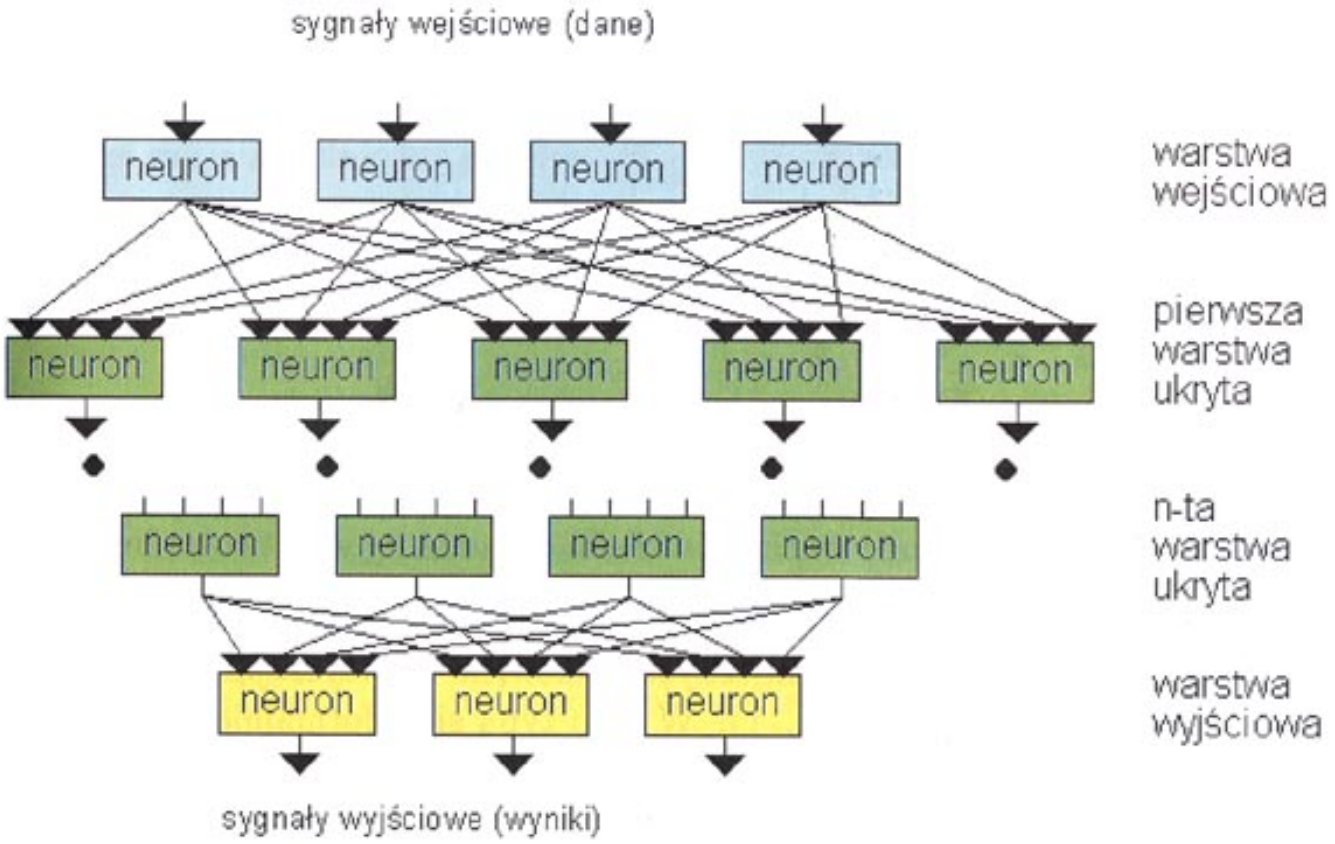
Sztuczne Sieci Neuronowe

dr inż. Jacek Czerniak

jczerniak@ukw.edu.pl

Schemat budowy sieci neuronowej



Do czego stosuje się SSN?

⇒ Bardzo znanym zastosowaniem sieci neuronowej jest wykorzystany przez firmę *Eaton Corporation* układ sterowania pozwalający na wspomaganie kierowcy dużej ciężarówki (pięć osi, osiemnaście kół) przy wykonywaniu niektórych szczególnie trudnych manewrów (cofanie z naczepą).

