Sieci Neuronowe

Wykład 3

1. Zdolności uogólniania sieci, weryfikacja procesu uczenia
2. Perceptron raz jeszcze.
3. Nieliniowe sieci wielowarstwowe.

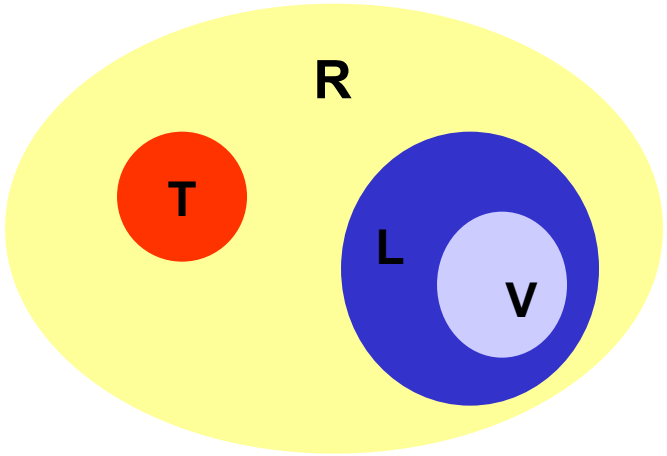
wykład przygotowany na podstawie.

S. Osowski, "Sieci Neuronowe w ujęciu algorytmicznym ", Rozdz. 3, PWNT, Warszawa 1996
S. Osowski, „Sieci neuronowe do przetwarzania informacji", Oficyna Wydawnicza PW, Warszawa 2000.
R. Tadeusiewicz, "Sieci Neuronowe", Rozdz. 4. Akademicka Oficyna Wydawnicza RM, Warszawa 1993.

Zdolności uogólniania sieci neuronowej

Podstawową cechą sieci neuronowej jest jej zdolność do uogólniania, a więc generowania właściwego rozwiązania dla danych, które nie pojawiły się w zestawie danych uczących.

Sieć zostaje poddana uczeniu na zbiorze L z bieżącym sprawdzeniem stopnia uczenia na zbiorze V. Zdolność odtworzenia zbioru L przez sieć jest miarą zdolności zapamiętania danych uczących, natomiast zdolność do generowania właściwych rozwiązań dla danych należących do zbioru T, na których sieć nigdy nie była trenowana, jest miarą zdolności uogólniania. Zakłada się że dane tworzące zarówno zbiór L jak i zbiór T są typowymi reprezentantami zbioru danych.

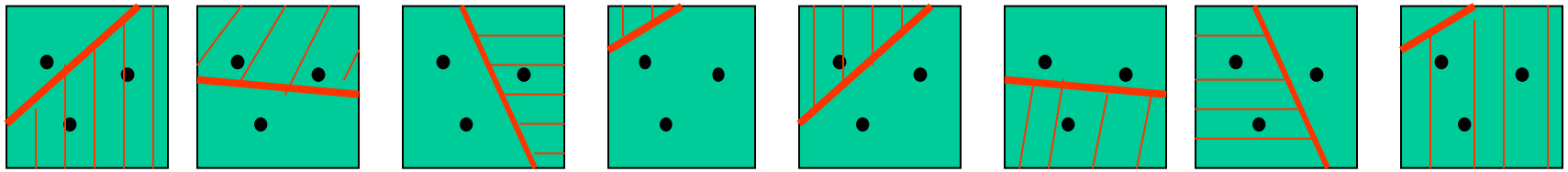


R – zbiór danych wejściowych
T - zbiór testujący (testing)
L - zbiór uczący (learning)
V - zbiór danych sprawdzających (validation)

Miara Vapkina-Chervonenkisa

Ilościowa miara uogólniania jest pojęciem trudnym do zdefiniowania i jest oparta na zależnościach statystycznych odnoszących się do zbiorów. Podstawową wielkością jest tu miara Vapkina-Chervonenkisa, zwana w skrócie *VCdim*. Miara *VCdim* systemu została zdefiniowana jako liczebność największego zbioru S danych wzorców, dla których system może zrealizować wszystkie możliwe 2^n dychotomii zbioru S (podział zbioru na dwie części przy pomocy lini).

Na przykład *VCdim* dla neuronu o dwóch wejściach wynosi n=3. Można wykazać, że zbiór złożony z trzech danych uczących jest największym zbiorem, w którym można przeprowadzić podział na dwie liniowo separowalne grupy na 2^3 sposobów.



Miara Vapkina-Chervonenkisa

Zwiększenie o jeden rozmiaru próbek uczących powoduje, że 2 neurony nie są w stanie zrealizować wszystkich 2^4 podziałów liniowo separowanych. W ogólności dla neuronu o N wejściach (wektor x N-elementowy) miara VCdim wynosi N+1. Innymi słowy,

Miara VCdim dla sieci rozwiązującej problem klasyfikacji binarnej oznacza maksymalną liczbę danych uczących, które mogą zostać bezbłędnie odtworzone we wszystkich możliwych konfiguracjach.

Błąd uczenia sieci

Niech $v_L(W)$ oznacza błąd uczenia sieci, czyli częstotliwość wystąpienia błędu klasyfikacji podczas procesu uczenia, a $P(W)$ – średnie prawdopodobieństwo wystąpienia błędnej klasyfikacji podczas uczenia. Oznaczając przez ε wartość dopuszczalnego błędu wykazano, że

$$\text{Prob}\{|P(W) - v_L(W)| > \varepsilon\} \rightarrow 0$$

jeśli liczba próbek uczących $p \rightarrow \infty$, przy czym Prob{} oznacza prawdopodobieństwo zdarzenia.

Błąd uczenia sieci

Niech α oznacza prawdopodobieństwo zdarzenia

$$\sup |P(W) - v_L(W)| > \varepsilon$$

prawdopodobieństwo to zostało oszacowane w postaci

$$\alpha = (2pe/h)^h \exp(-\varepsilon^2 p)$$

przy czym e jest liczba Eulera, p-liczbą próbek uczących, a h aktualną wartością VCdim.

Błąd uczenia sieci

Oznaczając przez ε_0 wartość ε spełniającą relacje przy przy zadanej wartości α otrzymuje się

$$\varepsilon_0 = \text{sqrt}\{ h/p [\ln(h/2p) + 1] - 1/p \ln(\alpha) \}$$

Wartość ε_0 reprezentuje przedział ufności.

Przedział ten jest funkcją aktualnej miary VCdim, liczby próbek uczących p oraz wartości α i nie zależy od błędu uczenia sieci $v_L(W)$. Miara ta obowiązuje tylko w przypadku dopuszczenia dużych wartości P(W).

