

Wstęp do inteligencji komputerowej – zajęcia nr 3

Jarosław Stańczak
WSISiZ

Sztuczne sieci neuronowe:

- neuron a sztuczny neuron
- modele sieci neuronowych
- sieci warstwowe
- algorytm propagacji wstecznej – uczenie sieci neuronowych wielowarstwowych
- sieci ze sprzężeniem zwrotnym - pamięć w sieci neuronowej
- sieci samoorganizujące się
- zastosowania sieci neuronowych.

Sztuczne sieci neuronowe a układ nerwowy

Sztuczne sieci neuronowe (SSN), konstruowane od kilkudziesięciu lat przez ludzi, bez wątpienia są wzorowane na budowie i działaniu układu nerwowego zwierząt, a w szczególności ssaków. Dlatego też zanim zaczniemy je omawiać, warto będzie poświęcić chwilę na przypomnienie sobie budowy i działania pierwowzoru.

Układ nerwowy

Układ nerwowy zwierząt, a szczególnie kręgowców i ssaków jest najbardziej skomplikowanym z układów, budujących ciała zwierząt i ludzi. Jednocześnie jego budowa i działanie nie są jeszcze do końca poznane.

Jest to zbiór wyspecjalizowanych komórek, pozostających ze sobą w złożonych relacjach funkcjonalnych i strukturalnych, odpowiadający za sterowanie aktywnością organizmu. Układ nerwowy jest w stanie wykryć określone zmiany zachodzące w otoczeniu i wywołać w związku z tym odpowiednią reakcję organizmu.

(za Wikipedia)

Układ nerwowy

Tkanka nerwowa – utworzona przez **neurony (komórki nerwowe)** i komórki glejowe, tworzy **układ nerwowy**. Odbiera, przekazuje i reaguje na bodźce pochodzące ze środowiska zewnętrznego, jak na przykład dotyk, temperatura czy światło. Przewodzi impulsy (elektryczne i chemiczne) z neuronu do efektorów, od receptorów, przetwarza impulsy w adekwatne odpowiedzi, przewodzi impulsy z neuronu do innego neuronu, wytwarza substancje przekaźnikowe. Komórki nerwowe umożliwiają organizmowi normalne funkcjonowanie w danym środowisku, adekwatną odpowiedź w zależności od sytuacji w środowisku zarówno wewnętrznym jak i zewnętrznym. Neurony stale rejestrują się, analizują informacje o stanie wewnętrznym organizmu jak i zewnętrznym stanie otoczenia, przez co przygotowują organizm do adekwatnej reakcji. Do neuronów należy również koordynacja aktywności intelektualnej, świadomości, podświadomości, aktywności ruchowej czy też czynności innych gruczołów.

(za Wikipedia)

Układ nerwowy

Z tkanki nerwowej zbudowane są:

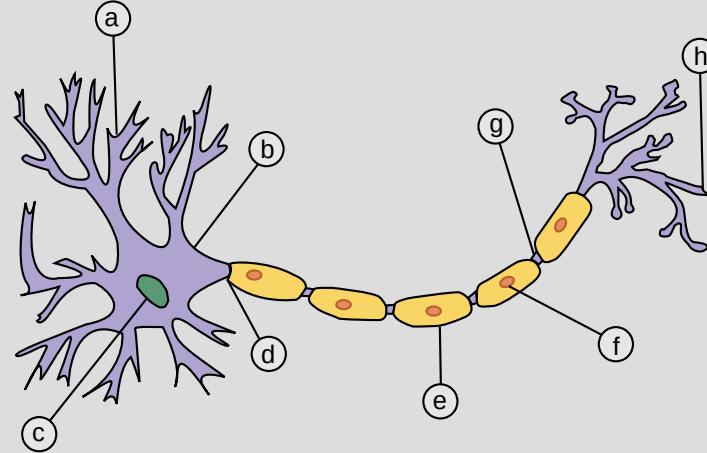
- ośrodkowy układ nerwowy
 - mózg
 - rdzeń kręgowy
- obwodowy układ nerwowy

W mózgu człowieka znajduje się ok. 10^{11} neuronów, a liczba synaps (połączeń między neuronami) jest szacowana na 10^{14} .

Nie jest prawdą, że mózg człowieka wykorzystuje tylko 10% swoich możliwości. Przyroda nie lubi tworzyć nadmiarowych i niewykorzystywanych układów, które dodatkowo pochłaniają znaczą ilość energii (mózg - 2,5% masy ciała zużywa ok. 20% energii w spoczynku!).

Mózg wykonuje wiele funkcji związanych z utrzymaniem funkcji życiowych naszego ciała, tak więc nigdy nie może w 100% poświęcić się „rozmyślaniom” na jakiś temat, a zajmuje się regulacją i podtrzymaniem działania naszego organizmu.

Neuron



Schemat budowy neuronu: a – dendryty, b – ciało komórki, c – jądro komórkowe, d – akson, e – otoczka mielinowa, f – komórka Schwanna, g – przewężenie Ranviera, h – zakończenia aksonu.
(źródło - Wikipedia)

Neuron - działanie

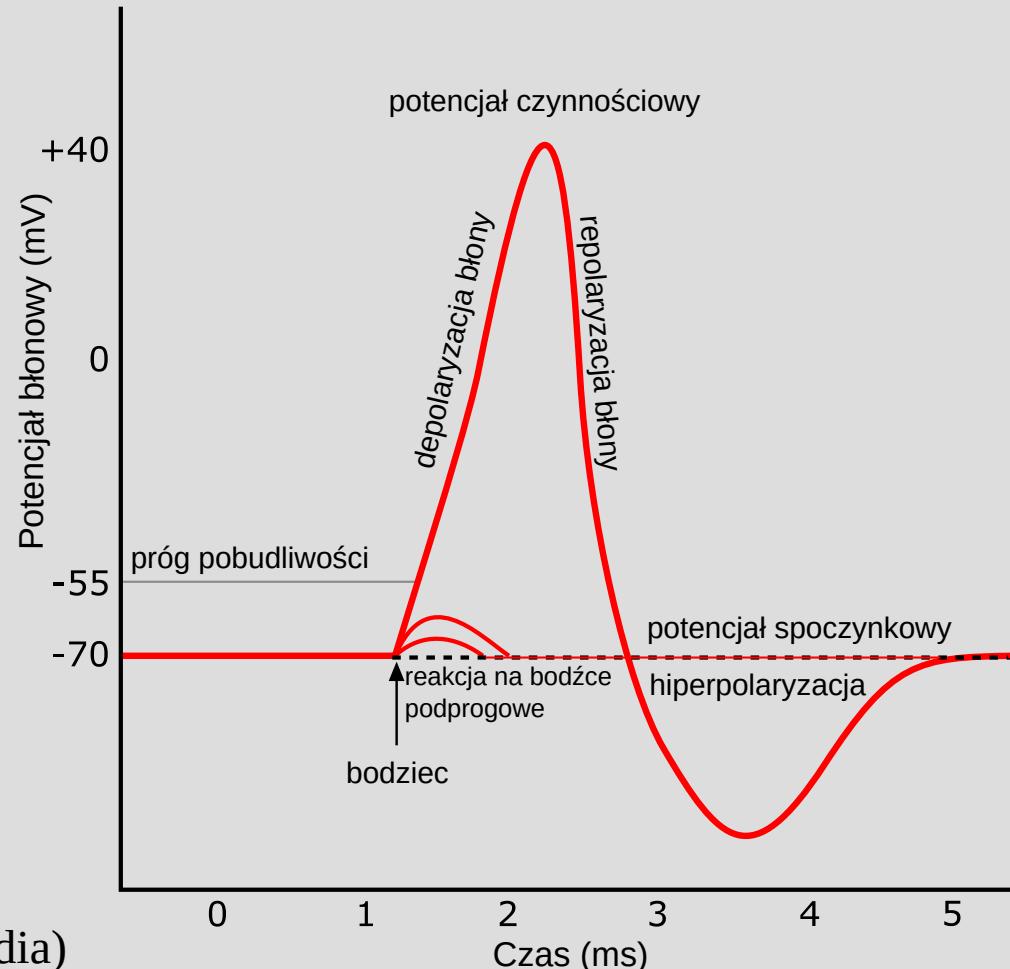
Podstawową funkcją neuronów jest przenoszenie i przetwarzanie informacji w postaci impulsów nerwowych (potencjałów czynnościowych), będących krótkotrwałymi, gwałtownymi zmianami potencjału błony komórkowej neuronu. Impulsy nerwowe w warunkach naturalnych są przewodzone tylko w jednym kierunku: od początkowego segmentu aksonu do synaps znajdujących się na jego zakończeniach.

Funkcjonalnie neuron można podzielić na cztery strefy:

- strefa wejścia – dendryty i ciało komórki, które odbierają impulsy od innych neuronów poprzez znajdujące się na nich synapsy
- strefa inicjacji – początkowy odcinek aksonu, tutaj powstaje potencjał czynnościowy neuronu
- strefa przewodzenia – akson
- strefa wyjścia – synapsy na zakończeniach aksonu.

(źródło - Wikipedia)

Neuron - działanie



(źródło rysunku - Wikipedia)

Historia powstania sztucznych sieci neuronowych (SSN)

- W 1943 r. powstał model sztucznego neuronu, podany przez McCullocha i Pittsa.
- W 1949 r. Hebb wyjaśnił mechanizm zapamiętywania i uczenia się nowych funkcji przez neurony, powstała tzw. reguła Hebba, wykorzystywana do nauki sieci także i współcześnie.
- W 1958 r. Rosenblatt zbudował pierwszą elektromechaniczną sztuczną sieć neuronową z możliwością nauki – Perceptron, udowodnił też zbieżność stosowanego procesu nauki.
- W 1960 r. Widrow i Hoff zbudowali pierwszy neurokomputer Adaline/Madaline składający się z 8 neuronów i 128 połączeń między nimi.

Historia powstania sztucznych sieci neuronowych (SSN)



Mark I Perceptron,
Cornell Aeronautical
Laboratory
<https://digital.library.cornell.edu/catalog/ss:550351>

Historia powstania sztucznych sieci neuronowych (SSN)

- Niestety (dla rozwoju sieci) w 1969 r. ukazała się praca Minsky'ego i Paperta, w której udowodnili oni, że sieć typu perceptronowego (jednowarstwowa) może nauczyć się problemów tylko liniowo-separowalnych. Spowodowało to upadek zainteresowania sieciami na prawie 20 lat, gdyż nie znano wtedy algorytmów nauki sieci wielowarstwowych, w których tego ograniczenia, jak się później okazało, nie ma.
- W 1982 r. powstała sieć Hopfielda – sieć ze sprzężeniami zwrotnymi (bez warstw ukrytych), mająca właściwość pamiętania informacji.
- Ponowne zainteresowanie sieciami pojawiło się po opublikowaniu przez McClellanda i Rumelharta (1986) i Andersona i Rosenfelda (1988) prac, zawierających m. in. algorytm „bacpropagation” - algorytm nauki sieci wielowarstwowej (z warstwami ukrytymi).
- W latach 1988-1990 ukazały się prace (m. in. Cybenki), dowodzące, że sieć zawierająca dwie warstwy ukryte może nauczyć się dowolnej funkcji klasyfikującej.
- Od tego czasu powstało wiele nowych rodzajów sieci, które z sukcesami zastosowano do rozwiązywania wielu problemów, m. in. tzw. głębokie uczenie i sieci konwolucyjne (CNN).