⇒ NASA wykorzystuje sieci neuronowe do sterowania ramieniem robota, którego zadaniem jest chwytanie przedmiotów znajdujących się w dowolnym położeniu (dotyczy to między innymi manipulatora działającego w ładowni promów kosmicznych, gdzie w warunkach nieważkości szczególnie trudno jest utrzymać manipulowane obiekty w stałym położeniu, a pewność chwytu jest uzależniona od stopnia zgodności orientacji przestrzennej osi przedmiotu i elementów chwytaka.

wała dla potrzeb *US Navy* oparty na sieciach neuronowych system klasyfikujący i rozpoznający sygnały sonarowe. System ten pozwala rozpoznawać szumy charakterystyczne dla napędów różnych typów statków i okrętów. Udało się tak dokładnie wytrenować sieć, że jest ona zdolna do rozróżniania między sobą szumów różnych okrętów tego samego typu, co pozwala na identyfikację nazwy jednostki, a także potrafi rozpoznawać za pomocą hydrofonów niektóre dźwięki nadwodne - na przykład szum wytwarzany przez helikopter unoszący się nad powierzchnią oceanu.

⇒ Sieci neuronowe często stosowane są w energetyce - zwłaszcza do prognozowania zapotrzebowania na moc elektryczną. Jedno z bardziej znanych wdrożeń w tym zakresie dotyczy elektrowni *BC Hydro w Vancouver*, szczególnie trudnej do sterowania, ponieważ jej cztery turbiny mają wyraźnie różniące się charakterystyki i zadanie ich włączania i wyłączania w odpowiedzi na zmienne zapotrzebowanie na moc elektryczną ze strony odbiorców energii jest wyjątkowo złożone i niewdzięczne.

⇒ Wielki producent rakiet, firma *General Devices Space Systems Division* użyła sieci neuronowych do sterowania pracą 150 zaworów doprowadzających paliwo i tlen do silników rakiety Atlas. Okazało się, że odpowiednio wytrenowana sieć potrafiła w oparciu dane na temat aktualnych fluktuacji mocy silnika sterować zaworami trafniej, niż poprzednio stosowany, kosztowny i zawodny złożony system automatyki oparty na wykorzystaniu danych z setek sensorów.

⇒ W szpitalu *Anderson Memorial Hospital* w Południowej Karolinie sieci wykorzystano do realizacji procesu optymalizacji leczenia, zyskując (jak podają publikacje) miliony dolarów oszczędności i ratując życie kilkudziesięciu ludzi. Niestety, bliższe dane na temat tego ciekawego przedsięwzięcia nie są znane.

Do czego stosuje się SSN?

⇒ Bardzo znanym zastosowaniem sieci neuronowej jest wykorzystany przez firmę *Eaton Corporation* układ sterowania pozwalający na wspomaganie kierowcy dużej ciężarówki (pięć osi, osiemnaście kół) przy

⇒ Wielki producent rakiet, firma *General Devices Space Systems Division* użyła sieci neuronowych do sterowania pracą 150 zaworów doprowadzających paliwo i tlen do silników rakiety Atlas. Okazało się, że odpowiednio wytrenowana sieć potrafiła w oparciu dane na temat aktualnych fluktuacji mocy silnika sterować zaworami trafniej, niż poprzednio stosowany, kosztowny i zawodny złożony system automatyki oparty na wykorzystaniu danych z setek sensorów.

drofonów niektóre dźwięki nadwodne - na przykład szum wytwarzany przez helikopter unoszący się nad powierzchnią oceanu.

Do czego stosuje się SSN?

Takich i podobnych doniesień o udanych zastosowaniach sieci neuronowych znaleźć można w najnowszym piśmiennictwie bez liku. Żeby nie przedłużać nadmiernie tego wątku wymienię jeszcze tylko kilka dalszych zastosowań w skrócie.

- ⇒ Amerykańskie siły powietrzne (US Air Force) stosują sieci neuronowe do rozwoju symulatorów lotu.
- ⇒ Do sieci neuronowych odwołał się koncern Ford Motor Company przygotowując nowy system diagnostyczny dla swoich silników.
- ⇒ Firma Halliburton stosuje sieci neuronowe do identyfikacji typu skał napotykanych podczas prowadzenia odwiertów poszukiwawczych przy poszukiwaniu złóż ropy i gazu.
- ⇒ Linie TWA stosują sieci neuronowe do poszukiwania bomb w swoim terminalu na lotnisku JFK w Nowym Yorku.

Uczenie sieci neuronowych

Jak można uczyć sieci neuronowe?