Błąd uczenia sieci

Przy wymaganiu małych wartości P(W) zmodyfikowana definicja przedziału ufności (oznaczona przez ε_1) zależy również od błędu uczenia $v_L(W)$ i przybiera postać.

$$\varepsilon_1 = \varepsilon_0^2 \left(1 + \mathrm{sqrt}\{1 + v_L(W)/\varepsilon_0^2\} \right)$$

Na podstawie zdefiniowanych przedziałów ufności można stwierdzić, że w ogólności, przy małym poziomie błędu uczącego $v_L(W)$, średnie prawdopodobieństwo wystąpienia błędu klasyfikacji spełnia nierówność

$$P(W) < v_L(W) + \varepsilon_1$$

Przy bardzo dużych błędach uczenia $v_L(W)$, dokładniejszą estymatę średniego prawdopodobieństwa wystąpienia błędu klasyfikacji określa relacja

$$P(W) < v_L(W) + \varepsilon_0$$

Błąd uogólniania sieci

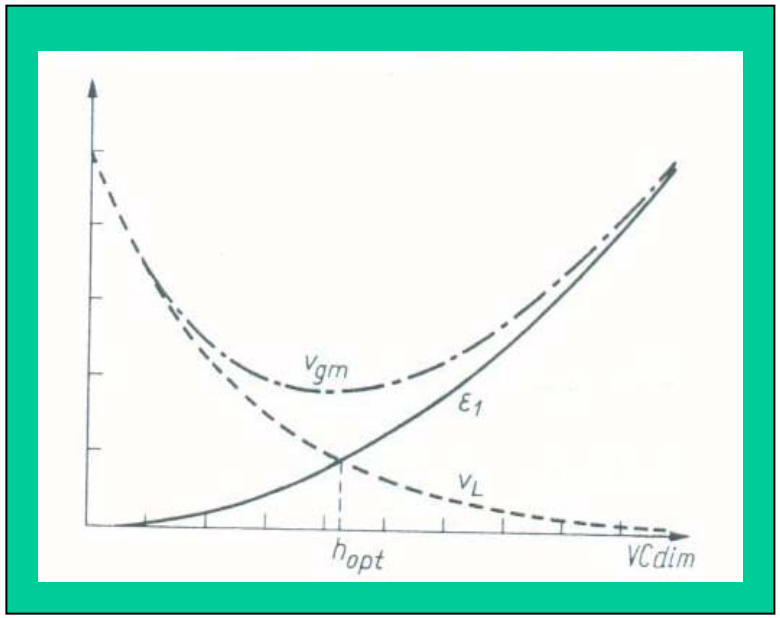
Podobnie jak błąd uczenia, definiuje się błąd uogólniania $v_g(W)$ jako częstotliwość wystąpienia błędu podczas testowania zbioru na danych testujących.

Przy liczbie próbek uczących $p > h$ (h – aktualna wartość *VCdim* sieci poddanej uczeniu) z prawdpodobienstwem $(1-\alpha)$ błąd uogólnienia jest mniejszy niż $v_{gm}(W)$, $v_g(W) \leq v_{gm}(W)$, przy czym

$$v_{gm}(W) = v_L(W) + \varepsilon_1$$

Zdolności uogólniania sieci neuronowej

Przy stałej liczbie próbek p i wzrastającej wartości miary *VCdim* błąd uczenia $v_L(W)$ maleje monotonicznie, a przedział ufności ε_1 rośnie. W efekcie maksymalny błąd uogólniania osiąga minimum. Zakres *VCdim* < h_{opt} odpowiada nadmiarowości danych bieżących względem aktualnej wartości *VCdim*. *Zakres VCdim* > h_{opt} odpowiada zbyt małej liczbie danych uczących przy aktualnej wartości *VCdim*.



Zdolności uogólniania sieci neuronowej

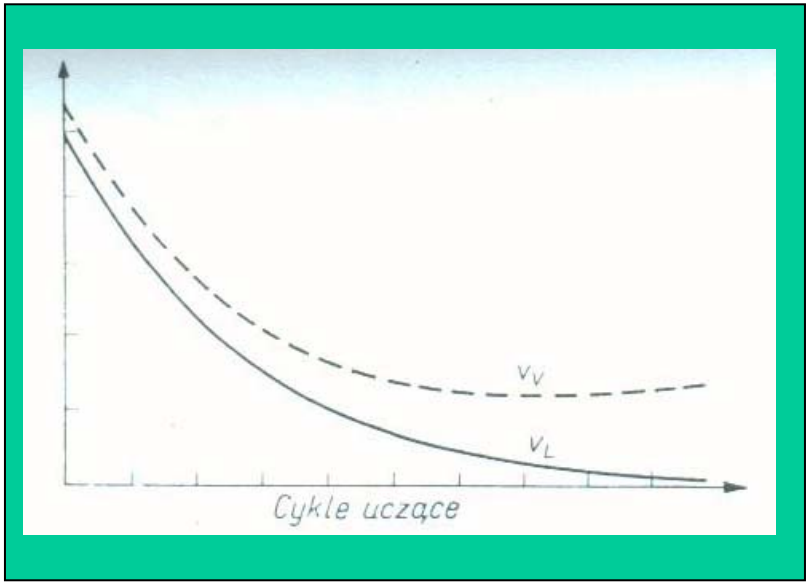
W przypadku ustalonej wartości *VCdim* błąd uogólniania zależy w istotnym stopniu od liczby próbek uczących. Dla zapewnienia odpowiednio małej wartości tego błędu liczba próbek musi spełniać odpowiednie proporcje względem *VCdim*. Dla każdego rodzaju sieci jest to oddzielny problem. Szczególnie jaskrawo występuje on w przypadku sieci wielowarstwowej, gdzie liczba wag jest zwykle bardzo duża w stosunku do liczby neuronów.

Trudność: oszacowanie wartości VCdim dla dowolnej sieci.

W praktyce, dla uzyskania dobrych zdolności uogólniania sieci należy ograniczać liczbę neuronów ukrytych oraz powiązań miedzy neuronowych, jak również stosować takie metody wstępnego przetwarzania danych, które umożliwiają zmniejszenie wymiarowości wektora wejściowego sieci. Każdy z tych czynników, pośrednio lub bezpośrednio, wpływa na zmniejszenie efektywnej liczby wag sieci neuronowej.

Cykla uczące i błąd veryfikacji

W ogólnym przypadku wraz z upływem czasu uczenia błąd uczenia $v_L(W)$ maleje i błąd testowania $v_V(W)$ również (przy ustalonej wartości liczby próbek uczących p oraz miary *VCdim*).



Od pewnego momentu błąd veryfikacji pozostaje stały, natomiast błąd uczenia nadal maleje. W ostatnich fazach procesu uczenia nieregularności w danych odbiegające od cech charakterystycznych danego procesu zaczynają odgrywać role i powodują wzrost błędu testowania.

Zdolności uogólniania sieci neuronowej

Tendencje te (przeuczenie) jest tym silniejsze im większe nadmiarowości wag występują w sieci. Te “niepotrzebne” wagi dopasowywują się do nieregularności danych uczących, traktując je jako cechę główną. Ważne jest aby kontrolować proces uczenia przez przeplatanie go z procesem testowania, monitorując jak daleko jest zaawansowany proces uczenia.

Sam proces uczenia powinien być powiązany ze sprawdzaniem zdolności do uogólniania, a więc powinien zawierać “fazę uczącą” i “fazę sprawdzającą”.

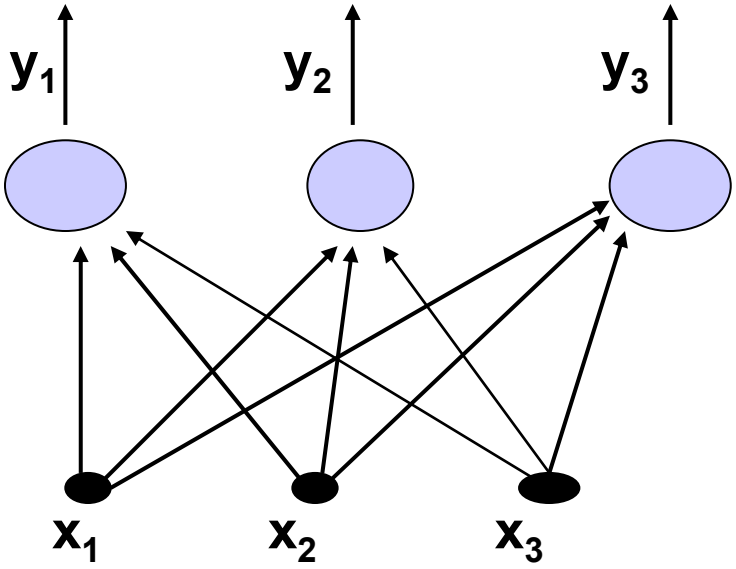
Proces uczenia kontynuuje się do chwili uzyskania minimum funkcji celu lub dopóki błąd testowania nie zacznie wzrastać (wskazując na przeuczenie).