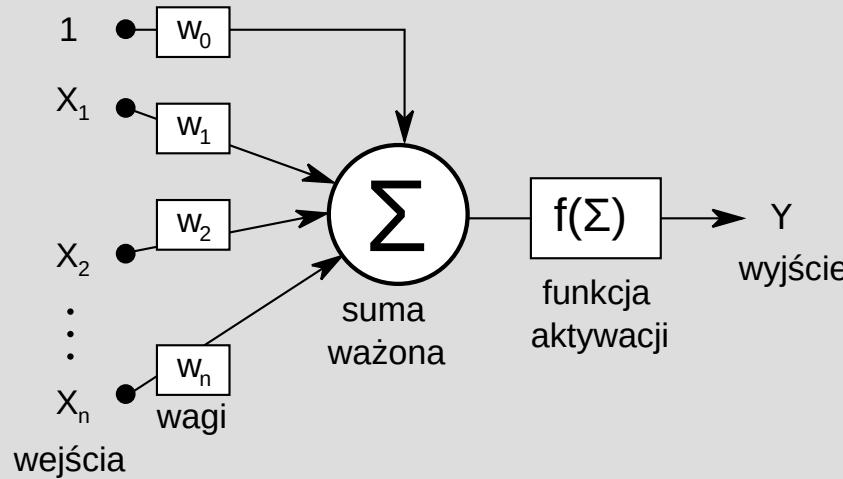
Cechy sztucznych sieci neuronowych (SSN)

Istniejące rodzaje sieci dość różnią się możliwościami i zastosowaniami, jednakże mają pewne cechy wspólne, które zadecydowały o ich popularności:

- zdolność uczenia się – wtedy gdy zostały wymyślone było to istotne novum, nie potrafiły tego robić żadne wymyślone wcześniej przez człowieka urządzenia;
- zdolność do generalizacji – sieci uczą się na przykładach, lecz generują także sensowne odpowiedzi dla danych, którymi nie były trenowane;
- odporność na uszkodzenia – sieć działa poprawnie nawet po usunięciu sporej części elementów;
- równolegle, bardzo szybkie przetwarzanie informacji;
- sieci dobrze nadają się do rozpoznawania obrazów, prognozowania, filtrowania zakłóceń, kompresji danych, modelowania działania skomplikowanych układów, rozwiązywania zadań klasy NP, rozwiązywania problemów w których posiadane dane są zaszumione i niepełne oraz wielu innych.

Te unikalne cechy dały im w pewnym momencie (lata 90 XX w. i początek XXI w.), wielki rozgłos i mnogość zastosowań. Obecnie ta popularność nieco osłabła z powodu powstania wielu innych, często dużo lepszych (lub może z lepszym wytłumaczeniem teoretycznym) metod rozwiązywania takich trudnych problemów.

Sztuczny neuron McCullocha-Pittsa



(źródło rysunku – Wikipedia)

$$Y = f \left(\sum_{i=0}^n w_i * x_i \right) \quad (1)$$

w_i – wagi, ich modyfikacja umożliwia naukę neuronu (w_0 – waga skojarzona z sygnałem stałym, może być uznawana za tzw. wartość progową aktywacji neuronu- Θ), X_i – sygnały wejściowe: wyjścia innych neuronów lub sygnały z zewnątrz ($X_0=1$ – wejściowa wartość stała), $f(\dots)$ – funkcja aktywacji, najczęściej o postaci skokowej lub z nasyceniem (tzw. funkcja sigmoidalna) dla wartości odległych od 0, Y – sygnał wyjściowy neuronu.

Funkcje aktywacji w sztucznym neuronie McCullocha-Pittsa

Najczęściej stosowane funkcje aktywacji:

- skokowa lub funkcja signum

$$y(x) = \begin{cases} 1 & \text{dla } x \geq 0 \\ 0 & \text{dla } x < 0 \end{cases} \quad (2)$$

lub

$$y(x) = sgn(x) = \begin{cases} -1 & \text{dla } x < 0 \\ 0 & \text{dla } x = 0 \\ 1 & \text{dla } x > 0 \end{cases} \quad (3)$$

- sigmoidalna, tangens hiperboliczny, Gaussa

$$y(x) = \frac{1}{1 + \exp(-\beta x)} \quad (4)$$
$$y(x) = \tanh(\beta x) \quad (5)$$
$$y(x) = a \cdot \exp\left(-\frac{(x-b)^2}{2 \cdot c^2}\right) \quad (6)$$

- liniowa, liniowa z nasyceniem lub liniowa obcięta (ReLU)

$$y(x) = a * x \quad (7)$$
$$y(x) = \begin{cases} 0 & \text{dla } x < 0 \\ a * x & \text{dla } 0 \leq x < c/a \\ c & \text{dla } x \geq c/a \end{cases} \quad (8)$$
$$y(x) = \begin{cases} 0 & \text{dla } x \leq 0 \\ x & \text{dla } x > 0 \end{cases} \quad (9)$$

Neuron rzeczywisty a sztuczny neuron McCullocha-Pittsa

Większość istniejących sieci neuronowych buduje się wykorzystując ten sam prosty model neuronu McCullocha-Pittsa. Jest on uważany za uniwersalny i mogący posłużyć za podstawową jednostkę budującą dowolnie skomplikowaną maszynę obliczeniową. Jednakże nie odzwierciedla on niektórych cech prawdziwego neuronu:

- rzeczywiste neurony nigdy nie mają charakterystyki progowej, zawsze nieliniową lecz ciągłą – ta cecha oczywiście często jest uwzględniona w budowanych modelach;
- wagi wejściowe nie muszą być liniowe, mogą same w sobie realizować już pewne funkcje przetwarzające, cecha ta jest zazwyczaj pomijana w konstruowanych SSN;

Neuron rzeczywisty a sztuczny neuron McCullocha-Pittsa

- wyjście wzbudzonego neuronu nigdy nie jest sygnałem stałym lecz impulsem bądź ciągiem impulsów, w których znaczenie może mieć wartość, faza, częstotliwość, opóźnienie sygnałów, czas działania, synchronizacja między neuronami – właściwości te nie zostały całkowicie przebadane i zazwyczaj nie są uwzględniane w SSN, a ich rola w działaniu sieci może być duża, jednakże ostatnio powstają sieci impulsowe, wykorzystujące ciągi impulsów do komunikacji między neuronami;
- działanie synaps (połączeń między neuronami) nie jest do końca deterministyczne, a w każdym razie wpływ na przekazywanie sygnału mają różne czynniki chemiczne, hormony, leki, substancje psychoaktywne, itp., również najczęściej nie są to zjawiska modelowane w SSN.

Niestety nie ma jednoznacznej odpowiedzi na pytanie, czy pominięcie tych cech nie zubaża w istotny sposób SSN w stosunku do pierwowzoru.

Architektura SSN

SSN mogą być budowane ze sztucznych neuronów na różne sposoby. Najczęściej przyjmuje się cztery główne architektury sieci:

- jednokierunkowe (typu perceptronowego), w których wyjścia z warstwy poprzedzającej są wejściami warstwy następnej, czyli sygnał wejściowy jest przekazywany od wejścia do wyjścia (nie dotyczy to fazy uczenia), np. sieć BP;
- rekurencyjne, zawierające sprzężenia zwrotne (połączenia wyjść neuronów z wejściami), sieci takie mają właściwość pamięci swojego stanu, np. sieć Hopfielda, Boltzmanna, BAM, LSTM;
- sieci samoorganizujące się (komórkowe), w których każdy neuron jest połączony tylko z kilkoma swoimi sąsiadami, np. sieć Kohonena.
- inne: ART, SVM, RBF.

Oczywiście można sobie wyobrazić różne inne sposoby łączenia sztucznych neuronów, jednakże nie są one w praktyce używane z uwagi na trudną analizę właściwości i brak algorytmów uczenia.

Architektura, uczenie i funkcje SSN

Przyjęta architektura sieci w dużej mierze determinuje jej funkcje, sposób wykorzystania i metodę uczenia:

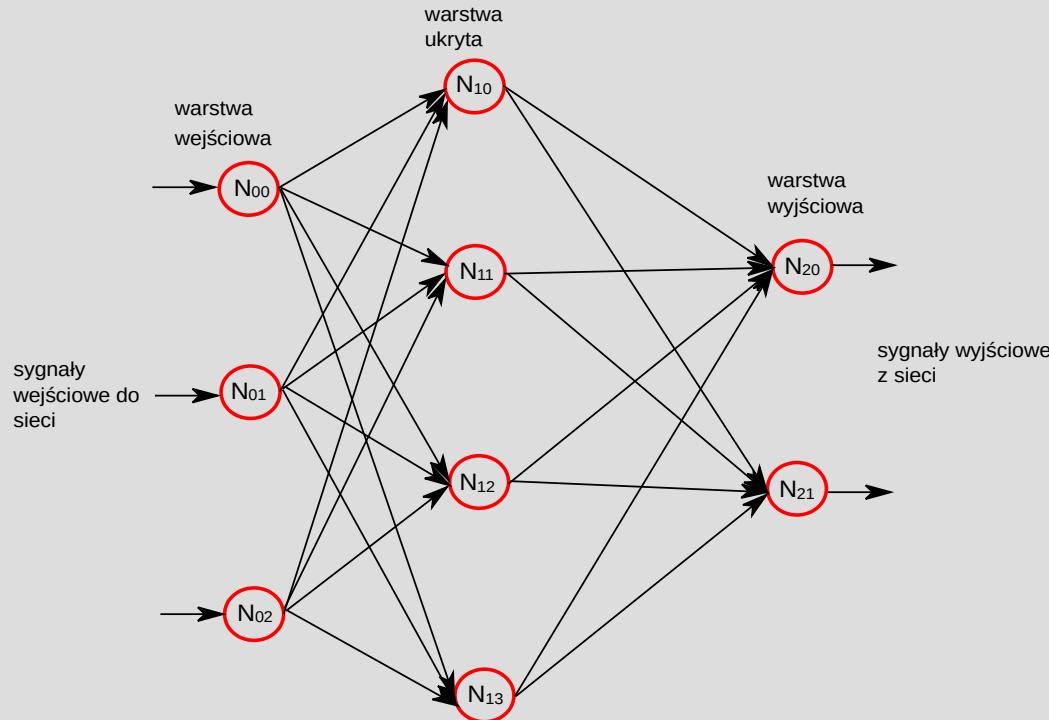
- sieci jednokierunkowe najlepiej nadają się do przetwarzania sygnałów, najczęściej wykorzystywane jest tu uczenie z nauczycielem przy użyciu różnych wariantów metody BP;
- sieci rekurencyjne mają zdolność pamiętania, wykorzystywane są jako pamięci adresowane zawartością, potrafią odtworzyć całość zapisanej informacji na podstawie jej fragmentu (autoasocjacja), informacja jest w nich zapisywana jako wagi wyliczane z pewnych funkcji, przypominających stany energetyczne częstek, których minima to zapamiętane wzorce;
- sieci komórkowe i pozostałe służą najczęściej jako klasyfikatory danych, uczą się (bez nauczyciela) przetwarzając nadchodzące dane, wykrywając na bieżąco ich cechy i klasyfikując do utworzonych kategorii.

Sieci jednokierunkowe

Sieci jednokierunkowe składają się z warstw. Mogą mieć warstwę wejściową, dowolną liczbę warstw ukrytych (niedostępnych bezpośrednio ani od strony wejść, ani od strony wyjść) i powinny mieć warstwę wyjściową (w szczególności mogą mieć też tylko jedną warstwę, która pełni wszystkie funkcje). Sygnał wyjściowy z każdego neuronu warstwy poprzedniej jest doprowadzany do wejść każdego neuronu warstwy następnej (przy niektórych metodach uczenia niektóre słabsze połączenia mogą być eliminowane dla uproszczenia sieci).

W sieciach jednokierunkowych sygnał (poza fazą uczenia) przebiega wyłącznie od wejść do wyjść, bez sprzężeń zwrotnych i wymiany między neuronami w obrębie tej samej warstwy lub z pominięciem którejś z warstw.

