowych. Druga warstwa jest rekurencyjna i służy jedynie w procesie uczenia do wyznaczenia wyjścia pierwszej warstwy o największej wartości (problem WTA).

W pierwszej warstwie każde z wyjść ***i*** jest połączone z każdym wejściem ***j***, zaś funkcja ***i***-tego wyjścia wynosi: $y_i = \mathbf{w}_i^T \mathbf{x}$

$$z_i = f(y_i) = \frac{y_i}{|\mathbf{w}_i|}$$

Wyjścia muszą zostać znormalizowane długością wektora wag:

W procesie uczenia dla każdej próbki wejściowej wybierany jest więc neuron wyjściowy o najwyższej aktywacji i jego wagi $\mathbf{w}_l$ są korygowane „w kierunku" aktualnego wektora wejściowego $\mathbf{x}(t)$:

$$\mathbf{w}_l(t+1) = \mathbf{w}_l(t) + \eta_l [\mathbf{x}(t) - \mathbf{w}_l(t)].$$

Jednocześnie umożliwia on częściową aktywację neuronów ze swojego „sąsiedztwa" w stopniu zależnym od odległości ich wektorów wag $\mathbf{w}_k$ od wag neuronu wygrywającego:

$$\mathbf{w}_k(t+1) = \mathbf{w}_l(t) + \eta_k G(k, l, \mathbf{x}(t)) [\mathbf{x}(t) - \mathbf{w}_k(t)].$$

Funkcja sąsiedztwa $G(k, l, \mathbf{x}(t))$ wyznacza stopień „uczenia" (wartość z przedziału [0,1]) sąsiada o indeksie k zwycięskiego neuronu l w iteracji t. Dla oryginalnej sieci Kohonena zachodzi:

$$G(k,l,x(t)) = \begin{cases} 0 & gdy \quad k \neq l \\ +1 & gdy \quad k = l \end{cases}$$

co oznacza, że do modyfikacji nie dopuszczano żadnych neuronów sąsiednich (strategia *„winner takes all"*).

1.6 Głębokie sieci neuronowe

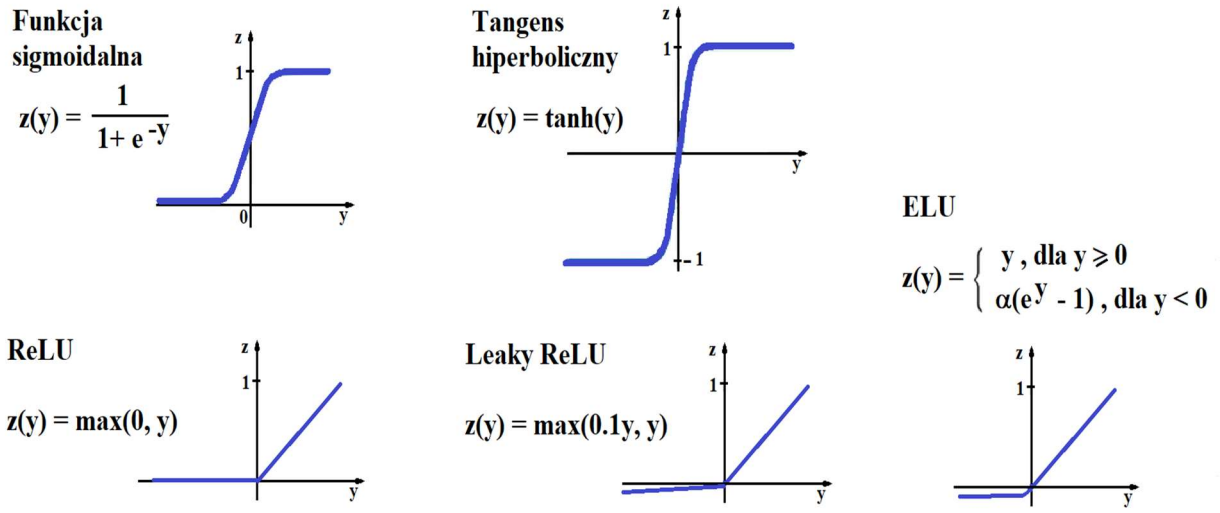
1.6.1 Architektury sieci głębokich

Typowe architektury głębokich sieci neuronowych (ang. *deep neural networks*, DNN):

1. Klasyfikacja wzorców poprzez sieć wielowarstwową („w pełni połączoną") z warstwą wyjściową typu „soft-max" lub klasyfikacja siecią MLP;
2. Splotowe sieci neuronowe (ang. *convolutive neural networks*, CNN) do wyznaczania cech i klasyfikacji obrazów 2D;
3. Sieci splotowe R-CNN i jej modyfikacje;
4. Rekurencyjne sieci neuronowe – w tym sieci LSTM do modelowania sekwencji czasowych i predykcji.
5. Sieć antagonistyczna;
6. Sieci korelacyjne i syjamskie;
7. Głębokie auto-enkodery.