PERCEPTRON … raz jeszcze …

Siecią neuronową, która odegrała historycznie bardzo istotną rolę był

PRECEPTRON

koncepcja w której wprowadzono nieliniowy element przetwarzający informację. Wprowadzenie nieliniowości było uzasadnione, biologiczne układy faktycznie są nieliniowe.



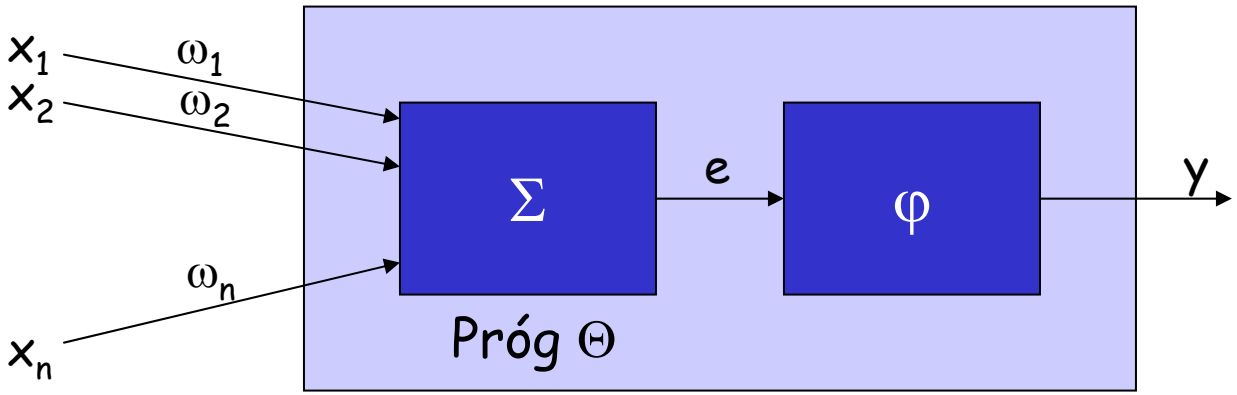
Perceptron prosty ma z definicji jedną warstwę neuronów

PERCEPTRON … raz jeszcze …

Nieliniowy element przyjmowany w sieciach neuronowych może być opisany równaniem.

$$y = \varphi(e)$$

gdzie $\varphi(e)$ jest wybraną funkcją nieliniową a sygnał e odpowiada łącznemu pobudzeniu neuronu.



PERCEPTRON … raz jeszcze …

Łączne pobudzenie neuronu możemy uznać za zgodne z formułą przyjmowaną uprzednio dla ADALINE

$$e = \sum_{i=1}^{n} \omega_i x_i$$

lub uzupełnioną o dodatkowo o stały składnik (bias)

$$e = \sum_{i=1}^{n} \omega_i x_i + \omega_0$$

Aby uprościć zapis, przyjmijmy, że oprócz $<x_1, x_2, \ldots, x_n>$ mamy również element x_0, co pozwoli formalnie zapisać:

$$e = \sum_{i=0}^{n} \omega_i x_i$$

lub wektorowo: $e = W^T X$

PERCEPTRON ... raz jeszcze ...

Formułę nieliniowego pobudzenia możemy też zapisać następująco, np. w postaci *sumy kumulowanej*, której postać w j-tym kroku symulacji może być wyznaczoną ze wzoru:

$$e^{(j+1)} = e^{(j)} + \sum_{i=1}^{n} \omega_i^{(j)} x_i^{(j)}$$

albo *funkcji majoryzacji*:

$$e = \sum_{i=0}^{n} \mu_i$$

gdzie μ_i jest miarą efektywności i-tego wejścia wyznaczaną ze wzoru:

$$\mu_i = 1 \quad \text{gdy} \quad \omega_i x_i > 0$$
$$\mu_i = 0 \quad \text{gdy} \quad \omega_i x_i \leq 0$$

PERCEPTRON … raz jeszcze …

Inne możliwe postacie:

maximum

$$e = MAX_i\ \omega_i\ x_i$$

minimum

$$e = MIN_i\ \omega_i\ x_i$$

produktowa

$$e = \Pi_i\ \omega_i\ x_i$$

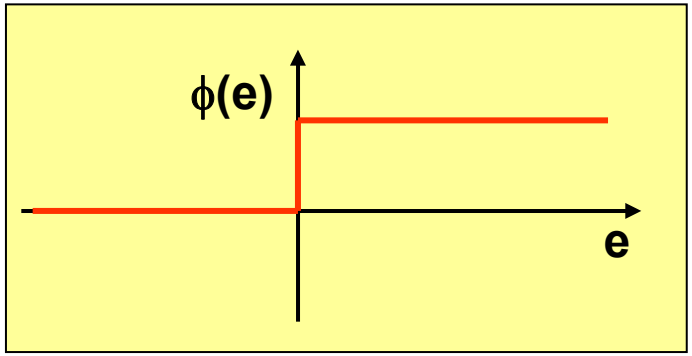
Te i inne funkcje scalające wejściowe sygnały x_i w łączne wypadkowe pobudzenie *e*, używane są w perceptronie jedynie jako wstępny etap przetwarzania informacji w neuronie.

PERCEPTRON ... raz jeszcze ...

O specyficznych własnościach *perceptonu* decyduje funkcja φ określającą nieliniowy związek miedzy sygnałem wypadkowego pobudzenia neuronu *e*, a jego odpowiedzią y.
W klasycznym perceptonie funkcja φ ma postać progowa:

$\varphi(e) = 1$ gdy $e \geq 0$;

$\varphi(e) = 0$ gdy $e < 0$;



Ta postać ma szereg wad ale jest łatwa do wyprowadzenia pewnych intuicji. Ponieważ sygnał wyjściowy przyjmuje wartość (y=1 lub y=0) , może być rozważany w kategoriach określonej *decyzji*.
Możliwa jest też interpretacja oparta na logice matematycznej, *prawda* lub *fałsz*.
Percepton może być interpretowany jako układ realizujący pewną funkcję logiczną, a więc automat skończony.

PERCEPTRON … raz jeszcze …

Dygresja:

Zależnie od postaci przyjętej funkcji $\varphi(e)$ sygnał y można rozpatrywać jako

binarny	$y \in \{ 0, 1 \}$
bipolarny	$y \in \{ -1, 1 \}$

Pozornie różnica jest nieistotna, trywialne przeskalowanie. Może mieć jednak poważne konsekwencje ponieważ punkty należące do zbioru { 0,1} są wierzchołkami jednostkowego hiperszescianu w R^n, natomiast punkty należące do zbioru {-1,1} leżą na powierzchni jednostkowej sfery R^n.
W n-wymiarowej przestrzeni sześcian i sfera różnią się w sposób zasadniczy.
Porównajmy objętości:

objetość szescianu: $V_s = a^n$

objetość kuli: $V_k = \pi^{n/2} / (n/2)!\ r^n$ gdy n jest parzyste

$V_k = 2^n\ \pi^{(n-1)/2}((n-1)/2)! / n!\ r^n$ gdy n jest nieparzyste

PERCEPTRON … raz jeszcze …

Tak więc dla jednostkowego boku a, objętość sześcianu jest stała $V_s = 1$, podczas gdy objętość kuli o jednostkowym promieniu r, $V_k \to 0$ dla $n \to \infty$.

Wszystkie punkty sfery są oczywiście jednakowo odległe od jej środka (odległością jest promień sfery), natomiast dla sześcianu, narożniki są odległe od środka o sqrt(n/2) (odległość rośnie).
Sześcian coraz bardziej przypomina "jeża".