Cykl działania sieci neuronowej podzielić można na etap nauki, kiedy sieć gromadzi informacje potrzebne jej do określenia, co i jak ma robić, oraz na etap normalnego działania (nazywany czasem także egzaminem), kiedy w oparciu o zdobytą wiedzę sieć musi rozwiązywać konkretne nowe zadania.

Możliwe są dwa warianty uczenia: **z nauczycielem** i **bez nauczyciela**. Uczenie z nauczycielem polega na tym, że sieci podaje się przykłady poprawnego działania, które powinna ona potem naśladować w swoim bieżącym działaniu (w czasie egzaminu).

Proces (schemat) uczenie

- Dane trenujące TRE
- Dane testujące TES
- Określić funkcję błędu
- Graniczne wartości błędów na TRE i TES
- Inicjalizacja początkowych wag i progu
- Adaptacja wag i progu to proces iteracyjny, w którym działa algorytm adaptacji, z nawrotami na zbiorze TRE a sprawdzany na TES, aż błędy są poniżej progów

Uczenie perceptronu

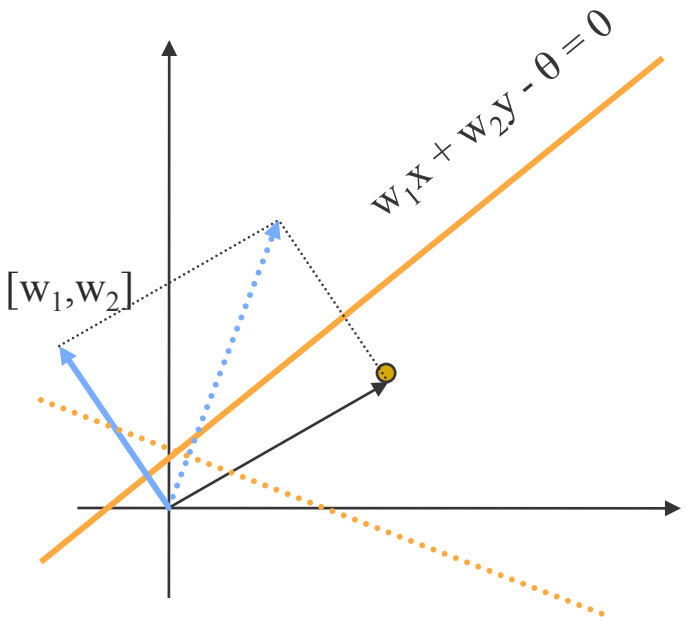
- **Wejście:**
  - Ciąg przykładów uczących ze znanymi odpowiedziami
- **Proces uczenia:**
  - Inicjujemy wagi losowo
  - Dla każdego przykładu, jeśli odpowiedź jest nieprawidłowa, to