Sieci jednokierunkowe

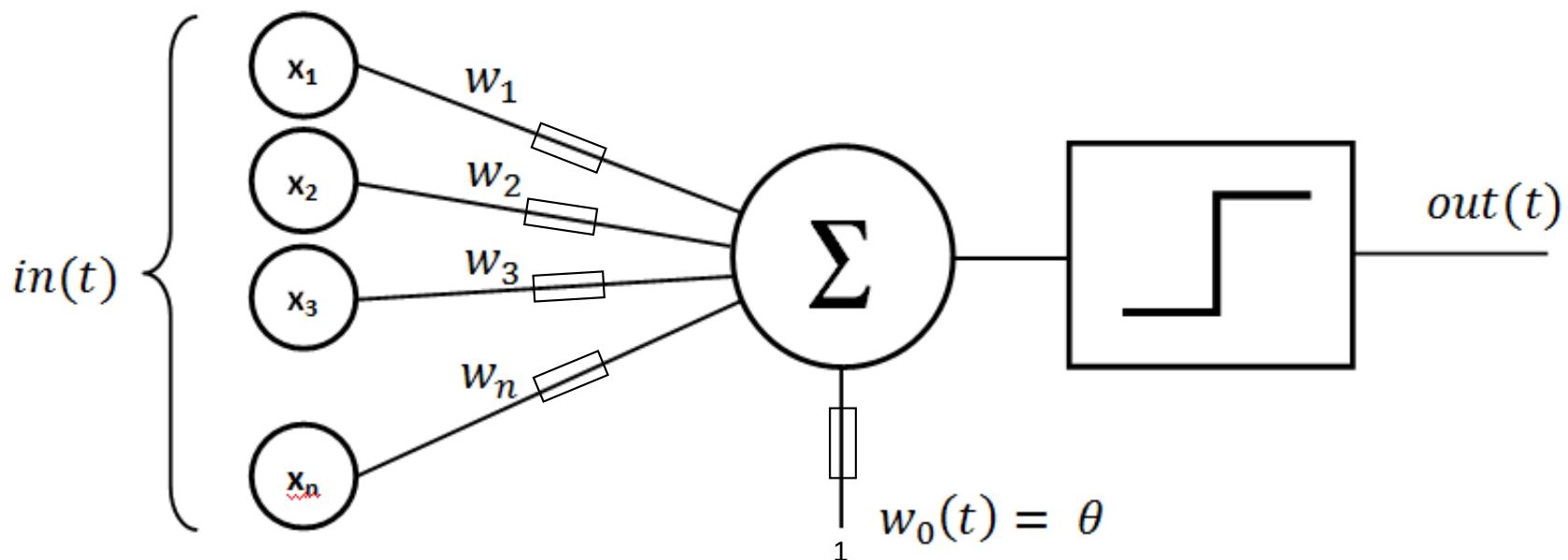


Sztuczna sieć neuronowa o 3 neuronach w warstwie wejściowej (N_{00} , N_{01} i N_{02}), 4 w warstwie ukrytej (N_{10} , N_{11} , N_{12} i N_{13}) i 2 w warstwie wyjściowej (N_{20} i N_{21}).

Sieci jednokierunkowe

perceptron

Perceptron był jedną z pierwszych реализациj (Rosenblatt-Wightman, 1957) sztucznej sieci neuronowej zbudowanej na bazie neuronu McCullocha-Pittsa. Była to realizacja sprzętowa, elektromechaniczna z wagami w postaci potencjometrów, poruszanych silniczkami (komputerowa symulacja takiego urządzenia była ówcześnie raczej niemożliwa), składała się z jednego, a później kilku właściwie niezależnych neuronów.



Sieci jednokierunkowe

perceptron

Niestety nie mogła to być sieć wielowarstwowa, gdyż nie potrafiono uczyć warstw ukrytych.
Wzór na działanie perceptronu można zapisać jako:

$$y(t) = \begin{cases} 1 & \text{if } \sum_{i=0}^n w_i * x_i(t) > 0 \\ 0 & \text{if } \sum_{i=0}^n w_i * x_i(t) \leq 0 \end{cases} \quad (10)$$

gdzie: w_0 – odpowiada progowi, $w_1 \dots w_n$ – wagom na wejściach $x_1(t) \dots x_n(t)$, $y(t)$ to sygnał wyjściowy.

Sieci jednokierunkowe

faza eksploatacji

Działanie sieci jednokierunkowej w fazie eksploatacji jest proste: po przyłożeniu sygnałów na wejściu, neurony wejściowe obliczają swoje wyjścia zgodnie ze wzorem (1) (czasem neurony wejściowe pełnią tylko funkcje dystrybucyjne i nie modyfikują sygnału) i podają je na wejścia neuronów warstwy kolejnej (najczęściej ukrytej), tu sytuacja się powtarza aż sygnał dotrze do wyjść sieci. Zdarza się, że funkcje aktywacji w różnych warstwach sieci są różnego typu.

W wersjach sprzętowych sieci neurony muszą być odpowiednio synchronizowane aby uniknąć „wyścigu” sygnałów związanych z różnym czasem propagacji, dla wersji programowych nie ma tego problemu.

Zdecydowanie inaczej sieć działa w fazie uczenia.

Sieci jednokierunkowe

faza uczenia

Aby sieć jednokierunkowa mogła pracować w fazie eksploatacji, najpierw należy ją nauczyć wykonywać swoją pracę. Uczenie w skrócie polega na przetwarzaniu sygnałów zgodnie z odpowiednimi przykładami zawierającymi wejścia i pożądane dla nich wyjście. Przykłady te trzeba najpierw sieci zaprezentować. Faza uczenia polega na wielokrotnym prezentowaniu przykładów, obliczaniu odpowiedzi sieci i korekcji wag, aż błąd między wyjściem sieci a wzorcem spadnie poniżej określonego minimum. Przykłady można prezentować po kolejno, ale znacznie lepsze efekty daje prezentacja w kolejności losowej, należy jednak pamiętać, aby wszystkie przykłady były pokazywane tak samo często. Wagi startowe dla sieci przed uczeniem są **losowane** najczęściej jako niewielkie liczby rzeczywiste.

Wagi startowe nie mogą być jednakowe!

Sieci jednokierunkowe

faza uczenia

Początkowo sieci neuronowe (np. ADALINE, MADALINE) budowane były z neuronów o skokowej funkcji aktywacji (poniżej pewnego progu na wyjściu było 0, powyżej 1 – była to swoista analogia do techniki cyfrowej). Jednakże powstał tu pewien problem, neurony takie można było uczyć np. zgodnie z tzw. regułą Hebb'a (o której więcej na następnym slajdzie), jednakże znamy dla nich wartości sygnałów wzorcowych tylko dla warstwy wyjściowej, nie wiadomo jakie sygnały wzorcowe powinny mieć warstwy ukryte. Dlatego też pierwsze sieci jednokierunkowe miały tylko jedną warstwę. Wspomniani już Minsky i Pappert wykazali, że sieć taka ma mocno ograniczone możliwości – może działać jako klasyfikator, w którym klasy są liniowo-separowalne, w praktyce oznacza to, że sieć typu perceptronowego nie mogła nauczyć się realizować np. funkcji logicznej XOR. Praca ta spowodowała prawie dwudziestoletni regres w badaniach nad sieciami. Jednakże problemy udało się przewyściężyć.

Sieci jednokierunkowe

faza uczenia – reguły Hebbia/Widrowa-Hoffa i Oji

Neurony w zasadzie wszelkich sieci neuronowych uczy się zgodnie z zasadami tzw. reguły Hebbia, która głosi, że jeśli neuron B jest cyklicznie pobudzany przez neuron A, to staje się on bardziej czuły na to pobudzanie, co można przedstawić jako wzrost wartości wagi łączącej te neurony (wzór (11) jest znany jako tzw. reguła Widrowa-Hoffa):

$$\Delta w_{AB}(t+1) = \alpha * x_A(t) * y_B(t) \quad (11)$$

Jednakże okazało się, że reguła ta może prowadzić do nieograniczonego wzrostu wagi łączącej neurony, gdyby zbadać graniczne właściwości tak uczonej sieci. Oczywiście w praktyce nigdy nie stosuje się nieskończonej liczby powtórzeń sygnałów uczących, jednak postarano się wyeliminować tę niedogodność. Można to zrobić przez sztuczne ograniczenie wielkości wag lub też przez normalizację wektora wag po każdej iteracji, co prowadzi do reguły uczenia Oji:

$$\Delta w_{AB}(t+1) = \alpha * y_B(t) * (x_A(t) - y_B(t) * w_{AB}(t)) \quad (12)$$

$w_{AB}(t+1)$, $\Delta w_{AB}(t+1)$ – waga i przyrost wagi połączenia między neuronami, α - niewielka stała uczenia, $y_B(t) = wzor_B(t) - x_B(t)$ – sygnał uczący, $x_A(t)$, $x_B(t)$ – sygnały wyjściowe neuronów, A i B oznaczają tu neurony z kolejnych warstwy sieci (A – poprzedniej, B – następnej).

Sieci jednokierunkowe

uczenie perceptronu

Reguła uczenia perceptronu wygląda następująco:

1. Wagi startowe powinny być zainicjowane niewielkimi wartościami rzeczywistymi lub zerami*.
2. Dla każdego z s uczonych wzorców d_j ($j=1 \dots s$) należy:
 - obliczyć wyjście sieci $y_j(t)$ na podstawie wzoru (2);
 - skorygować wagi według wzoru $w_i(t+1) = w_i(t) + \eta * (d_j - y_j(t)) * x_{ij}$, gdzie η to współczynnik uczenia;
3. Powtarzać pkt. 2 aż miara błędu $\frac{1}{s} * \sum_{j=1}^s |d_j - y_j(t)| < \varepsilon$, gdzie ε to wartość ustalona przez użytkownika.

* W niektórych publikacjach zerowanie wag startowych nie jest zalecane.

Sieci jednokierunkowe

ćwiczenie uczenia perceptronu

Proszę dobrać wagi perceptronu, realizującego funkcję logiczną AND z $\varepsilon < 0,5$, współczynnik uczenia η przyjąć jako 0,3, wagi startowe równe odpowiednio 0; 0; 0,1.

Funkcja AND:

X1 \ X2	0	1
0	0	0
1	0	1

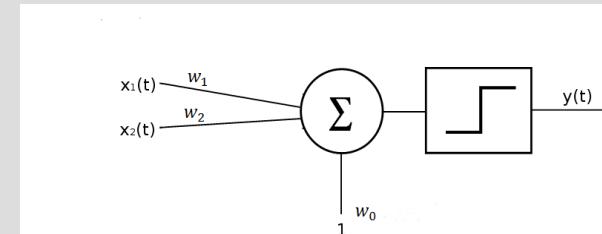
$$f(x) = \begin{cases} 1 & \text{dla } x > 0 \\ 0 & \text{dla } x \leq 0 \end{cases} - f. \text{aktywacji neuronu}$$

1. $w_0(0)=0; w_1(0)=0; w_2(0)=0,1; d(0)=0; X_0(0)=1; X_1(0)=0; X_2(0)=0$
 $y(0)=f(w_0(0)*X_0(0) + w_1(0)*X_1(0) + w_2(0)*X_2(0))=f(0*1+0*0+0,1*0)=0;$
 $w_0(1)=w_0(0)+\eta*(d(0)-y(0))*x_0(0)=0+0,3*(0-0)*1=0$
 $w_1(1)=w_1(0)+\eta*(d(0)-y(0))*x_1(0)=0+0,3*(0-0)*0=0$
 $w_2(1)=w_2(0)+\eta*(d(0)-y(0))*x_2(0)=0,1+0,3*(0-0)*0=0,1$

2. $w_0(1)=0; w_1(1)=0; w_2(1)=0,1; d(1)=0; X_0(1)=1; X_1(1)=0; X_2(1)=1$
 $y(1)=f(w_0(1)*X_0(1) + w_1(1)*X_1(1) + w_2(1)*X_2(1))=f(0*1+0*0+0,1*1)=1;$
 $w_0(2)=w_0(1)+\eta*(d(1)-y(1))*x_0(1)=0+0,3*(0-1)*1=-0,3$
 $w_1(2)=w_1(1)+\eta*(d(1)-y(1))*x_1(1)=0+0,3*(0-1)*0=0$
 $w_2(2)=w_2(1)+\eta*(d(1)-y(1))*x_2(1)=0,1+0,3*(0-1)*1=-0,2$

3. $w_0(2)=-0,3; w_1(2)=0; w_2(2)=-0,2; d(2)=0; X_0(2)=1; X_1(2)=1; X_2(2)=0$
 $y(2)=f(w_0(2)*X_0(2) + w_1(2)*X_1(2) + w_2(2)*X_2(2))=f((-0,3)*1+0*1+(-0,2)*0)=0;$

...