W większych wymiarach..... należy dość ostrożnie podchodzić do intuicji geometrycznych.

Czasami warto jest przejść do układu w ktorym neuron przyjmuje wartość {-1, +1}. Wtedy sieć staje się podobna do układu magnetycznego, w którym momenty magnetyczne atomów mogą mieć dwa przeciwne kierunki. W opisie takich sieci można stosować metody z teorii układów magnetycznych.

Prosty przypadek deterministyczny

Zacznijmy od najprostszego przypadku deterministycznego:

$$\varphi(e) = 1$$

oraz przyjmijmy że odpowiedź wyjściowa jest pewnym wektorem którego składowe przyjmują wartości ± 1.
Wówczas

$$Y = \mathrm{sgn}\,(W \otimes X)$$ (sgn oznacza "znak")

Pożądane jest aby

$$\mathrm{sgn}\,(W \otimes X) = Z$$

A więc wektor wag musi być tak dobrany (uczenie) aby rzut wzorca X^{μ} na ten wektor mial taki sam znak jak Z^{μ}.
Granica miedzy dodatnimi a ujemnymi rzutami na kierunek wektora W jest płaszczyzną $W \otimes X$ przechodzącą przez początek układu współrzędnych, prostopadłą do wektora W.

Własności nieliniowych sieci wielowymiarowych

Przyjmując interpretację progowej funkcji $\phi(e)$ jako funkcji rozdzielającej przestrzeń wejściowych sygnałów X na obszar wyróżniony, w którym y=1, oraz na resztę – należy stwierdzić, że przy przyjęciu najczęściej rozważanej reguły scalania wejściowych sygnałów w postaci

$$e = \sum_{i=1}^{n} \omega_i x_i$$

podział ten formułuje granica mająca postać hiperpłaszczyzny.
Istotnie, jeśli $\varphi(e) = 1$ gdy $e \geq 0$; oraz $\varphi(e) = 0$ gdy $e < 0$; to obszar w którym neuron podejmuje decyzje y=1 ogranicza powierzchnia e=0, czyli twór o równaniu

$$\sum_{i=1}^{n} \omega_i x_i = 0$$

Dla n=2 jest to równanie lini prostej, dla n=3 – równanie płaszczyzny, a dla n > 3 twór nazywany prawidłowo *rozmaitością liniową stopnia n-1*, a popularnie traktowany jako płaszczyzna w n-wymiarowej przestrzeni czyli w skrócie *hiperpłaszczyzna*.

.

Własności nieliniowych sieci wielowymiarowych

Możemy interpretować działanie neuronu budującego perceptron jako dyskryminatora liniowego.

Może on zrealizować te wszystkie odwzorowania, w których wystarczy oddzielenie podobszaru przestrzeni X mającego formę otwartej podprzestrzeni ograniczonej hiperpłaszczyzną.

Proces uczenia, polegający zawsze na zmianie wartości współczynników ω_i , pozwala ustalić graniczną hiperpłaszczyznę w dowolnym położeniu, nie pozwala jednak na zmianę charakteru realizowanego odwzorowania, niezależnie od tego jak długo by się go uczyło.

Separowalność liniowa

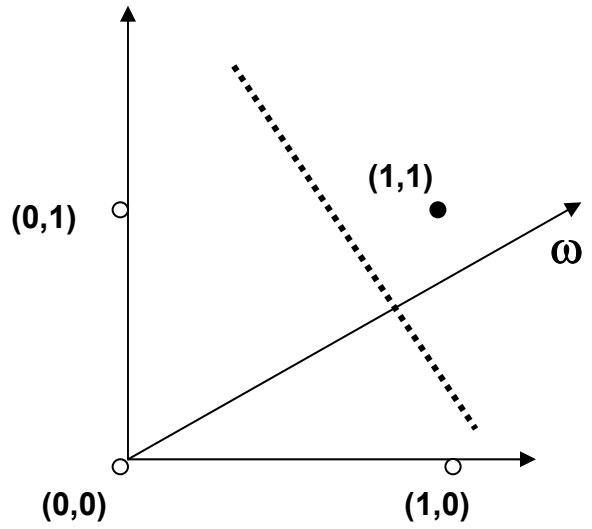
Co sie stanie jezeli nie istnieje taka plaszczyzna?

Wtedy zadanie nie może być rozwiązane – sieć nie może osiągnąć funkcji celu niezależnie od sposobu jej uczenia. Warunkiem rozwiązania za pomocą perceptronu prostego z jednostkowymi progami jest wymaganie aby dany problem był *liniowo separowalny*.

$$Y = \mathrm{sgn}(\omega_1 x_1 + \omega_2 x_2 - \omega_0)$$

Funkcja logiczna: **AND**

x_1	x_2	Y
0	0	-1
0	1	-1
1	0	-1
1	1	1



$\omega_1 = 1$
$\omega_2 = 2$
$\omega_0 = 1.5$

Separowalność liniowa

Przykład dla którego brak jest liniowej separowalności: funkcja **XOR**

$-\omega_1 - \omega_2 < \omega_0$

$\omega_1 - \omega_2 > \omega_0$

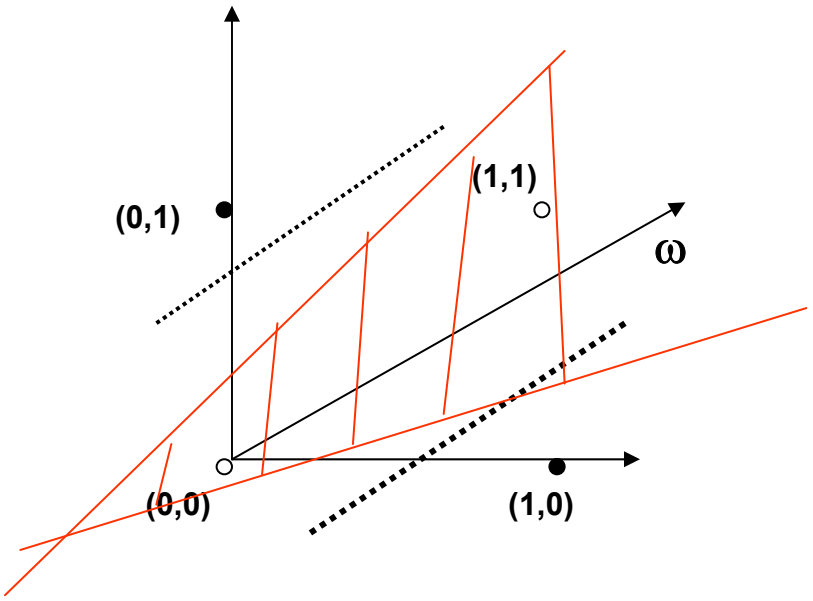
$\omega_1 + \omega_2 < \omega_0$

$-\omega_1 + \omega_2 > \omega_0$

Nie istnieje rozwiązanie dla takiego układu równań.

Funkcja logiczna: **XOR**

x_1	x_2	Y
0	0	- 1
0	1	+1
1	0	+1
1	1	- 1



Własności nieliniowych sieci wielowymiarowych

Nierozwiązywalne zadanie: “problem XOR”

Percepton ***nie może*** się nauczyć realizacji odwzorowania

$$y = x_1 \oplus x_2$$

gdzie operator $\oplus$ oznacza alternatywę wyłączającą (eXclusive OR).