1.7 Funkcje aktywacji

Typowe funkcje aktywacji (funkcja generująca wyjście neuronu na podstawie pobudzenia od wejść i wag połączeń) (rys. 14.4).



Rys. 14.9 Przykłady funkcji aktywacji stosowanych w głębokich sieciach neuronowych

1.8 Uczenie sieci z warstwą softmax

Warstwa „**softmax**" posiada funkcję aktywacji, która zapewnia normalizację wyjść sieci „w pełni połączonej" stanowiącej klasyfikator o K klasach. Normalizacja polega na tym, że suma wartości wyjść zawsze wynosi 1. Pozwala to interpretować wyjścia jako wartości prawdopodobieństwa klas dla jednej zmiennej losowej.

Funkcja wejściowa (liniowa) k-tego neuronu y_k i jego funkcja aktywacji „softmax" z_k mają odpowiednio postać:

$$y_k = \sum_{i=1}^{d} w_{ki} x_i + b_k$$

$$z_k = \frac{\exp(y_k)}{\sum_{j=1}^{K} \exp(y_j)}$$

Funkcja aktywacji:

Dla sieci o warstwie wyjściowej „softmax" jako miara błędu w procesie uczenia sieci stosowana jest **entropia krzyżowa** (ang. *cross-entropy*).

Rozpatrzmy problem dwu-klasowy (K_1, K_2). Wyjście „z" stanowi aproksymację prawdopodobieństwa klasy „1" dla wejścia $\boldsymbol{x}$ przy wagach $\boldsymbol{W}$ (zmienna s jest indeksem klasy):

$$z \sim P(K_1 \mid \boldsymbol{x}) = P(s = 1 \mid \boldsymbol{x}; \boldsymbol{W})$$

$$(1 - z) \sim P(K_2 \mid \boldsymbol{x}) = P(s = 0 \mid \boldsymbol{x}; \boldsymbol{W})$$

Rozkład Bernoulliego zmiennej losowej s o 2 realizacjach ma postać:

$$P(s \mid \boldsymbol{x}; \boldsymbol{W}) = z^{s} \cdot (1 - z)^{1-s}$$

Optymalizacja wag $\boldsymbol{W}$ polega na minimalizacji negacji logarytmu z prawdopodobieństwa zmiennej losowej s (jest to **entropia krzyżowa**):

$$E = -\ln P(s \mid \boldsymbol{x}; \boldsymbol{W}) = -\left(s \ln z + (1 - s) \ln(1 - z) \right)$$

Kryterium uczenia głębokich sieci neuronowych z warstwą softmax polega na minimalizacji **funkcji błędu** wyrażonej przez entropię krzyżową, czyli błędu zachodzącego pomiędzy rozkładem prawdopodobieństwa oczekiwanym a rzeczywistym.

Proces uczenia takich sieci również opiera się na algorytmie wstecznej propagacji błędu, podobnie jak w przypadku sieci MLP.

Ciekawą własnością gradientu entropii krzyżowej jest to, że zanika w nim zależność od gradientu funkcji aktywacji z=f(y):

$$\frac{\partial E}{\partial z} = -\frac{s}{z} + \frac{1 - s}{1 - z} = \frac{-(1 - z)s + z(1 - s)}{z(1 - z)} = \frac{(z - s)}{z(1 - z)}$$

$$\boxed{\frac{\partial E}{\partial w_i}} = \frac{\partial E}{\partial z} \cdot \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial w_i} = \frac{(z - s)}{z(1 - z)} \cdot z(1 - z) \cdot x_i = \boxed{(z - s)\, x_i}$$

2 Pytania

1. Omówić model neuronu.
2. Omówić pojęcie wielowarstwowego perceptronu MLP.
3. Jak wpływa liczba warstw i wagi sieci na jakość aproksymacji?
4. Omówić podstawowy algorytm uczenia sieci MLP.
5. Przedstawić sieć rekurencyjną i jej zastosowanie jako kWTA (*k-winners-take-all*).
6. Przedstawić funkcje aktywacji stosowane w głębokich sieciach neuronowych.
7. Przedstawić funkcję „softmax" i kryterium uczenia sieci głębokiej z „warstwą softmax".

3 Bibliografia

Podstawowa

1. S. Russel, P. Norvig: *Artificial Intelligence. A modern approach.* Prentice Hall, 2002 (2nd ed.), 2013 (3d ed.). [Rozdziały 1, 2]
2. M. Flasiński: *Wstęp do sztucznej inteligencji*. Wydawnictwo Naukowe, PWN, 2011. [Rozdziały 2, 14]
3. S. Osowski: Sieci neuronowe do przetwarzania informacji. Wydawnictwo OW Politechniki Warszawskiej, wyd. IV, 2020 (Rozdziały: 2,3,6,10, 14).