Sieci jednokierunkowe

faza uczenia – metoda Backpropagation

Wraz z opracowaniem i opublikowaniem w 1986 r metody BP (metoda ta została wynaleziona niezależnie przez kilku badaczy, po raz pierwszy już w 1969, lecz nie doczekała się wtedy rozwisu) badania nad sieciami nabraly nowego rozpedu. Wreszcie mozna bylo konstruowac w pełni uniwersalne sieci wielowarstwowe. Warunkiem bylo zastosowanie w nich neuronów o różniczkowalnych funkcjach aktywacji. Wtedy możliwe stało się wykorzystanie typowych, gradientowych metod minimalizacji funkcji błędu popełnianego przez sieć, np. o postaci (13), a następnie propagacja tego błędu do kolejnych warstw ukrytych zmodyfikowanego przez wartości pochodnych funkcji aktywacji neuronów. Uczenie sieci w dalszym ciągu polega na prezentacji jej w odpowiedniej (np. losowej) kolejności wejść z wymaganymi wyjściami, obliczenie wartości błędu, a następnie jego propagacja w kierunku odwrotnym (wstecznym) do normalnej propagacji sygnału i korekcja wag na podstawie sygnału błędu.

Minimalizowane kryterium jakości:

$$Q(W) = \frac{1}{2} \sum_{i=0}^n \sum_{j=0}^{m_s} (Y_{ij}^s - X_{ij}^s)^2 \quad (13)$$

$Q(W)$ – funkcja jakości zależna od zbioru wag (W) sieci, Y_{ij}^s – wzorcowe wartości wyjść z sieci, n – liczba wzorców, i – indeks wzorca, j – indeks neuronu wyjściowego, m_s – liczba neuronów warstwy s (wyjściowej), X_{ij}^s – wyjście neuronów warstwy s (wyjściowej) otrzymane dla uczonych wzorców.

Sieci jednokierunkowe

faza uczenia – metoda Backpropagation

Wzory na korekcje wag w tej metodzie można otrzymać różniczkując kryterium (13) względem wag, po podstawieniu do niego wzorów na obliczenie wyjścia z sieci (1). Jest to tzw. **reguła delta**.

- Dla warstwy wyjściowej ($k=s$) będzie to:

$$\Delta W_{jl}^s = \eta \sum_{i=0}^n \delta_{ij}^s * X_{il}^{s-1}$$

$$\delta_{ij}^s = g'_{j}^s \left(\sum_{p=0}^{m_{s-1}} W_{pj}^s * X_{ip}^{s-1} \right) * (Y_{ij}^s - X_{ij}^s) \quad (14)$$

- Dla warstw ukrytych ($k=2..s-1$):

$$\Delta W_{jl}^k = \eta * \sum_{i=0}^n \delta_{ij}^k * X_{il}^{k-1}$$

$$\delta_{ij}^k = g'_{j}^k \left(\sum_{q=0}^{m_{k-1}} W_{jq}^k * X_{iq}^{k-1} \right) * \sum_{p=0}^{m_{k+1}} W_{pj}^{k+1} * \delta_{ip}^{k+1} \quad (15)$$

- Dla warstwy wejściowej ($k=1$):

$$\Delta W_{jl}^1 = \eta * \sum_{i=0}^n \delta_{ij}^1 * We_{il}^0$$

$$\delta_{ij}^1 = g'_{j}^1 \left(\sum_{q=0}^{m_0} W_{jq}^1 * We_{iq}^0 \right) * \sum_{p=0}^{m_2} W_{pj}^2 * \delta_{ip}^2 \quad (16)$$

n – liczba wzorców, i – indeks wzorca, j – indeks neuronu w warstwie, m_k – liczba neuronów warstwy k , m_0 – liczba wejść do sieci, Y_{ij}^s – wzorcowe wartości wyjść z sieci, We_{il}^0 – wzorcowe wartości wejść do sieci, X_{ij}^k – wyjścia neuronów warstwy k dla uczonych wzorców, ΔW_{jl}^k , W_{jl}^k – zmiana wagi, waga łącząca neurony l w warstwie $k-1$ i j w warstwie k , η – współczynnik nauki, $g'_{j}^k(\dots)$ - wartość pochodnej funkcji aktywacji $g_j^k(\dots)$ neuronu j .

Sygnal **delta** jest odpowiednikiem **błędu** propagującego od wyjścia sieci do warstwy wejściowej.

Sieci jednokierunkowe

faza uczenia – metoda Backpropagation

Wzory (14-16) mogą wyglądać nieco inaczej w różnych opracowaniach nie tylko z powodu użycia innych oznaczeń, ale także dlatego, że:

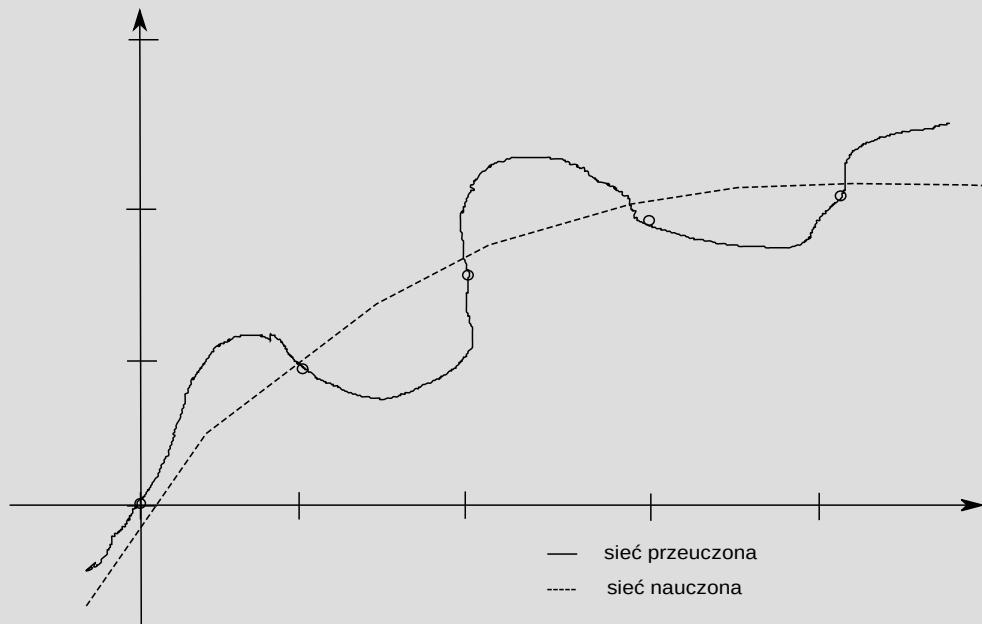
- możliwa jest optymalizacja nieco innej postaci kryterium jakości niż (13),
- możliwe jest użycie bardziej skomplikowanych metod optymalizacji niż metoda gradientowa największego spadku, np. metody Levenberga-Marquardta lub zmiennej metryki, są to metody o znacznie szybszej zbieżności,
- możliwe jest użycie dodatkowych technik poprawiających zbieżność, np. metody momentum (uśredniania kolejnych zmian wag) lub adaptacji parametrów – strojenia wartości współczynnika uczenia η ,
- ewentualnie technik pozwalających pokonać minima lokalne występujące w optymalizowanej funkcji.

Stosuje się też czasem metody heurystyczne do znajdowania wag sieci (działają one również dla sieci z progową funkcją aktywacji, dla których nie ma metod gradientowych), np. algorytmy ewolucyjne.

Sieci jednokierunkowe

problem przeuczenia sieci

W sztucznych sieciach neuronowych typu BP czasem zauważa się efekt przeuczenia sieci. Polega on na tym, że sieć bardzo dobrze i z małym błędem nauczyła się trenowanych przykładów, jednakże traci zdolność generalizacji, czyli sensownych odpowiedzi na przykłady nieprezentowane. Niewielka zmiana sygnałów na wejściu, daje znaczne zmiany na wyjściu. Jest to efekt niekorzystny i jak widać należy zachować umiar w długości uczenia, czyli „pogoń” za zmniejszeniem błędu nauczenia sieci.



Sieci jednokierunkowe

architektura sieci

Oprócz uczenia sieci, dużym problemem jest też jej zaprojektowanie, czyli dobranie liczby warstw i liczb neuronów w warstwach. Generalnie nie ma tu zbyt wielu wytycznych. Warstwa wejściowa i wyjściowa mają liczby neuronów zgodne z liczbą sygnałów w danych trenujących. Wiadomo, że w pełni uniwersalna sieć powinna mieć przynajmniej jedną (w niektórych pracach podane jest, że przynajmniej dwie, zależy to od tego, jakiego rodzaju funkcje są brane pod uwagę: tylko ciągłe, czy dowolne) warstwę ukrytą z np. $2n+1$ neuronami, gdzie n to liczba wejść. Zazwyczaj stosuje się jednak mniejsze sieci. Niektóre wersje algorytmów nauki sieci mają możliwość likwidacji wag a nawet całych neuronów, których wkład do działania sieci jest niski – np. algorytm Optimal Brain Damage, wykorzystujący drugie pochodne kryterium jakości liczone po wagach. Współczynnik istotności zależny od tych pochodnych pokazuje, które neurony można usunąć. Innym rozwiązaniem tego problemu jest zastosowanie algorytmów heurystycznych, które mogą posłużyć do nauczenia sieci i wybrania jej struktury.

Sieci jednokierunkowe

głębokie uczenie

Rozwijająca się ostatnio gałąź SSN, nazywana sieciami z „głębokim uczeniem” odnosi się zazwyczaj do sieci jednokierunkowych (choć mogą istnieć warianty dla innych rodzajów sieci z podobnym do BP algorymem uczenia).

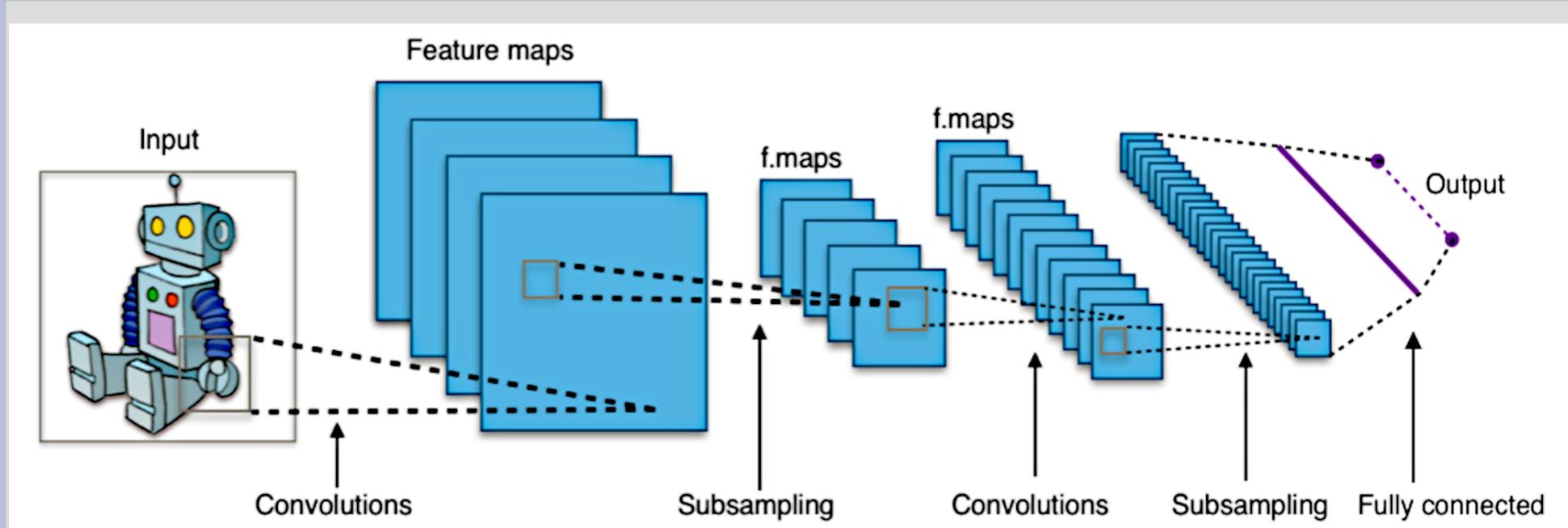
SSN z „głębokim uczeniem” posiadają zazwyczaj dużo warstw (stąd nazwa), co jest niejako przeciwnie do obowiązujących wcześniej standardów, kiedy to uważano, że sieć z 3 warstwami jest zupełnie wystarczająca.