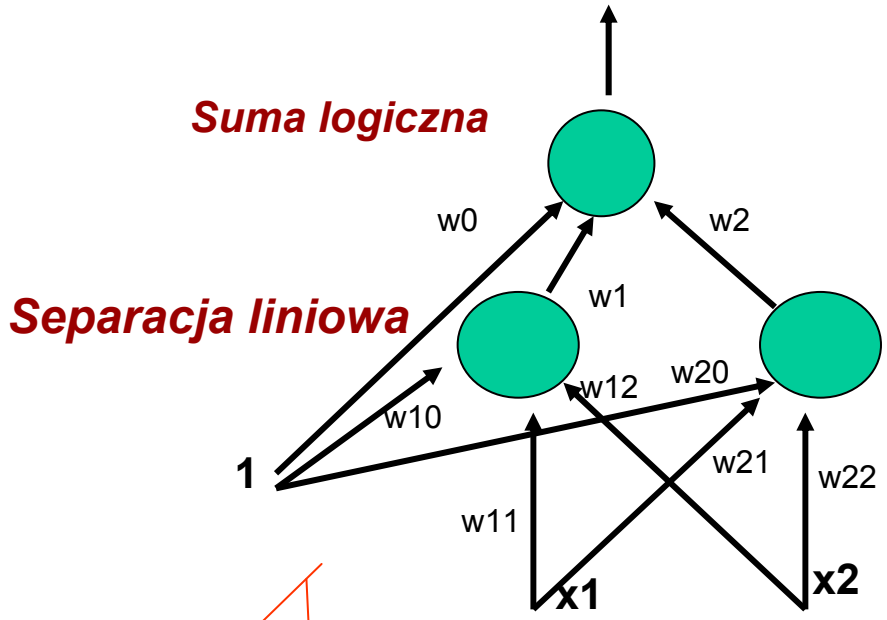
Kilkuwarstwowa sieć:

Jednak, czego nie potrafi zrobić jeden neuron, może zrobić kilkuwarstwowa sieć, ponieważ dla nieliniowych neuronów dodanie nowych warstw istotnie poszerza zakres odwzorowań, które sieć potrafi zrealizować.

Separowalność dla sieci dwuwarstwowej

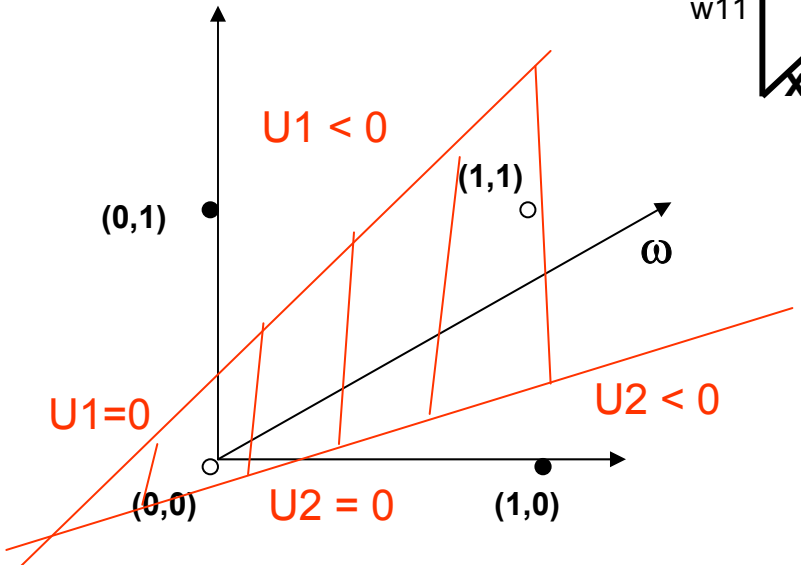
Funkcja **XOR:**
rozwiązuje problem nieliniowej separowalności przez złożenie dwóch separowalności liniowych

$U1 = w11\ x1 + w12\ x2 + w10$
$U2 = w21\ x1 + w22\ x2 + w20$



Funkcja logiczna: **XOR**

x_1	x_2	Y
0	0	- 1
0	1	+1
1	0	+1
1	1	- 1

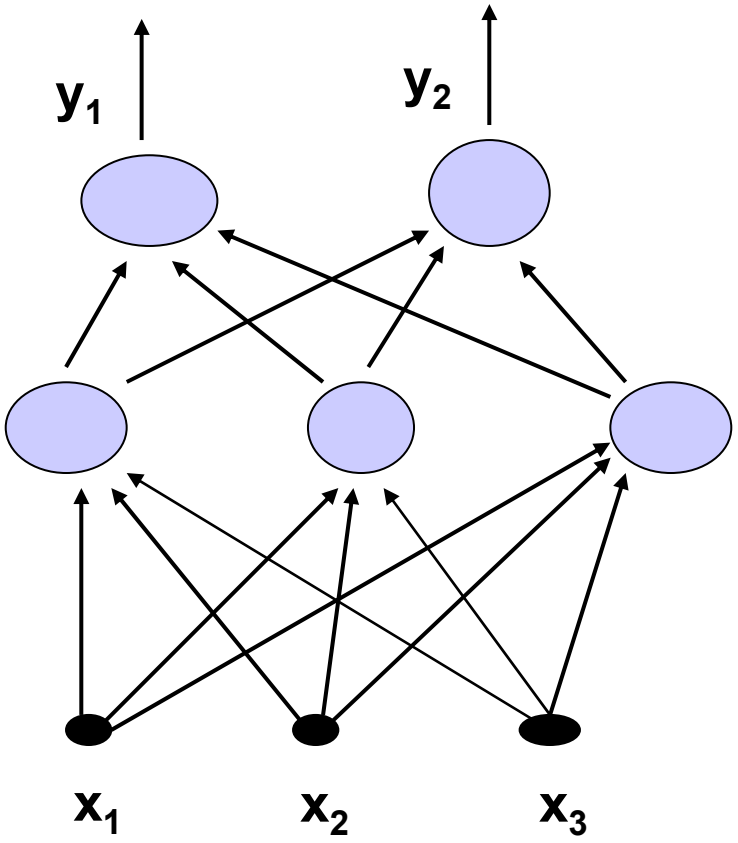


Neuron w warstwie ukrytej realizuje separowalność liniową, neuron w warstwie wyjściowej wykonuje odpowiednią kombinację liniową, np. sumę logiczną

Własności sieci wielowarstwowych perceptronowych

Rozważmy przykładową *sieć dwuwarstwową* :

Pierwsza warstwa, złożona z k neuronów otrzymujących sygnały wejściowe X, dzieli przestrzeń X tych sygnałów za pomocą k oddzielnych hiperpłaszczyzn. Powstaje w ten sposób układ 2k liniowo rozdzielnych obszarów, które sygnalizowane są przez odpowiednie zestawy 0 i 1 jako wartości sygnałów neuronów pierwszej warstwy.