$$w_1 += \alpha x_1$$

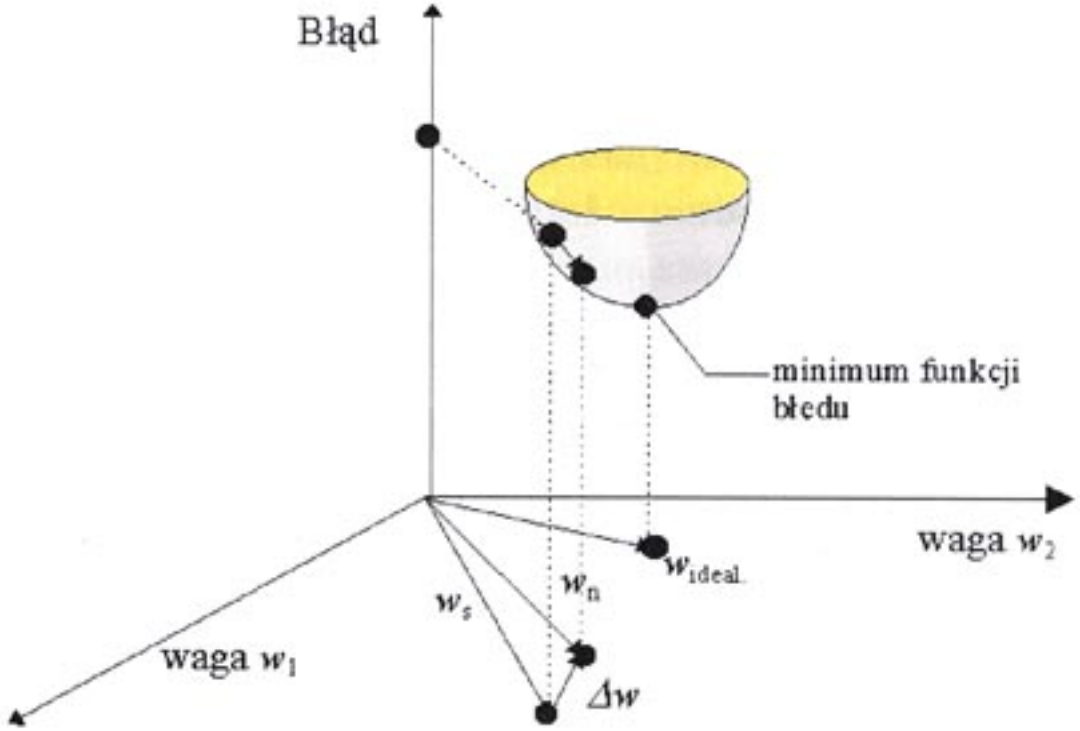
$$w_2 += \alpha x_2$$

$$\theta -= \alpha$$

gdzie α jest równe różnicy między odpowiedzią prawidłową a otrzymaną.



Zmiana wektora wag podczas uczenia



Samouczenie się sieci

Obok opisanego wyżej schematu uczenia z nauczycielem jest w użyciu także szereg metod tak zwanego uczenia bez nauczyciela (albo **samouczenia** sieci). Metody te polegają na podawaniu na wejście sieci wyłącznie szeregu przykładowych danych wejściowych, bez podawania jakiejkolwiek informacji dotyczącej pożądanych czy chociażby tylko oczekiwanych sygnałów wyjściowych. Okazuje się, że odpowiednio zaprojektowana sieć neuronowa potrafi wykorzystać same tylko obserwacje wejściowych sygnałów i zbudować na ich podstawie sensowny algorytm swojego działania - najczęściej polegający na tym, że automatycznie wykrywane są klasy powtarzających się (być może z pewnymi odmianami) sygnałów wejściowych i sieć uczy się (zupełnie spontanicznie, bez jawnego nauczania) rozpoznawać te typowe wzorce sygnałów.

Reguła Delta

Reguła delta polega na tym, że każdy neuron otrzymawszy na swoich wejściach określone sygnały (z wejść sieci albo od innych neuronów, stanowiących wcześniejsze piętra przetwarzania informacji) wyznacza swój sygnał wyjściowy wykorzystując posiadaną wiedzę w postaci wcześniej ustalonych wartości współczynników wzmocnienia (wag) wszystkich wejść oraz (ewentualnie) progu. Sposoby wyznaczania przez neurony wartości sygnałów wyjściowych na podstawie sygnałów wejściowych omówione zostały dokładniej w poprzednim rozdziale. Wartość sygnału wyjściowego, wyznaczonego przez neuron na danym kroku procesu uczenia porównywana jest z odpowiedzią wzorcową podaną przez nauczyciela w ciągu uczącym. Jeśli występuje rozbieżność (a na początku procesu uczenia prawie zawsze taka rozbieżność się pojawi, bo skąd niby ten neuron ma wiedzieć, czego my od niego chcemy?) - neuron wyznacza różnicę pomiędzy swoim sygnałem wyjściowym a tą wartością sygnału, która była by - według nauczyciela prawidłowa. Ta różnica oznaczana jest zwykle symbolem greckiej litery δ (delta) i stąd nazwa opisywanej metody.

Delta c.d.

Sygnał błędu (delta) wykorzystywany jest przez neuron do korygowania posiadanej przez niego wiedzy - na tym właśnie polega proces nauki. Konkretnie neuron dokonuje modyfikacji swoich współczynników wagowych (i ewentualnie progu), stosując następujące dwie proste reguły:

⇒ wagi zmieniane są tym silniej, im większy błąd został wykryty;

⇒ wagi związane z tymi wejściami, na których występowały duże wartości sygnałów wejściowych, zmieniane są bardziej, niż wagi wejść, na których sygnał wejściowy był niewielki.

Metoda Delta - formalnie

Omawianie algorytmów uczenia z nauczycielem zaczniemy od pokazania sposobu uczenia pojedynczego (nieliniowego) neuronu. Podstawowym wzorem, wykorzystywanym podczas uczenia sieci neuronowych jest wzór:

$$\nabla w_i^{(j)} = \eta \delta^{(j)} \frac{d\varphi(e)}{de^{(j)}} x_i^{(j)}$$

z którego wynika, że zmiana $\nabla w_i^{(j)}$ współczynnika wagi $w_i^{(j)}$ na i-tym wejściu neuronu po pokazaniu j-tego obiektu ciągu uczącego (patrz ramka 7) jest proporcjonalna do błędu $\delta^{(j)}$ popełnionego przez neuron na tym etapie procesu uczenia:

$$\delta^{(j)} = z^{(j)} - y^{(j)},$$

gdzie oczywiście

$$y^{(j)} = \varphi\left(\sum_{i=0}^{n} w_i^{(j)} x_i^{(j)}\right)$$

Delta c.d.