SSN z głębokim uczeniem posiadają warstwy niekoniecznie o jednakowej strukturze i sieci połączeń, jak budowano sieci do niedawna, a pojawiają się warstwy ze specyficzną strukturą połączeń, często wzorowaną na budowie np. kory wzrokowej zwierząt.

Pojawiają też warstwy o innych sposobach działania neuronów, np. warstwy obliczające splot funkcji wejściowych (sieci konwolucyjne, CNN), a nie ich proste sumy ważone.

Sieci jednokierunkowe

głębokie uczenie – sieć CNN



(Źródło: Wikipedia)

Sieć konwolucyjna (CNN) jest strukturą hierarchiczną z połączeniami tylko w obrębie wybranych segmentów (subsampling) i splotem (convolution) jako operacją modyfikującą otrzymane sygnały, co umożliwia ekstrakcję cech sygnałów (feature maps). Sieć taka świetnie nadaje się do przetwarzania obrazów i wyszukiwania ich cech, za co odpowiedzialne są warstwy i odpowiednio połączone ich segmenty.

Sieci jednokierunkowe

głębokie uczenie

Splot (konwolucja) w wersji dyskretnej:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f(n-m)g(m) \quad (17)$$

przy czym jedną z tych funkcji są sygnały wejściowe do neuronu, a drugą wag, czyli jest to suma iloczynów macierzy/wektorów wartości wejściowych i wag (wyrazy spoza macierzy/wektorów są traktowane jako 0). Wynikiem są macierze, które są redukowane do pojedynczych wartości przez warstwy próbujące (średnia, min, max). Z punktu widzenia teorii sygnałów neuron w takiej sieci pełni rolę filtra, modyfikującego sygnał wejściowy.

Te wszystkie modyfikacje struktury sieci odbijają się też na działaniu metody uczenia, zblżonej do BP, jednak z innymi wzorami na uczenie różnych warstw, zależnymi od ich funkcji i struktury.

BP – to algorytm uczenia oparty na gradientowej metodzie największego spadku poszukiwania minimum funkcji błędu, więc jeśli tylko istnieją pochodne (a tu istnieją) zastosowanych funkcji aktywacji i przejścia, to da się ją wyprowadzić dla prawie dowolnych struktur sieci i funkcji aktywacji neuronów.

Sieci jednokierunkowe

zastosowania

Sieci jednokierunkowe najczęściej stosuje się do:

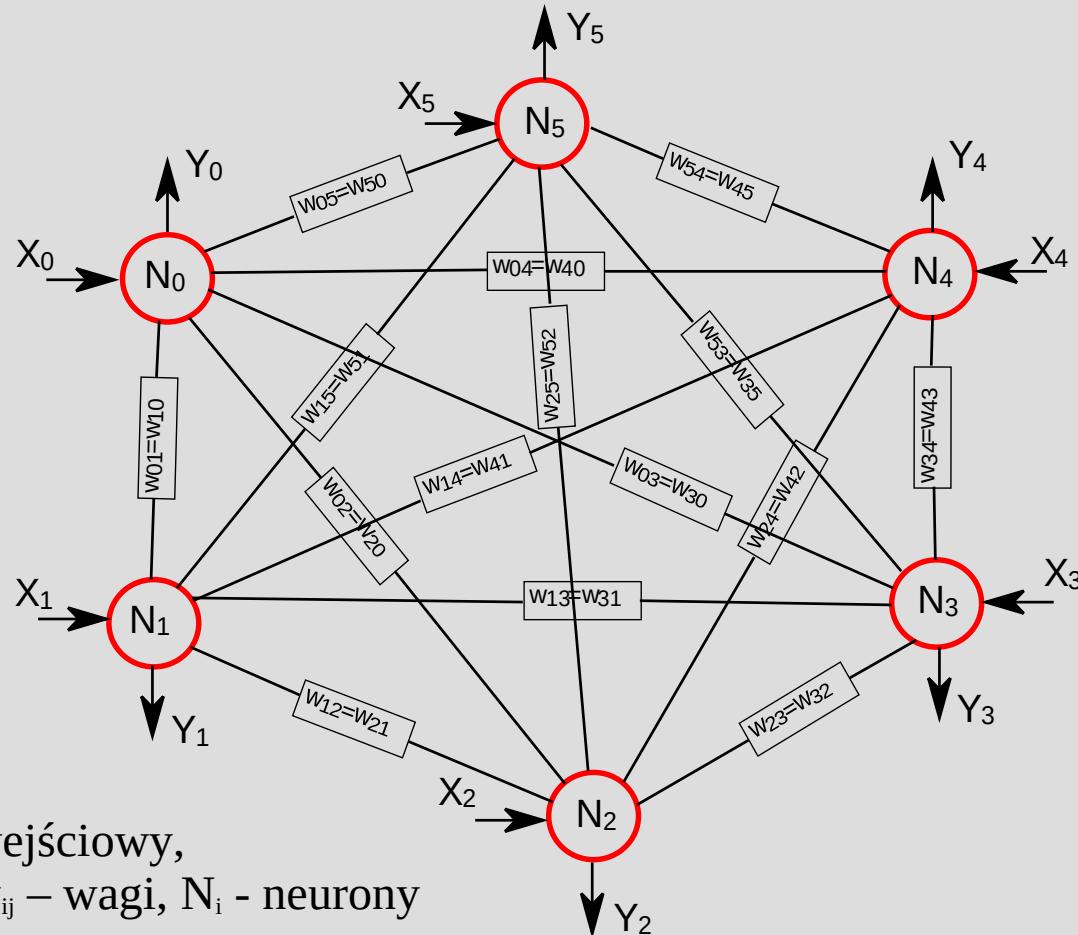
- rozpoznawania pisma i obrazów
- przetwarzanie obrazów – np. kolorowanie cz.-b. zdjęć i filmów
- predykcji i prognozowania sygnałów
- kompresji obrazów
- w algorytmach kierowania pojazdami
- nauki strategii w grach
- rozpoznawania mowy
- modelowania skomplikowanych obiektów np. w automatyce.

Sieci ze sprzężeniem zwrotnym – rekurencyjne sieci neuronowe

Sieci ze sprzężeniem zwrotnym to drugi po sieciach jednokierunkowych model sieci, opracowany jeszcze przed ponownym wzrostem zainteresowania nimi w latach 90 XX w. Pierwsza taka sieć została wymyślona przez J. Hopfielda w 1982 r. W zasadzie można ją rozpatrywać jako sieć jednowarstwową w której wyjście z każdego neuronu (dostępne także jako sygnał wyjściowy z sieci) jest podawane na wejścia wszystkich innych neuronów oprócz swojego wejścia ($w_{ii}=0$). Wagi w tak połączonej sieci są symetryczne ($w_{ij}=w_{ji}$). Oczywiście można sobie też wyobrazić wariant niesymetryczny, lecz nie jest on stosowany, zapewne z uwagi na brak lub małą liczbę stanów stabilnych sieci.

Budowę takiej sieci zilustrowano na następnym slajdzie.

Sieci ze sprzężeniem zwrotnym – sieć Hopfielda



X_i – zewnętrzny sygnał wejściowy,

Y_i – sygnał wyjściowy, w_{ij} – wagi, N_i - neurony

Sieci ze sprzężeniem zwrotnym – działanie sieci Hopfielda

Stan neuronu w sieci Hopfielda wyraża się wzorem:

$$s_i(k) = \sum_{j=1}^n (w_{ij} * y_j(k-1) - \Theta_i + x_i(k)) \quad (18)$$

a sygnał wyjściowy:

$$y_i(k) = \begin{cases} -1 & \text{dla } s_i(k) < 0 \\ y_i(k-1) & \text{dla } s_i(k) = 0 \\ 1 & \text{dla } s_i(k) > 0 \end{cases} \quad (19)$$

y_i – sygnał wyjściowy, Θ_i – próg zadziałania neuronu (często traktowany jako dodatkowa waga), w_{ij} – waga, $s_i(k)$ – stan neuronu w chwili k , $x_i(k)$ – sygnał wejściowy do neuronu.

Sieci ze sprzężeniem zwrotnym – działanie sieci Hopfielda

Sieć neuronowa ze sprzężeniem zwrotnym staje się układem dynamicznym, obdarzonym pamięcią swojego stanu. Stan ten może się zmienić na skutek działania sygnałów zewnętrznych (wejścia X oraz wejść od innych neuronów). Oczywiście wpływ na stan sieci mają też wagi.

Decydują one o powstaniu stanów stabilnych sieci, do których sieć dąży po podaniu jej na wejścia pewnych wartości.

Sieć taką najczęściej opisuje się używając pewnej funkcji analogicznej do energii układu dynamicznego, zwanej także funkcją Lapunowa:

$$E(s) = -\frac{1}{2} * \sum_{i=1}^n \sum_{j=1}^n w_{ij} * y_i * y_j + \sum_{i=1}^n \Theta_i * y_i \quad (20)$$

y_i – stan neuronu, Θ_i - próg zadziałania neuronu, n – liczba neuronów

Minima takiej funkcji można traktować jako zapamiętane przez sieć wzorce, o ile zapiszemy je przez odpowiedni dobór wag.

Sieć Hopfielda można potraktować jak pamięć skojarzeniową adresowaną zawartością (a dokładnie „fragmentami” zawartości).

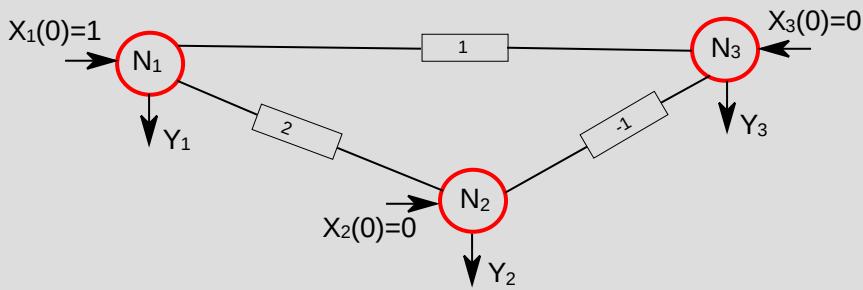
Sieci ze sprzężeniem zwrotnym – działanie sieci Hopfielda

Sieć Hopfielda może działać:

- synchronicznie, wtedy jest siecią z zegarem i wszystkie neurony jednocześnie zmieniają swój stan
- asynchronicznie, wtedy tylko jedne neuron (np. wylosowany) może zmienić swój stan.

Po podaniu na wejście sieci (z nauczonymi wzorcami) pewnego sygnału, sieć dochodzi w kilku krokach do stanu równowagi „energetycznej” odnajdując wzorzec najbardziej pasujący do zadanego sygnału. Sygnał wejściowy może być niekompletny, zaszumiony, przekłamany (oczywiście w pewnych granicach), a sieć i tak odnajduje właściwy wzorzec.

Sieci ze sprzężeniem zwrotnym – symulacja działania sieci Hopfielda



$$s_i(k+1) = \sum_{j=1}^n (w_{ij} * y_j(k) - \Theta_i + x_i(k))$$

$$y_i(k) = f(k) = \begin{cases} -1 & \text{dla } s_i(k) < 0 \\ y_i(k-1) & \text{dla } s_i(k) = 0 \\ 1 & \text{dla } s_i(k) > 0 \end{cases}$$

Symulacja zachowania sieci Hopfielda z rysunku powyżej.

Stan wejściowy $X_1(0)=1$, $X_2(0)=0$, $X_3(0)=0$, $Y_1(0)=1$, $Y_2(0)=-1$, $Y_3(0)=-1$, $\Theta_1=\Theta_2=\Theta_3=0$. W każdym kroku jest aktualizowany tylko 1 neuron (po kolej), pozostałe się nie zmieniają.