Własności nieliniowych sieci wielowymiarowych

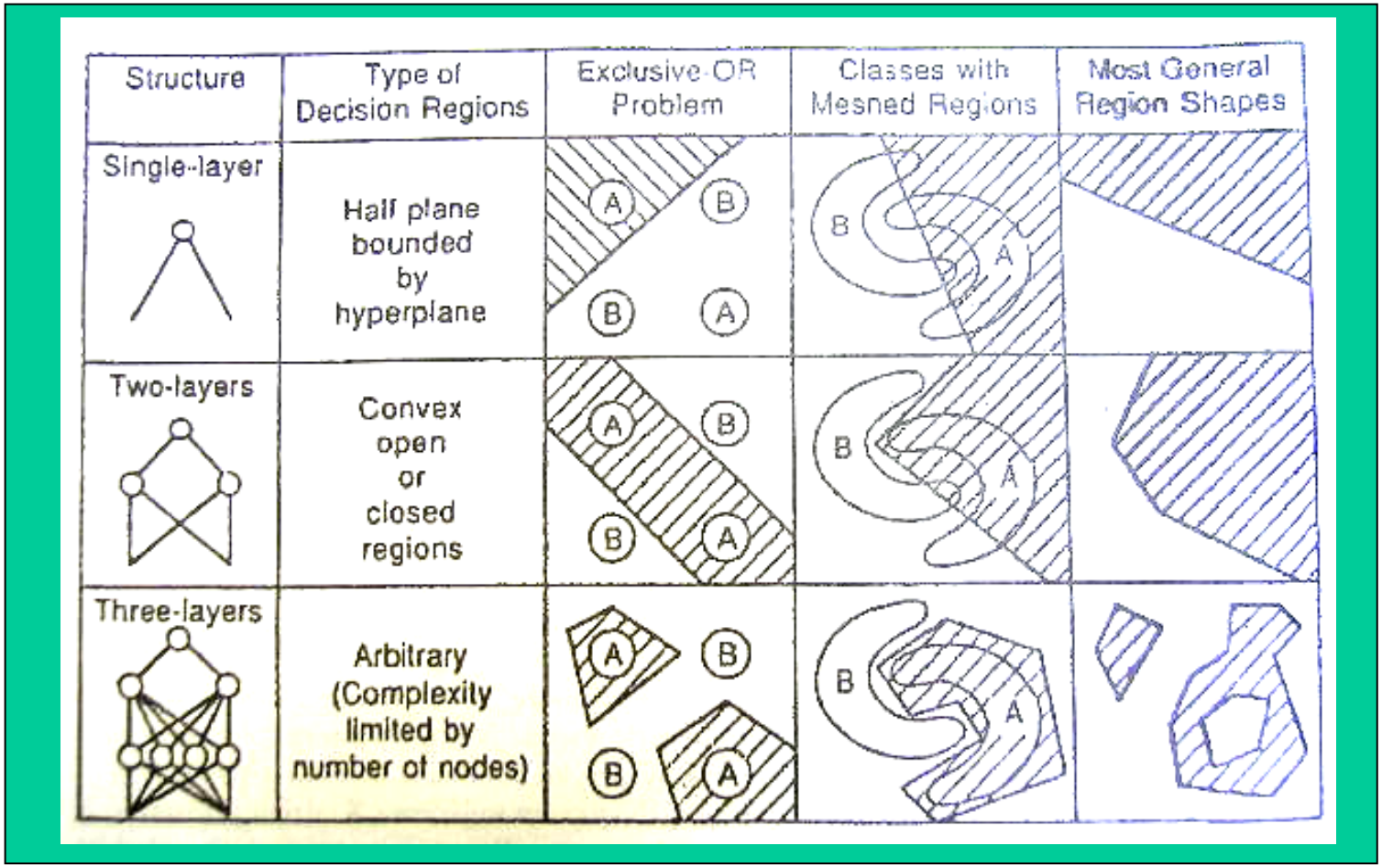
Sygnały te podawane są z kolei na wejścia neuronów drugiej warstwy, które dokonują klasyfikacji zestawów tych sygnałów według zasady: sygnał wyjściowy neuronu drugiej warstwy ma wartość 0 lub 1 w zależności od tego, jaki podzbiór neuronów pierwszej warstwy sygnalizuje 0, a jaki 1. W efekcie neurony drugiej warstwy mogą rozpoznawać (sygnalizować) pojawienie się wektorów wejściowych X zawartych w pewnych ograniczonych obszarach przestrzeni X.

Obszary te nie muszą być już równoważne do całej podprzestrzeni X, ponieważ możliwe jest sygnalizowanie bardziej złożonego podobszaru, ograniczonego z wielu stron fragmentami wielu hiperpłaszczyzn.

Własności nieliniowych sieci wielowymiarowych

Sieć dwuwarstwowa nie pozwala jeszcze rozpoznać dowolnego podobszaru przestrzeni X, ponieważ łatwo sprawdzić, że obszary sygnalizowane przez neurony drugiej warstwy musza być *wypukłe* oraz *jednospójne* (simpleksy). Jest to dość istotne ograniczenie. Aby się od niego uwolnić należy wprowadzić trzecią warstwę neuronów. Dopiero w rezultacie dołączenia trzeciej warstwy możliwe jest utworzenie *dowolnych* obszarów.

Własności nieliniowych sieci wielowymiarowych



Za pomocą nieliniowej sieci neuronowej o przynajmniej trzech warstwach można zrealizować *dowolne odwzorowanie, wiążące w całkowicie dowolny sposób wejściowe sygnały X z wyjściowymi sygnałami sieci.*

Formy nieliniowości neuronu

Funkcja wiążąca łączne pobudzenie neuronu *e* z jego sygnałem wyjściowym y

$$y = \varphi(e)$$

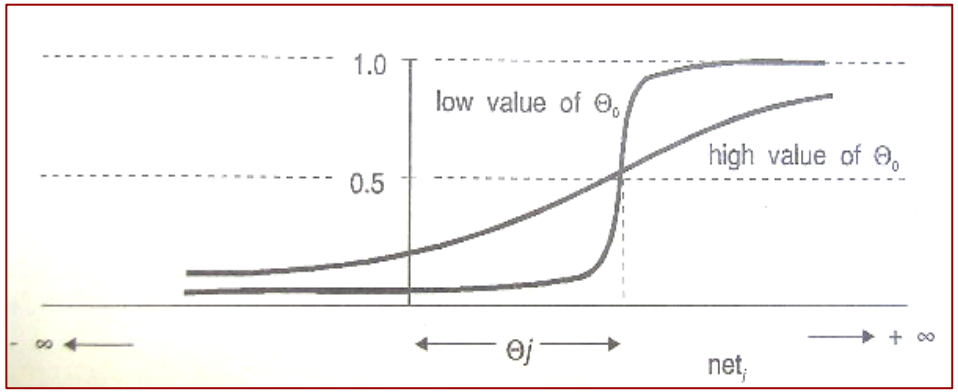
Funkcja ta może mieć różne postacie, dotychczas omawialiśmy postać progową.

Sigmoidalna funkcja wywodząca się z funkcji logistycznej.

$$d\varphi / de = y (1 - y)$$

Ważną własnością funkcji logistycznej jest fakt, że zbiór jej wartości należy do otwartego zbioru $y \in (0,1)$, co oznacza, że wartości 0, 1, mające istotne znaczenie przy niektórych interpretacjach funkcjonowania sieci neuronowych są nieosiągalne.

Przebieg funkcji logistycznej dla dwóch wartości parametrów β (oznaczenie na rysunku θ = β



Formy nieliniowości neuronu

Funkcja tangens hiperboliczny:

$$y = \tanh (\beta e)$$

który można rozpisać jako

$$y = \frac{\exp (\beta e) - \exp (-\beta e)}{\exp (\beta e) + \exp (-\beta e)}$$

Przy zastosowaniu tej funkcji $y \in (0,1)$
Zaletą tej funkcji jest prosta formuła określająca pochodną tej funkcji w zależności od jej wartości

$$d \varphi / de = (1 + y) (1 - y)$$

Funkcja sinus:
chętnie stosowana, można doformułować do przedziału zamkniętego [-1,1]:

$$y = \begin{cases} -1 & \text{gdy } e < -\pi/2 \\ \sin(\beta e) & \text{gdy } -\pi/2 < e < \pi/2 \\ 1 & \text{gdy } e > \pi/2 \end{cases}$$

Ta postać funkcji jest szczególnie przydatna przy budowie sieci dokonującej transformaty Fouriera wejściowego sygnału.