Jak z tego wynika, jeśli neuron podczas uczenia nie popełnia błędów, jego wagi nie są zmieniane. W ten sposób proces uczenia zatrzymuje się samorzutnie po osiągnięciu sukcesu. Poprawka współczynnika wagi na określonym wejściu $\nabla w_i^{(j)}$ proporcjonalna jest także do wielkości sygnału na tym wejściu $x_i^{(j)}$, co powoduje, że zmianom i aktualizacji podczas uczenia podlegają wagi tylko tych wejść, które są aktywne. Współczynnik η zwany jest współczynnikiem uczenia (*learning rate*) i wpływa na szybkość procesu uczenia.

Aby łatwo można było łatwo obliczyć pochodną $\frac{d\varphi(e)}{de^{(j)}}$ przyjmuje się zwykle jako $\varphi(e)$ funkcję logistyczną

$$y = \varphi(e) = \frac{1}{1 + \exp(-\beta e)}$$

dla której

$$\frac{d\varphi(e)}{de^{(j)}} = y^{(j)}\left(1 - y^{(j)}\right)$$

Delta c.d.

W związku z tym ostatecznie otrzymujemy formułę uczenia dla pojedynczego neuronu (i dla dowolnej sieci **jednowarstwowej**) w postaci:

$$\nabla w_i^{(j)} = \eta \left(z^{(j)} - y^{(j)} \right) \left(1 - y^{(j)} \right) x_i^{(j)} y^{(j)}$$

W przypadku sieci wielowarstwowych sytuacja jest inna i dlatego dopiero w oddzielnej ramce przeniesiemy omówioną tu zasadę uczenia na sieć wielowarstwową.

Metoda Momentum – aby przyspieszyć

Metoda *momentum* polega na nadaniu procesowi uczenia pewnej bezwładności - zmiany wag zależą zarówno od błędów popełnianych przez sieć w danym kroku, jak i od przebiegu procesu uczenia w krokach wcześniejszych co można zapisać w postaci:

$$\nabla w_i^{(m)(j)} = \eta_1 \delta_m^{(j)} \frac{d\varphi(e)}{de_m^{(j)}} y_i^{(j)} + \eta_2 \nabla w_i^{(m)(j-1)}$$

Parametr η_2 przyjmuje wartości od 0 do 1; często stosuje się wartość 0.9.

Inne algorytmy uczenia

Ze względu na szybkość uczenia poszukuje się więc odmiennych (od klasycznego opisywanego tu algorytmu delta) nowych metod uczenia sieci. Uważa się, że czas uczenia sieci algorytmem delta rośnie wykładniczo ze wzrostem liczby elementów sieci, co może stanowić w przyszłości jeden z zasadniczych czynników limitujących rozmiary budowanych sieci. To bardzo pesymistyczne oszacowanie, dominuje jednak pogląd, że nie musi ono dotyczyć algorytmów "konkurencyjnych" dla reguły delta. Najczęściej wskazywane są tu algorytmy PNN (*Probabilistic Neural Network*), RBF (*Radial Basis Functions*), LVQ (*Linear Vector Quantization*) i RCE (*Restricted Coulomb Energy*). Algorytmy te są jednak chwilowo mało popularne (większość gotowych programów modelujących sieci neuronowe opiera się na algorytmie delta i jego odmianach), a równocześnie wzmiankowane metody są dość złożone, nie będą więc tu omawiane.

Uczenie warstw ukrytych

Istota problemu pojawiającego się przy uczeniu neuronów warstwy ukrytej polega na tym, że dla neuronów tych nie da się bezpośrednio wyznaczyć wielkości błędów (*delta*), ponieważ nauczyciel podaje wzorcowe wartości sygnałów jedynie dla warstwy wyjściowej, natomiast sygnałów wyznaczonych przez neurony warstwy ukrytej (też zapewne na początku uczenia błędnych) nie ma z czym porównać. Pewnym rozwiązaniem dla tego problemu jest metoda tak zwanej wstecznej propagacji błędu (*backpropagation*), zaproponowana po raz pierwszy przez Paula Werbosa a potem spopularyzowana przez Davida Rumelharta i wiązana zwykle z jego imieniem.