1. $Y_1(1) = f((-1)*1 + (-1)*2 + 1) = f(-2) = -1$
2. $Y_2(2) = f((-1)*2 + (-1)*(-1) + 0) = f(-1) = -1$
3. $Y_3(3) = f((-1)*1 + (-1)*(-1) + 0) = f(0) = Y_3(2) = Y_3(0) = -1$
4. $Y_1(4) = f((-1)*1 + (-1)*2 + 1) = f(-2) = -1$
5. $Y_2(5) = f((-1)*2 + (-1)*(-1) + 0) = f(-1) = -1$
6. $Y_3(6) = f((-1)*1 + (-1)*(-1) + 0) = f(0) = Y_3(3) = -1$

Stany neuronów nie zmieniły się – sieć osiągnęła stan stabilny.

Sieci ze sprzężeniem zwrotnym – uczenie sieci Hopfielda

Sieć Hopfielda można uczyć metodą wzorowaną na regule Hebla:

$$w_{ij} = \frac{1}{n} \sum_{m=1}^p x_{im} * x_{jm} \quad (21)$$

Jak widać wzorce można zapisać „na raz” bez długotrwałego uczenia.

Istnieje też przyrostowa reguła Storky'ego:

$$\Delta w_{ij}(k) = \frac{1}{n} x_i(k) * x_j(k) - \frac{1}{n} x_{im}(k) * h_{ji}(k) - \frac{1}{n} x_j(k) * h_{ij}(k) \quad (22)$$

$$h_{ij}(k) = \sum_{l=1, l \neq i, j}^n w_{il} * x_l(k) \quad (23)$$

w_{ij} , Δw_{ij} – waga (zmiana wagi) między neuronami i, j , x_{im} – wzorzec m dla i -tego neuronu, p – liczba wzorców, n – liczba neuronów, k – chwila czasowa.

Reguła Storky'ego umożliwia nauczenie sieci większej liczby wzorców niż reguła Hebla.

Sieć Hopfielda - pojemność pamięci

Sieć Hopfielda użyta jako pamięć asocjacyjna ma pojemność

$$p_{max} = n / (2 \log n) \approx 0,138n \quad (24)$$

n – liczba neuronów.

Sieć Hopfielda -

przykład zastosowania do rozwiązania problemu TSP

Ponieważ stan sieci Hopfielda opisuje się funkcją analogiczną do energii (20), a sama sieć ma właściwość samorzutnej minimalizacji tej energii, to w takim razie można spróbować „zakodować” w tej funkcji energii jakiś problem minimalizacji (a po niewielkich modyfikacjach także i maksymalizacji) i odpowiednio dobrać wagi tak, aby wykorzystać tę naturalną właściwość sieci do rozwiązania tego typu problemów.

Sieć Hopfielda -

przykład zastosowania do rozwiązania problemu TSP

Wobec tego należy zbudować sieć o wielkości N^2 neuronów (gdzie N to liczba miast w rozwiązywanym problemie) składających się z N sekcji, w których tylko jeden neuron może mieć wartość jeden, a numer sekcji będzie określać, jako które z kolei miasto reprezentowane przez neuron o wartości 1 będzie odwiedzone (kod 1 z N). Pamiętając, że wyjścia z sieci są binarne, to można łatwo zastosować takie zakodowanie problemu.

Miasto\Kolejność	1	2	3	4
Miasto 1	0	0	1	0
Miasto 2	0	1	0	0
Miasto 3	0	0	0	1
Miasto 4	1	0	0	0

Rozwiązanie zakodowane w tabeli reprezentuje drogę: Miasto 4 – Miasto 2 – Miasto 1 – Miasto 3 – Miasto 4.

Sieć Hopfielda -

przykład zastosowania do rozwiązania problemu TSP

Do zbudowania sieci potrzebna jest funkcja celu, zawierająca również człony odpowiadające karze za ewentualne naruszenie ograniczeń:

- odwiedzenie wszystkich miast ($C/2^*$...),
- tylko jeden neuron o wartości 1 w „wierszu” (reprezentacja jak w tablicy na poprzednim slajdzie)), czyli tylko 1 miasto odwiedzone w każdym kroku sieci ($B/2^*$),
- oraz bez powtórzeń miast na trasie ($A/2^*$...) - tylko 1 miasto w „kolumnie”,
- człon ($D/2^*$...) to minimalizowana odległość w problemie TSP.

$$E(x) = \frac{A}{2} * \sum_{p=1}^N \sum_{j=1}^N \sum_{i=1}^N x_{pi} * x_{pj} + \frac{B}{2} * \sum_{i=1}^N \sum_{p=1}^N \sum_{q=1}^N x_{pi} * x_{qi} + \frac{C}{2} * \left(\sum_{p=1}^N \sum_{i=1}^N x_{pi} - N \right)^2 + \frac{D}{2} * \sum_{p=1}^N \sum_{q=1}^N \sum_{i=1}^N d_{pq} * x_{pi} * (x_{q,i+1} + x_{q,i-1}) \quad (25)$$

x_{pi} – neuron reprezentujący miasto p na pozycji i trasy (może mieć wartość 0 lub 1); A, B, C, D – pewne dodatnie stałe, d_{pq} – odległość między miastami p i q , N – liczba miast i jednocześnie liczba neuronów w „wierszu” i „kolumnie” sieci.

Sieć Hopfielda -

przykład zastosowania do rozwiązania problemu TSP

Wagi w sieci są symetryczne i są dobierane następująco:

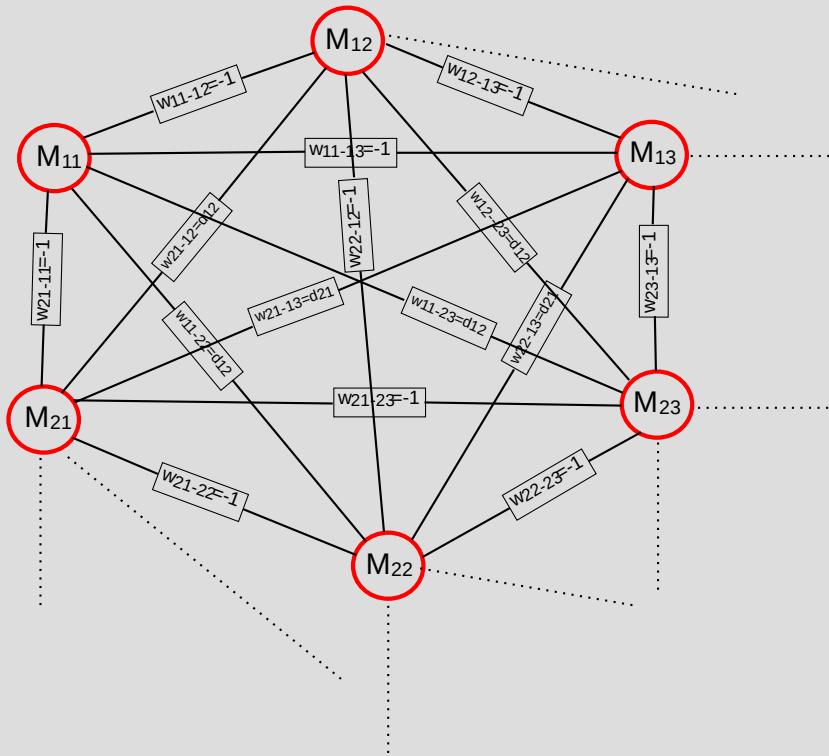
- wagi między neuronami z kolejnych „kolumn” (czyli kolejnych etapów podróży) poza będącymi na tej samej pozycji są równe odległościom między miastami d_{pq} ;
- wagi w obrębie tej samej kolumny lub między tymi samymi miastami w różnych kolumnach są niewielkimi liczbami ujemnymi, realizując ujemne sprzężenia zwrotne, które opisują ograniczenia na tylko jedną 1 w wierszu i kolumnie.

Sieć inicjowana jest N jedynkami jako sygnałami wejściowymi.

W podobny sposób można rozwiązywać także inne problemy optymalizacyjne klasy NP.

Sieć Hopfielda -

przykład zastosowania do rozwiązywania problemu TSP



Fragment symetrycznej sieci Hopfielda służącej do rozwiązywania problemu TSP

Sieć Hopfielda - zastosowania

Sieć Hopfielda jest używana:

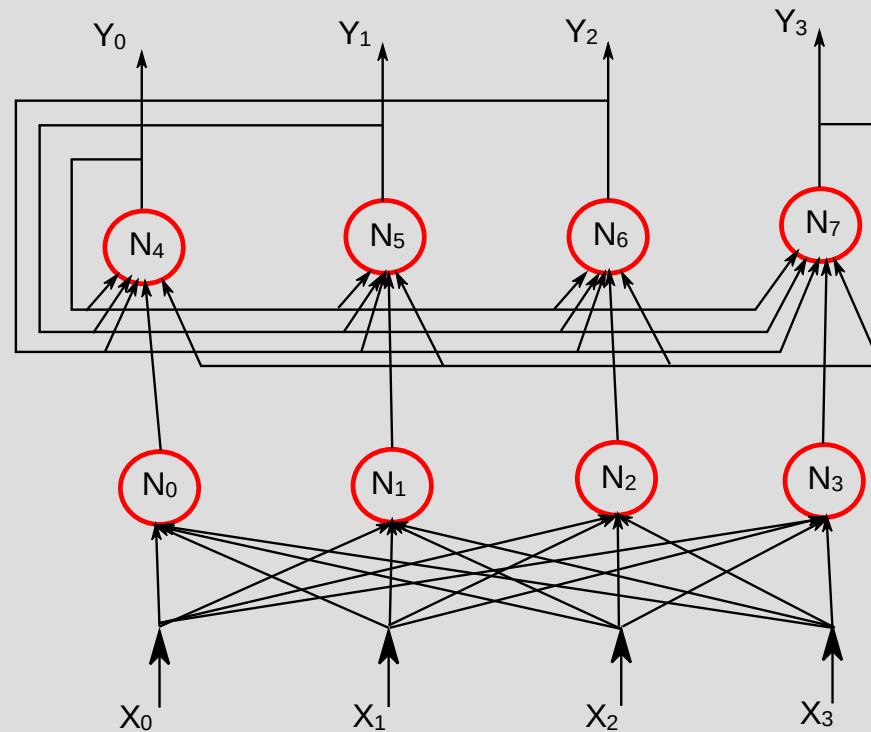
- jako pamięć asocjacyjna (adresowana zawartością);
- jako optymalizator w problemach optymalizacji kombinatorycznej (zadania klasy NP typu problem plecakowy, problem komiwojażera...);
- w przetwarzaniu obrazów;

Sieć Hamminga

Sieć Hamminga jest rozszerzeniem sieci Hopfielda. Rozszerzenie polega na dodaniu do sieci Hopfielda jednej warstwy (typu perceptronowego) realizującej obliczanie odległości Hamminga (jest to np. liczba bitów, którą różnią się dwa wzorce, sieć działa na sygnałach binarnych) podanego sygnału od najbliższego wzorca, co ma zwiększyć pojemność pamięciową całej sieci. W praktyce wygląda to tak, że po podaniu sygnału wejściowego powinno aktywować się tylko jedno wyjście z warstwy wejściowej, co pomaga odróżnić wzorce, a ich pamiętaniem zajmuje się warstwa wyjściowa o strukturze zbliżonej do sieci Hopfielda. Czasem sieć ta ma jeszcze jedną warstwę na wyjściu, która odpowiada nie tylko zapamiętaną kategorią, ale też jej najbardziej charakterystycznym przedstawicielem i na wyjściu ma warstwę podobną do jednowarstwowego perceptronu.

Ilustruje to rysunek pokazany na następnym slajdzie.

Sieć Hamminga



Sieć Hamminga

Sieć Hamminga może zapamiętać tyle wzorców, ile jest neuronów w warstwie wyjściowej (rekurencyjnej). Jest to pewien postęp w stosunku do sieci Hopfielda, ale okupiony dołożeniem kolejnej warstwy, w której jest najczęściej drugie tyle neuronów co w warstwie rekurencyjnej.