Formy nieliniowości neuronu

Niekiedy nieliniowość ma postać nie różniczkowalną, przydatną w praktycznych zastosowaniach, ale kłopotliwą do teoretycznej analizy. Z bardziej znanych postaci można wymienić:

Funkcje signum:

$$y = \begin{cases} 1 & \text{gdy } e > 0 \\ 0 & \text{gdy } e = 0 \\ -1 & \text{gdy } e < 0 \end{cases}$$

Funkcja perceptonowa:

$$y = \begin{cases} e & \text{gdy } e > 0 \\ 0 & \text{gdy } e \le 0 \end{cases}$$

Funkcja BSB
(Brain State in a Box)

$$y = \begin{cases} 1 & \text{gdy } e > 1 \\ e & \text{gdy } 1 > e > -1 \\ -1 & \text{gdy } e < -1 \end{cases}$$

Zmodyfikowana funkcje signum:

$$y = \begin{cases} 1 & \text{gdy } e > 0 \\ -1 & \text{gdy } e \le 0 \end{cases}$$

Funkcje signum:

$$y = \begin{cases} 1 & \text{gdy } e > 0 \\ 0 & \text{gdy } e = 0 \\ -1 & \text{gdy } e < 0 \end{cases}$$

Funkcja SPR
(Spatio-Temporal Pattern Recognition)

$y^{(j+1)} = y^{(j)} + A [-a y^{(j)} + b e^{+}]$

gdzie "funkcja ataku"

$$A[u] = \begin{cases} u & \text{gdy } u > 0 \\ \gamma u & \text{gdy } u \le 0 \end{cases}$$

zapis e^{+} oznacza

$$e^{+} = \begin{cases} e & \text{gdy } e > 0 \\ 0 & \text{gdy } e \le \end{cases}$$

Funkcja skoku jednostkowego:

$$y = \begin{cases} 1 & \text{gdy } e > 0 \\ 0 & \text{gdy } e \le 0 \end{cases}$$

Funkcja BAM
(Bidirectorial Associative Memory)

$$y^{(j+1)} = y^{(j)} = \begin{cases} 1 & \text{gdy } e > 0 \\ & \text{gdy } e = 0 \\ -1 & \text{gdy } e < 0 \end{cases}$$

Powyższe funkcje są "inżynierskie": opis który pozwala na wygodną analizę matematyczną, łatwą realizację techniczną (perceptron) lub wygodne modelowanie w formie programu symulacyjnego (signum).

Uczenie nieliniowego neuronu

Rozważmy problem uczenia nieliniowych sieci neuronowych.
Dla uproszczenia analizujemy wyłącznie regułę DELTA w jej podstawowej postaci.

$$U = < \; <X^{(1)} , z^{(1)} >, < X^{(2)} , z^{(2)} > , ... , <X^{(N)} ,z^{(N)} > \quad >$$

Formułę uczenia opieramy na regule minimalizacji funkcjonału błędu średniokwadratowego:

$$Q = \tfrac{1}{2} \sum_{j=1}^{N} (z^{(j)} - y^{(j)})^2$$

gdzie

$$y^{(j)} = \phi \left(\sum_{j=1}^{N} \omega_i^{(j)} x_i^{(j)} \right)$$

Uczenie nieliniowego neuronu

Rozkładając funkcjonał błędu na elementy składowe związane z poszczególnymi krokami procesu uczenia

$$Q = \Sigma\, Q(j)$$ gdzie $$Q(j) = \tfrac{1}{2}\,(\, z(j) - (y(j)\,)2$$

Możemy zgodnie z gradientową strategią procesu uczenia zapisać algorytm zmian czynników wag

$$\omega_i(j+1) - \omega_i(j) = \Delta\omega_i(j) = -\,\eta\;\partial Q(j) / \partial\omega_i$$

Uczenie nieliniowego neuronu

Analogiczny wzór wyprowadzono wcześniej dla sieci ADALINE, jednak treść tego wzoru jest w tym wypadku bogatsza ze względu na nieliniową funkcję $\phi(e)$.

$$\partial Q^{(j)} / \partial \omega_i = \partial Q^{(j)} / \partial y_i^{(j)} \; \partial y_i^{(j)} / \partial \omega_i = \partial Q^{(j)} / \partial y_i^{(j)} \; \partial y_i^{(j)} / \partial e^{(j)} \; \partial e^{(j)} / \partial \omega_i$$

łatwo możemy obliczyć:

$$\partial Q^{(j)} / \partial y_i^{(j)} = - (z^{(j)} - y^{(j)}) = - \delta^{(j)}$$

$$\partial e^{(j)} / \partial \omega_i = x_i^{(j)}$$

Uczenie nieliniowego neuronu

Problem może być natomiast z wyrażeniem

$$\partial y_i^{(j)} / \partial e^{(j)} = \partial \phi(e) / \partial e^{(j)}$$

gdzie $\phi(e)$ nie zawsze jest różniczkowalne.

Ostateczny wzór, na podstawie którego prowadzi się proces uczenia ma postać

$$\Delta \omega_i^{(j)} = - \eta \, \delta^{(j)} \, \partial \phi(e) / \partial e^{(j)} \, x_i^{(j)}$$

Uczenie nieliniowego neuronu

Dość chętnie (bezkrytycznie) stosuje się w rozważaniach funkcje logistyczną

$$y = \phi(e) = 1/(1 + \exp(-\beta e))$$

która ma łatwą postać pochodnej,

$$\partial\phi(e) / \partial e^{(j)} = y^{(j)} (1 - y^{(j)})$$

Ostateczny wzór dla funkcji logistycznej może być zapisany w prostszej postaci

$$\Delta\omega_i^{(j)} = - \eta (z^{(j)} - y^{(j)}) (1 - y^{(j)}) y^{(j)} x_i^{(j)}$$

Powyższy algorytm uczenia jest możliwy do bezpośredniego zastosowania jedynie w przypadku *sieci jednowarstwowej*.

Uczenie sieci nieliniowej

Dla *sieci wielowarstwowych*, które mają istotnie szersze możliwości przetwarzania informacji niż sieci jednowarstwowe, omawiany poprzednio wzór nie daje się zastosować. Dla *warstw wewnętrznych* nie ma możliwości bezpośredniego określenia oczekiwanych (wymaganych) wartości sygnałów wejściowych $z^{(j)}$, a tym samych określenia wartości błędu $\delta^{(j)}$. Rozważając ciąg

$$U = < <X^{(1)} , Z^{(1)} >, < X^{(2)} , Z^{(2)} > , ... , <X^{(N)} ,Z^{(N)} > >$$

mamy do dyspozycji n-wymiarowe wektory wejściowe X oraz k-wymiarowe wektory wyjściowe Z z neuronów terminalnych. Jeżeli odnotujemy błąd, czyli różnice $(X^{(j)} - Z^{(j)})$, to nie będziemy w stanie ustalić w jaki sposób za pojawienie się błędu odpowiadają neurony warstwy wyjściowej a jaki sposób powstał w elementach wcześniejszych (wewnętrznych) warstw. (Noszą one nazwę *warstw ukrytych*, “hidden layers”).

Przez wiele lat nie było dobrego pomysłu w jaki sposób uczyć warstwy ukryte.

Uczenie nieliniowej sieci wielowarstwowej

W latach 80-tych zaproponowano algorytm tzw. *wstecznej propagacji błędów* (backpropagation), polegający na tym że mając wyznaczony błąd $\delta^{(m)(j)}$ (j-ty krok uczenia m-tego neuronu) możemy "rzutować" ten błąd wstecz do wszystkich tych neuronów, których sygnały stanowiły wejścia do m-tego neuronu.

W sieci wielowarstwowej niemożliwe jest ściśle rozgraniczenie sygnałów wyjściowych od sygnałów wejściowych, z tego względu wprowadzamy jednolitą numeracje wszystkich neuronów, oraz stosujemy stałe oznaczenia $y_m^{(j)}$, niezależnie od tego czy jest to sygnał wejściowy czy wyjściowy (zawsze można założyć istnienie neuronu buforującego o odpowiedzi jednostkowej, takiego że $y_m^{(j)} = x_i^{(j)}$).