Metoda propagacji wstecznej

Metoda ta polega na odtwarzaniu przypuszczalnych wartości błędów głębszych (ukrytych) warstw sieci na podstawie rzutowania wstecz błędów wykrytych w warstwie wyjściowej. Z grubsza można powiedzieć (w tej chwili - bo w dalszych rozdziałach książki proces ten poddany będzie bardzo szczegółowej analizie) robi się to w ten sposób, że rozważając każdy kolejny neuron warstwy ukrytej bierze się pod uwagę błędy wszystkich tych neuronów, do których wysyłał on swój sygnał wyjściowy i **sumuje się je**, uwzględniając wielkości współczynników wag połączeń między rozważanym neuronem a neuronami, których błędy są sumowane. W ten sposób "odpowiedzialność" za błędy neuronów warstwy ukrytej obciąża neuron warstwy ukrytej tym silniej, im silniej (bo z większą wagą) wpływał sygnał danego neuronu ukrytego na wypracowanie określonych sygnałów warstwy ukrytej (czyli na powstanie określonych błędów).

Backpropagation - formalnie

Algorytm opisany nie daje się zastosować dla sieci wielowarstwowej, gdyż w przypadku sieci wielowarstwowych nie można bezpośrednio określić wymaganych odpowiedzi sygnałów wyjściowych $z^{(j)}$ dla warstw ukrytych sieci, a tym samym określić błąd $\delta^{(j)}$. Dlatego w odniesieniu do neuronów tzw. warstw ukrytych także stosuje się wzór

$$\nabla w_i^{(m)(j)} = \eta \delta_m^{(j)} \frac{d\varphi(e)}{de_m^{(j)}} y_i^{(j)}$$

(gdzie m jest numerem rozważanego neuronu) ale zamiast prostego wzoru:

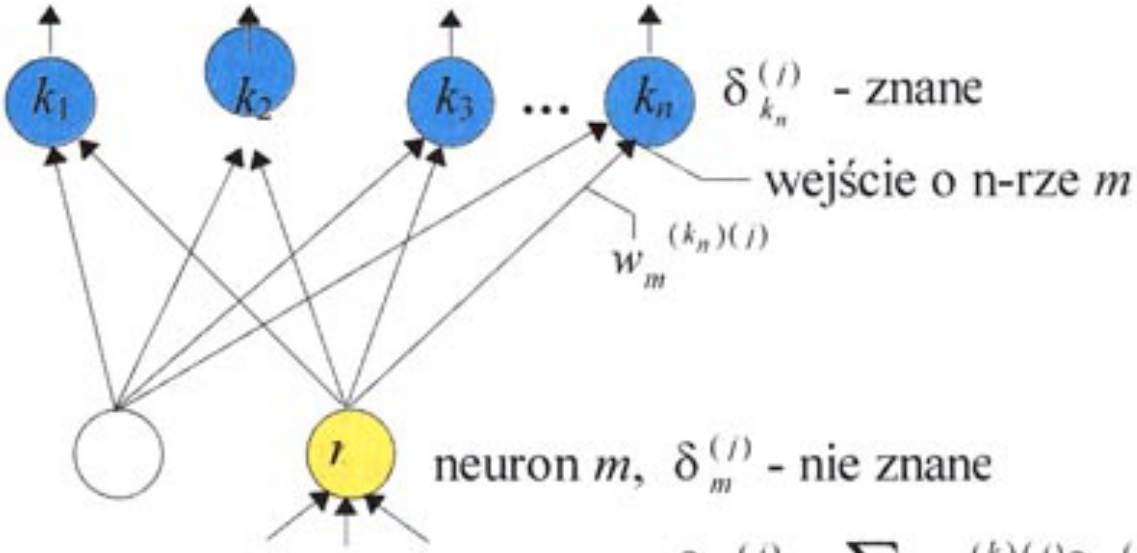
$$\delta_m^{(j)} = z_m^{(j)} - y_m^{(j)}$$

Backpropagation c.d.