Wagi w warstwie wejściowej są ustawiane według wzorów:

$$w_{ij} = x_i^j \text{ i } \Theta_j = N/2 \quad (26)$$

Wartość wyjścia neuronów warstwy wejściowej oblicza się na podstawie znanego wzoru opisującego neuron McCullocha-Pittsa:

$$U_j = f\left(\sum_{i=0}^{N-1} w_{ij} * x_i - \Theta_j\right) \quad (27)$$

w_{ij} – waga od wejścia i do j -tego neuronu warstwy wejściowej, x_i^j – i -ty bit j -tego wzorca wejściowego (każdy neuron warstwy wejściowej ma reagować na jeden wybrany wzorzec, dlatego indeks j odnosi się do neuronów i wzorców), Θ_j – próg aktywacji neuronów warstwy wejściowej; N – liczba wejść do sieci, $f(\dots)$ - funkcja aktywacji, najczęściej skokowa .

Sieć Hamminga

Wagi neuronów od warstwy wejściowej do wyjściowej ustawiane są na wartość 1 (sygnał bez zmian).

W warstwie rekurencyjnej dobór wag jest dość prosty, określa go poniższy wzór:

$$w_{kl} = \begin{cases} 1 & \text{dla } k=l \\ -\varepsilon & \text{dla } k \neq l, \varepsilon < \frac{1}{M} \end{cases} \quad (28)$$

Neurony w warstwie wyjściowej nie mają progu aktywacji

$$Y_k = F\left(\sum_{l=1}^M w_{kl} * U_l\right) \quad (29)$$

w_{kl} – waga między neuronami k i l w warstwie wyjściowej, M – liczba neuronów w warstwie wyjściowej, U_l – wejścia z warstwy wejściowej, $F(\dots)$ - funkcja aktywacji neuronów warstwy wyjściowej (rekurencyjnej).

Sygnał wejściowy z warstwy wejściowej doprowadzany jest do sieci rekurencyjnej tylko na jedną iterację, następnie jest odłączany, a sieć iteruje aż znajdzie się w stanie równowagi, a jej wyjścia podadzą wynik.

Sieć Hamminga -

zastosowanie

Sieć Hamminga jest stosowana jako:

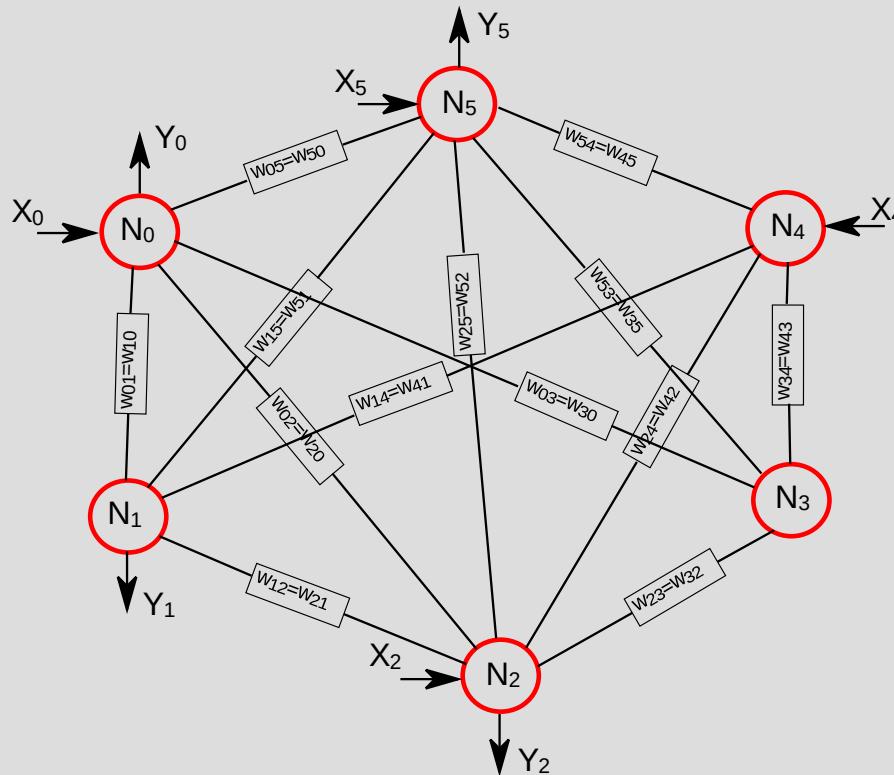
- klasyfikator (w literaturze podkreśla się bardzo dobre jej właściwości w tej dziedzinie – może dobrze sklasyfikować wzorce przekłamane nawet o 43%)
- pamięć asocjacyjna o lepszej pojemności niż sieć Hopfielda
- maszyna wnioskująca i baza wiedzy w systemach ekspertowych.

Maszyna (sieć) Boltzmanna

Maszyna (lub sieć) Boltzmanna to kolejne uogólnienie sieci Hopfielda. Jest to sieć rekurencyjna ze sprzężeniami zwrotnymi o symetrycznych wagach, która zawiera neurony ukryte. Jednocześnie sieć taka jest stochastyczna, gdyż neurony posiadają prawdopodobieństwa osiągnięcia pewnego stanu określone rozkładem Boltzmanna. Inspiracją do powstania tej sieci był algorytm symulowanego wyżarzania, w którym również wykorzystuje się rozkład Boltzmanna, a inspiracja polegała na tym, że neurony mogą tu działać nieco podobnie do cząstek, które osiągają stan minimum energetycznego, zależnego od temperatury.

Architektura takiej sieci jest podobna do sieci Hopfielda – każdy neuron połączony jest z każdym (mogą istnieć bezpośrednie połączenia z wyjścia na wejście tego samego neuronu), choć oczywiście niektóre wagi mogą być zerowe (brak połączenia), jednak nie do wszystkich neuronów dochodzą sygnały wejściowe, nie wszystkie widoczne są też na wyjściu. Te, które nie są widoczne ani od strony wejścia, ani od strony wyjścia, są neuronami ukrytymi.
Ilustruje to rysunek przedstawiony na następnym slajdzie.

Maszyna Boltzmanna



W sieci Boltzmanna mogą występować neurony ukryte, na rysunku jest to N₃.

Maszyna Boltzmanna

Neurony w tej sieci mogą przyjmować stan $s_i = \{-1, 1\}$ zgodnie z rozkładem prawdopodobieństwa $p(h_i)$ (rozkład Fermiego-Diraca):

$$p(h_i) = \frac{1}{1 + \exp(-2 * \beta * h_i)} = \frac{1}{1 + \exp(-2 * \frac{h_i}{T})} \quad h_i = \sum_{j=0}^n w_{ij} s_j \quad (30)$$

w_{ij} – waga między neuronami i i j , n – liczba neuronów, s_j – stan neuronu j , T – parametr rozkładu będący analogią do temperatury (można go modyfikować w trakcie działania sieci, zmieniając nieco jej działanie).

Rozkład prawdopodobieństwa stanów dla całej sieci opisywany jest rozkładem Fermiego-Diraca (uogólnienie rozkładu Boltzmanna dla układu dyskretnego), a jej właściwości również opisuje się funkcją energetyczną o postaci zbliżonej do tej, która opisuje sieć Hopfielda, jednak należy pamiętać, że w tym przypadku jest to funkcja probabilistyczna. Regułę uczenia sieci wyprowadza się różniczkując funkcję energii (która pełni tu rolę kryterium jakości działania sieci) po wagach analogicznie jak dla sieci BP i otrzymuje się stochastyczny wariant metody BP, którego wzorów, ze względu na ich skomplikowanie, nie będę przytaczał. Dla zainteresowanych można znaleźć je w pracy J. Hertz, A. Krogh, R. Palmer „Wstęp do teorii obliczeń neuronowych”, WNT, 1993.

Maszyna Boltzmana -

właściwości i zastosowania

Generalnie maszyna Boltzmana jest siecią bardzo skomplikowaną i trudną w realizacji, uczeniu i wykorzystaniu, doczekała się niewielu realizacji praktycznych.

Jej zastosowania są podobne do sieci Hopfielda:

- pamięć asocjacyjna
- element doradczy w systemach ekspertowych
- klasyfikacja.

Sieć Kohonena

Sieć Kohonena jest przykładem sieci samoorganizującej się (zwanej także mapą samoorganizującą się), uczącej się bez nadzoru. Jej zadaniem jest mapowanie danych wielowymiarowych (wymiarowość danych, to w uproszczeniu liczba wejść do sieci) do przestrzeni o wymiarowości niższej, zazwyczaj dwuwymiarowej. Działanie sieci polega na grupowaniu przykładów jej prezentowanych.

W fazie uczenia sieć generuje nowe zgrupowania danych (dlatego dane trenujące muszą być reprezentatywne dla problemu), a w fazie działania przyporządkowuje prezentowane dane do którejś z wytworzonych w czasie uczenia grup.

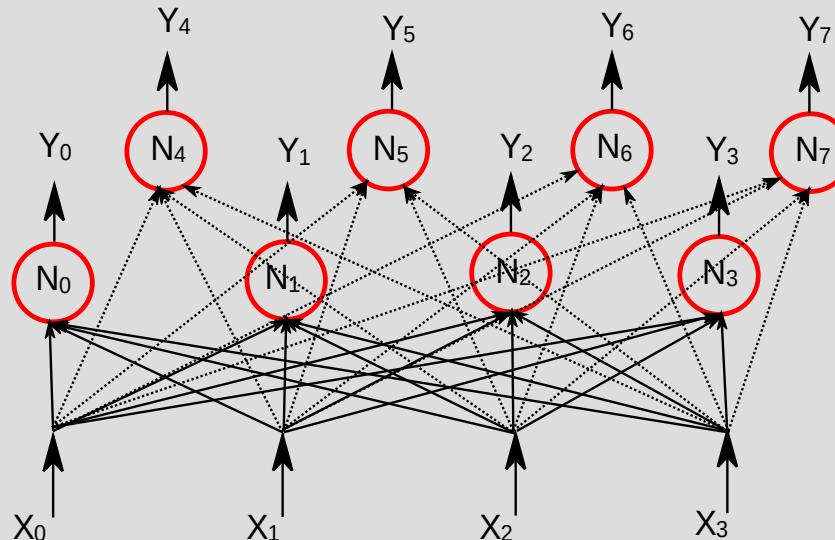
Sieć wykrywa podobieństwa w prezentowanych danych do wytworzonych samodzielnie w trakcie uczenia wzorców i przydziela je do odpowiednich grup tak, aby jak najlepiej pasowały do wytworzonych wzorców. Odpowiedzią sieci jest „odpalenie” jednego neuronu, który reprezentuje wzorzec najbliższy prezentowanego. Sieć działa według zasady WTA - „Winner Takes All” - „zwycięzca bierze wszystko” (czasem również WTM „zwycięzca bierze prawie wszystko”).

Sieć Kohonena -

architektura

Sieć Kohonena jest siecią jednokierunkową, składa się z dwóch warstw:

- warstwy wejściowej, dystrybuującej sygnały wejściowe do wszystkich neuronów
- warstwy wyjściowej – topologicznej, neurony są połączone tylko z warstwą wejściową (istnieją zbliżone sieci, w których realizowana jest idea hamowania obocznego, realizująca model WTA, tu jednak zwycięzcę wykrywa „siła wyższa”, czyli komputer sterujący siecią).



Sieć Kohonena -

działanie

W sieci Kohonena wyjście neuronu oblicza się według tradycyjnego wzoru:

$$h_i = \sum_{j=0}^n w_{ij} x_j \quad (31)$$

w_{ij} – waga między neuronem i a wejściem j , x_j – sygnał na wejściu j , h_i – sygnał na wyjściu i . jednakże na wyjście przekazywana jest wartość 1 tylko dla najlepszego neuronu (ten najlepszy wybiera układ sterujący siecią – owa „siła wyższa”). Ponieważ najczęściej sieć neuronowa jest realizowana przy użyciu standardowego komputera, to możliwe jest obliczanie różnicy między wejściem, a wektorem wag bez „zabawy” w model neuronu.