Uczenie nieliniowej sieci wielowarstwowej

Backpropagation

w j-tym kroku uczenia, sygnał na wyjściu m-tego neuronu może być wyznaczony z następującej zależności:

$$y_m^{(j)} = \varphi\left(\sum_{i \in \Omega_i} \omega_i^{(m)(j)} y_i^{(j)} \right)$$

gdzie Ω_i oznacza zbiór neuronów dostarczających sygnał wejściowy do konkretnego, rozważanego aktualnie m-tego neuronu.

Pozornie zapis jest niejednoznaczny, ale ta niejednoznaczność jest pozorna ponieważ można (jeżeli chwilowo pominiemy problem sprzężeń zwrotnych) wyznaczyć taką kolejność wyliczania elementu $y_m^{(j)}$, aby wszystkie $y_i^{(j)}$, konieczne do jego wyliczenia, były już znane.

Uczenie nieliniowej sieci wielowarstwowej

- Na samym początku wyznacza się poprawki dla neuronów stanowiących wyjściową warstwę sieci. Dla poszczególnych sygnałów $y_m^{(j)}$ istnieją w ciągu uczącym wzorcowe (oczekiwane) wartości $z_m^{(j)}$, z którymi można je porównywać, wyznaczając bezpośrednio błąd $\delta_m^{(j)}$.

$$\delta_m^{(j)} = (y_m^{(j)} - z_m^{(j)})$$

$$\Delta\omega_i^{(m)(j)} = \eta\, \delta_m^{(j)}\, d\phi(e)/de_m^{(j)}\, y_i^{(j)}$$

dla funkcji logistycznej wzór ten ulega znacznemu uproszczeniu:

$$\Delta\omega_i^{(m)(j)} = -\eta\, (z_m^{(j)} - y_m^{(j)})\, (1 - y_m^{(j)})\, y_i^{(j)}\, y_m^{(j)}$$

$k \in \Omega_0$

Uczenie nieliniowej sieci wielowarstwowej

- Dla warstwy ukrytej, przez analogię możemy zapisać

$$\Delta\omega_i^{(m)(j)} = \eta\, \delta_m^{(j)}\, d\phi(e)/de_m^{(j)}\, y_i^{(j)}$$

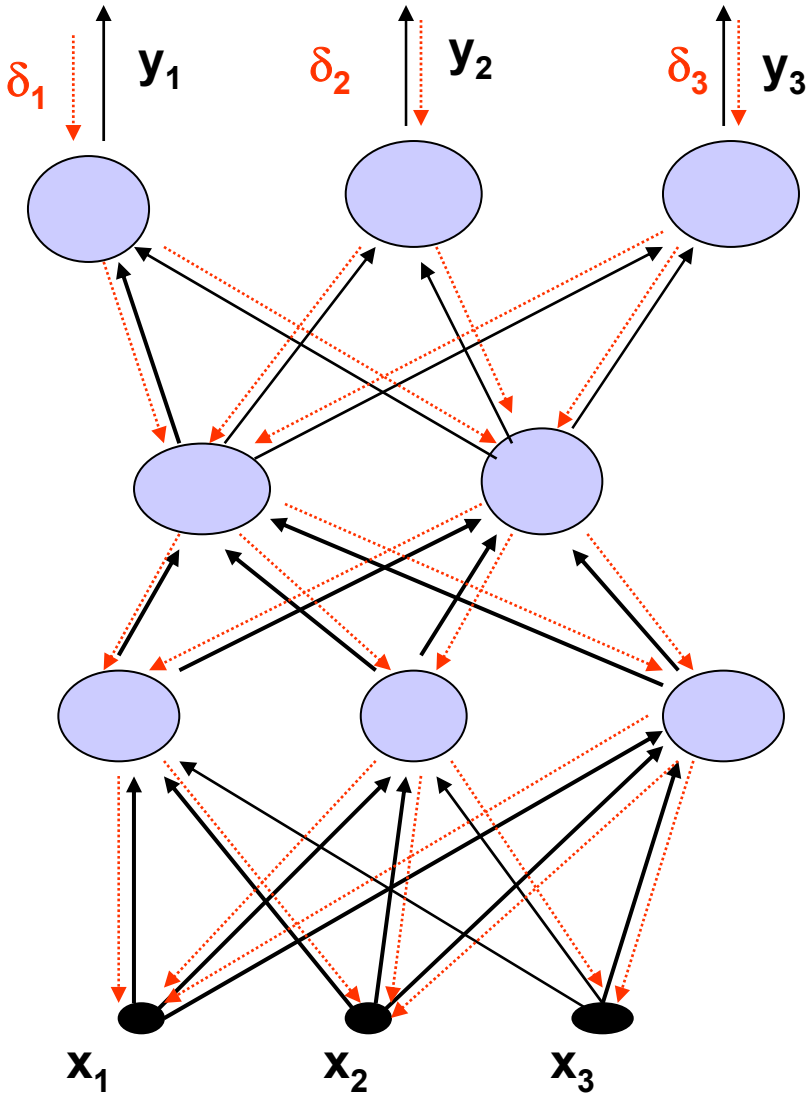
ale teraz nie mamy możliwości bezpośrednego wyznaczenia $\delta_m^{(j)}$. Załóżmy, że rozważany neuron należy do warstwy ukrytej, ale sygnały od niego docierają tylko do warstwy wyjściowej (dla której potrafimy określić $\delta_k^{(j)}$).
Wówczas (*backpropagation*)

$$\delta_m^{(j)} = \Sigma\ \omega_m^{(k)(j)}\, \delta_k^{(j)}$$

$\omega_m^{(k)(j)}$ – waga w neuronie o numerze k, przy jego wejściu m

Rzutowane wstecznie błędy przemnażane są przez te same współczynniki, przez które mnożone były sygnały, tyle tylko, że kierunek przesyłania informacji zostaje w tym przypadku odwrócony; zamiast od wejścia do wyjścia przesyła się je od wyjścia kolejno w kierunku wejścia.

Uczenie nieliniowej sieci wielowarstwowej



Propagacja sygnału

Propagacja błędu

Uczenie nieliniowej sieci wielowarstwowej

- Powyższą technikę propagacji wstecznej błędów można powtarzać dla kolejno coraz głębszej warstwy sieci. Każdy neuron z warstwy ukrytej albo przesyła sygnały do wartości wyjściowych, albo znajduje się w jednej z głębszych warstw, wówczas jego błąd można oszacować z chwilą określenia błędów dla wszystkich neuronów w sieci które są odbiorcą jego sygnałów.

W sieci *feedforward* zawsze daje się określić taką kolejność wstecznej propagacji błędów, która pozwala obliczyć błędy dla wszystkich elementów sieci. Uczenie tą metoda jest stosunkowo skuteczne ale powolne.

Przyspieszanie: techniki momentu, kumulanty błędów, odrzucania małych poprawek, itp.

Zestaw pytań do testu

1. Co to znaczy że sieć neuronowa ma zdolność uogólniania?
2. Co to jest miara Vapkina-Chervonenkisa?
3. Co to jest błąd weryfikacji i błąd uogólniania?
4. Narysuj typowy przebieg błędu uogólniania i błędu weryfikacji w funkcji ilości cykli uczących.
5. Podaj przykład nieliniowej formuły na pobudzenie sygnału perceptronu.
6. Jaka jest minimalna ilość perceptronów z której można zbudować zbudować funkcję XOR. Czy będzie to sieć warstwowa?
7. Co to znaczy „neuron ukryty”, „warstwa ukryta”.