musimy przeprowadzić następujące rozumowanie. Niech rozważany neuron należący do warstwy ukrytej wysyła sygnał wyjściowy wyłącznie do neuronów warstwy wyjściowej, dla których wartości błędów $\delta_k^{(j)}$ można łatwo określić przez porównanie ich sygnałów z informacją pochodzącą od nauczyciela. Błąd popełniany przez ten neuron $\delta_m^{(j)}$ otrzymujemy sumując błędy $\delta_k^{(j)}$ i uwzględniając wartości współczynników wag połączeń między rozważanym neuronem a neuronami do których wysyła on sygnał wyjściowy

$$\delta_m^{(j)} seum_k \, w_m^{(k)(j)} \frac{d\varphi(e)}{de_k^{(j)}} \delta_k^{(j)}$$

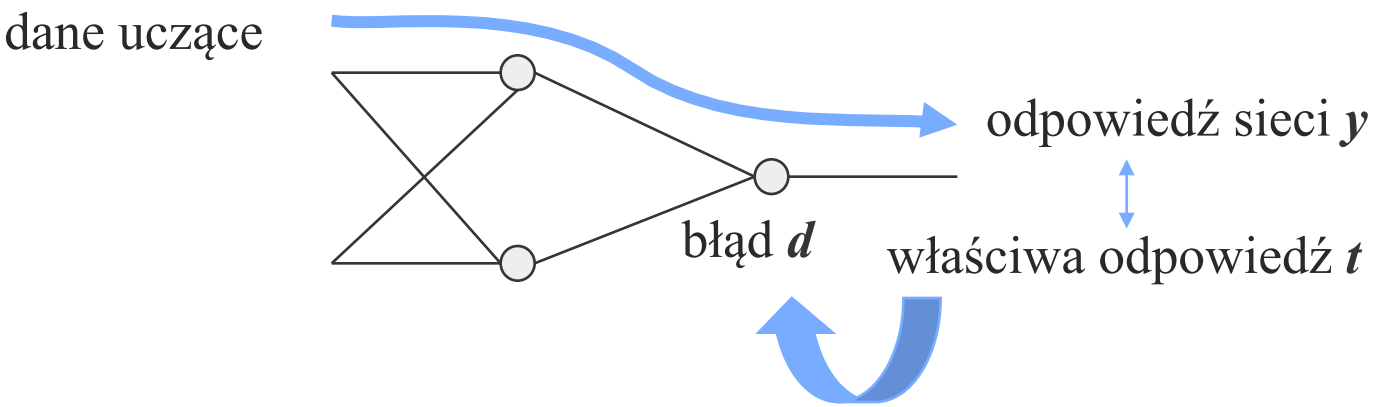
Wyjaśnienie oznaczeń



$$\delta_m^{(j)} = \sum_k w_m^{(k)(j)} \delta_k^{(j)}$$

$$\delta_m^{(j)} seum_k \; w_m^{(k)(j)} \frac{d\varphi(e)}{de_k^{(j)}} \delta_k^{(j)}$$

PROPAGACJA WSTECZNA (2)



Błąd sieci definiowany jest zazwyczaj jako

$$d = \frac{1}{2}(y - t)^2$$

PROPAGACJA WSTECZNA (3)

- Oznaczmy przez:
  - ***g: R → R*** – funkcję aktywacji w neuronie
  - $w_1, \ldots, w_K$ – wagi połączeń wchodzących
  - $z_1, \ldots, z_K$ – sygnały napływające do neuronu z poprzedniej warstwy
- Błąd neuronu traktujemy jako funkcję wag połączeń do niego prowadzących:

$$d(w_1, \ldots, w_K) = \frac{1}{2}\left(g\left(w_1 z_1 + \ldots + w_K z_K\right) - t\right)^2$$

Wzór adaptacyjny schematu propagacji wstecznej błędu

$$w_j(k+1) - w_j(k) = -\eta \frac{\partial d\left(w_1, \ldots, w_K\right)}{\partial w_j}$$

$gdzie$

$k - krok$

PRZYKŁAD (1)

- Rozpatrzmy model, w którym:
  - Funkcja aktywacji przyjmuje postać

$$g(s) = \frac{1}{1 + e^{-3(s+2)}}$$

  - Wektor wag połączeń = [1;-3;2]
- Załóżmy, że dla danego przykładu:
  - Odpowiedź powinna wynosić ***t*** = 0.5
  - Z poprzedniej warstwy dochodzą sygnały [0;1;0.3]

PRZYKŁAD (2)

- Liczymy wejściową sumę ważoną:

$$s = w_1 x_1 + w_2 x_2 + w_3 x_3 = 1 \cdot 0 + (-3) \cdot 1 + 2 \cdot 0.3 = -2.4$$