Sieć Kohonena - uczenie

Uczenie sieci Kohonena nie jest skomplikowane:

$$\Delta w_{ij}(k+1) = \begin{cases} \alpha(k) * (x_j - w_{ij}(k)) & \text{dla } i \in \Omega \\ 0 & \text{dla } i \notin \Omega \end{cases} \quad (32)$$

w_{ij} – waga między neuronem i i wejściem j , x_j – sygnał na wejściu j , Ω - otoczenie zwycięskiego neuronu (z nim samym włącznie), $\alpha(k)$ – współczynnik uczenia, malejący w funkcji czasu. Otoczenie zwycięskiego neuronu bywa definiowane rozmaicie, najczęściej jest to funkcja tzw. meksykańskiego kapelusza o postaci:

$$\%idzeta_i(k) = \begin{cases} 1 & \text{dla } r_i(k)=0 \\ \frac{\sin(a * r_i(k))}{a * r_i(k)} & \text{dla } |r_i(k)| \in (0, 2\pi/a) \\ 0 & \text{dla } |r_i(k)| > 2\pi/a \end{cases} \quad (33)$$

r_{il} - oznacza odległość między neuronem i a zwycięzcą (jest zależny od czasu), a – parametr regulujący zasięg funkcji.

W takim przypadku wzór na modyfikację wag można uprościć do:

$$\Delta w_{ij}(k+1) = \alpha(k) * \%idzeta_i(k) * (x_j - w_{ij}(k)) \quad (34)$$

Sieć Kohonena -

zastosowania

Sieć Kohonena używana jest do:

- klasyfikacji danych
- odwzorowywanie przestrzeni danych
- klasteryzacja danych.

Niestety działanie sieci, a szczególnie jej uczenie nie należy do szybkich.

Sieci neuronowe -

podsumowanie

Sieci neuronowe były pierwszymi praktycznie wykorzystanymi urządzeniami sztucznej inteligencji. Ich użycie budziło ogromne emocje, gdyż wydawało się, że droga do zbudowania sztucznego mózgu jest już bardzo łatwa i krótka. Jednak praktyka nie okazała się aż tak prosta. Mimo, że powstają coraz nowsze rozwiązania w tej dziedzinie: nowe sieci, rozwiązania hybrydowe różnych sieci, a także różnych dziedzin SI z sieciami neuronowymi, to jednak droga do budowy sztucznego umysłu wydaje się jeszcze bardzo daleka. I w tej dziedzinie sieci na pewno nie spełniły pokładanych w nich nadziei, jednakże są z powodzeniem stosowane w tzw. „miękkiej sztucznej inteligencji” do przetwarzania informacji w zadaniach trudnych do wykonania innymi metodami.

Sieć ART

Sieć ART nazywana jest tak od pojęcia adaptacyjnej teorii rezonansu (*ang. Adaptive Resonance Theory*) i jest kolejnym przykładem sieci uczącej się bez nauczyciela. Sieć klasyfikuje dochodzące do niej przykłady, zgodnie z nauczonymi wcześniej samodzielnie klasami, lecz jeśli napotka nowy przykład, który jej zdaniem nie jest podobny do wyuczonych wcześniej (decyduje o tym tzw. próg czujności sieci τ), to zapamiętuje go jako wzorzec nowej klasy danych.

Sieć ART

Sieć ART jest przykładem sieci dwuwarstwowej, które to warstwy przez wzajemne oddziaływanie decydują o przydzielaniu nowej danej do jednej z znanych klas lub utworzeniu nowej kategorii danych. Liczby neuronów w obu warstwach są dość dobrze określone przez przetwarzane dane. Warstwa wejściowa ma tyle neuronów ile jest sygnałów wejściowych, np dla obrazu o rozdzielczości $n \times m$ będzie to $n \times m$ neuronów. W warstwie drugiej neuronów powinno być nieco więcej niż spodziewanych kategorii danych.

Sieć jest cały czas w fazie uczenia i działania jednocześnie, musi wobec tego posiadać cały czas nadmiarowe neurony do wykorzystania na ewentualne nowe klasy danych.

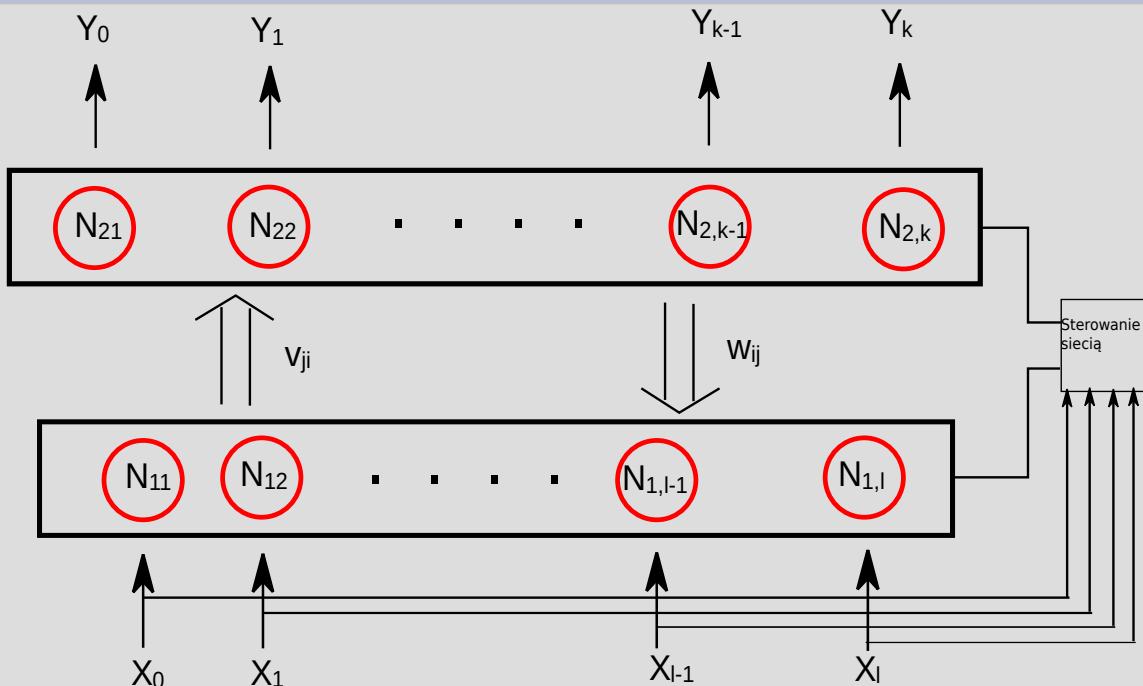
Modyfikując wartość parametru czuwania można nieco modyfikować sposób działania sieci on-line, zmniejszając lub powiększając zdolność do tworzenia nowych kategorii danych.

Sieć ART1, ART2, ...

Powstało kilka odmian sieci ART o nazwach odpowiednio ART1, ART2 i ART3. Różnią się one tym, że pierwsza, nieco prostsza, działa na sygnałach binarnych, pozostałe na analogowych lub dyskretnych, lecz na pewno mających więcej niż 2 wartości poziomu sygnału wejściowego, dodatkowo sieć ART3 zapewnia większą stabilność działania w porównaniu do ART2.

W dalszym ciągu zajmiemy się siecią binarną ART1, co jednak nie powinno wpływać negatywnie na zrozumienie sposobu działania tych sieci.

Sieć ART



Schemat budowy sieci ART.

$N_{11} \dots N_{1,l}$ - neurony warstwy wejściowej; $N_{21} \dots N_{2,k}$ - neurony warstwy wyjściowej - konkurencyjnej;
 X_0, \dots, X_l – sygnały wejściowe, Y_0, \dots, Y_k – sygnały wyjściowe, v_{ji} – wagi z warstwy wejściowej do wyjściowej, w_{ij} – wagi z warstwy wyjściowej do wejściowej, Sterowanie siecią – synchronizuje i kontroluje działania warstw, wykrywa obecność nowego sygnału na wejściu i inicjuje działanie sieci.

Sieć ART

działanie

1. Sieć otrzymuje na wejście nowy sygnał X , warstwa 2 jest zerowana, otrzymuje od warstwy 1 niezmieniony obraz X .
2. Neurony warstwy 2 obliczają swoje wyjścia, zwycięża ten, który daje najwyższy sygnał wyjściowy (jego wagi w są najbardziej zbliżone do obrazu) inne neurony są gaszone przez połączenia hamujące.
3. Obliczany jest iloraz między sygnałem wejściowym, a zwrotnym z warstwy 2 (od zwycięskiego neuronu oznaczonego *),

$$D = \frac{\sum_{i=1}^l w_{ij}^* x_i}{\sum_{i=1}^l x_i} \quad (35)$$

a następnie:

- jeśli sygnały są prawie identyczne (decyduje o tym porównanie D z progiem czujności sieci τ , czyli $D < \tau$), to sieć jest w rezonansie, następuje uaktualnienie wag, aby zmniejszyć różnice zapamiętanego wzorca i otrzymanego sygnału;
- jeśli sygnały są różne, to zostaje przydzielony do zwycięskiego neuronu, który staje się reprezentantem nowej klasy danych, jego wagi są odpowiednio korygowane.

Sieć ART

działanie

4. Następnie sieć oczekuje na nowy sygnał, po otrzymaniu którego wraca do pkt. 1.

W trakcie pracy sieci istotne jest działanie układu sterującego, którego zadaniem jest wykrywanie stanów wejścia oraz stanów sieci i odpowiednie reagowanie na nie, przez zmianę aktywności działań warstw. Oczywiście przy komputerowej symulacji sieci, steruje nią odpowiednio napisany program.

Sieć ART

działanie

1. Wagi są inicjowane następująco:

$$w_{ij}(0) = 1 \quad (36)$$

$$v_{ji}(0) = \frac{1}{1+l} \quad (37)$$

2. Po zakończeniu cyklu pracy sieci (wytypowaniu zwycięzcy lub zwycięzcy i nowej kategorii) wagi są korygowane następująco:

$$w_{ij}^*(t+1) = w_{ij}^*(t) * x_i \quad (38)$$

$$v_{ji}^*(t+1) = \frac{w_{ij}^*(t+1)}{0,5 + \sum_{j=1}^k w_{ij}^*(t+1) * x_i} \quad (39)$$

w_{ij}^* – waga zwycięskiego neuronu z warstwy 2 do 1; v_{ij}^* – waga do zwycięskiego neuronu z warstwy 1 do 2; l – liczba neuronów w warstwie 1; k – liczba neuronów w warstwie 2; $t, t+1$ – kolejne chwile czasowe/kroki działania sieci.

Sieci z rodziny ART są stosowane przede wszystkim jako klasyfikatory nieznanych danych, które bez wcześniejszej nauki sieć podzieli na kategorie podobnych danych. O liczbie wyznaczonych kategorii zadecyduje zadaną wartość parametru τ .

Sieci neuronowe

podsumowanie

Sieci neuronowe były pierwszymi praktycznie wykorzystanymi urządzeniami sztucznej inteligencji. Ich użycie budziło ogromne emocje, gdyż wydawało się, że droga do zbudowania sztucznego mózgu jest już bardzo łatwa i krótka. Jednak praktyka nie okazała się aż tak prosta. Mimo, że powstają coraz nowsze rozwiązania w tej dziedzinie: nowe sieci, rozwiązania hybrydowe różnych sieci, a także różnych dziedzin SI z sieciami neuronowymi, to jednak droga do budowy sztucznego umysłu wydaje się jeszcze bardzo daleka. I w tej dziedzinie sieci na pewno nie spełniły pokładanych w nich nadziei. Jednakże sieci neuronowe są z powodzeniem stosowane w tzw. „miękkiej sztucznej inteligencji” do przetwarzania informacji w zadaniach trudnych do wykonania innymi metodami.

Sieci neuronowe na pewno nie doszły do końca swoich możliwości, o czym świadczy np. kolejny obecnie występujący „boom” na ich wykorzystanie w dziedzinie przetwarzania obrazów związany z tzw. głębokim uczeniem i wykorzystaniem sieci do:

- kolorowania zdjęć i filmów;
- poprawiania jakości i rozdzielczości archiwalnych materiałów filmowych;
- efektywnego przetwarzania obrazów w innych zagadnieniach, np. ich rozpoznawaniu.

Dziękuję za uwagę!