- Liczymy odpowiedź neuronu:

$$y = g(s) = \frac{1}{1 + e^{-3(-2.4+2)}} = \frac{1}{1 + e^{1.2}} \approx 0.23$$

- Błąd wynosi:

$$d = \frac{1}{2}(0.23 - 0.5)^2 \approx 0.036$$

IDEA ROZKŁADU BŁĘDU

- Musimy „rozłożyć” otrzymany błąd na połączenia wprowadzające sygnały do danego neuronu
- Składową błędu dla każdego ***j***-tego połączenia określamy jako pochodną cząstkową błędu względem ***j***-tej wagi
- Składowych tych będziemy mogli użyć do zmodyfikowania ustawień poszczególnych wag połączeń

OBLICZANIE POCHODNEJ

$$\frac{\partial d(w_1,\ldots,w_K)}{\partial w_j} = (y-t)\cdot g'(s)\cdot z_j$$

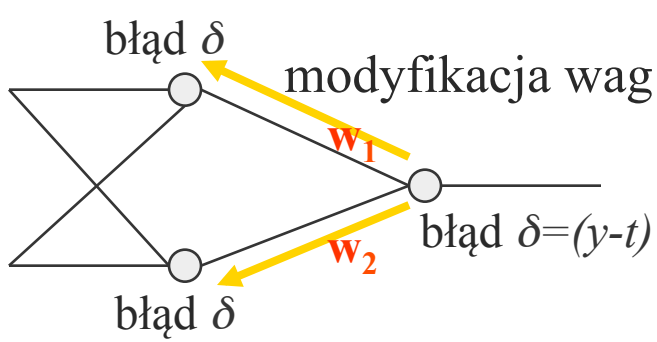
$$= \frac{\partial \frac{1}{2}\left(g(w_1 z_1 + \ldots + w_K z_K) - t\right)^2}{\partial w_j}$$

$$= \frac{\partial \frac{1}{2}(y-t)^2}{\partial y} \cdot \frac{\partial g(s)}{\partial s} \cdot \frac{\partial(w_1 z_1 + \ldots + w_K z_K)}{\partial w_j}$$

PROPAGACJA WSTECZNA (4)

- Idea:
  - Wektor wag połączeń powinniśmy przesunąć w kierunku przeciwnym do wektora gradientu błędu (z pewnym współczynnikiem uczenia η)
- Realizacja:

$$\Delta w_j = \eta \cdot (y - t) \cdot g'(s) \cdot z_j$$



Błędy są następnie propagowane w kierunku poprzednich warstw, tzn. zamiast $(y-t)$ wstawiamy:

$$\delta = \sum_{i=1}^{n} w_i \delta_i$$

czyli neuron w warstwie ukrytej "zbiera" błąd z neuronów, z którymi jest połączony (uwzględniając wagi sprzed modyfikacji).

Backpropagation - koniec

Jak widzimy rzutowane wstecznie błędy są mnożone przez te same współczynniki, przez które mnożone były sygnały przy propagowaniu sygnału w kierunku od wejścia do wyjścia.

Opisaną zasadę można zastosować do całej sieci, przy zachowaniu odpowiedniej kolejności obliczania błędów.

Uczenie bez nauczyciela

Przy uczeniu bez nauczyciela (zwane też samouczeniem sieci) nie podaje się prawidłowych odpowiedzi. Sieć musi sama wydobyć interesujące kategorie lub cechy w zbiorze danych wejściowych. Samouczenie sieci jest analogią do działania ludzkiego mózgu. Człowiek również ma zdolność do spontanicznego klasyfikowania napotkanych obiektów i zjawisk, a później do rozpoznawania nowych obiektów jako należących do jednej z wcześniej wyuczonych klas.

Sieci samouczące się

Otóż okazuje się, że jest w pełni możliwe i nawet stosunkowo łatwo osiągalne spontaniczne samoorganizowanie się sieci neuronowej - przy spełnieniu kilku prostych warunków. Najprostsza koncepcja samouczenia sieci oparta jest na spostrzeżeniu fizjologa i psychologa amerykańskiego, **Donalda Hebba**, że w mózgu zwierząt (i w umyśle człowieka) zachodzą procesy wzmacniania połączeń między neuronami (lub całymi ich zespołami, tak zwanymi ośrodkami), pod warunkiem, że zostały one pobudzone do pracy **równocześnie**. W ten sposób powstają skojarzenia, tak (zdaniem Hebba) kształtują się odruchy, tak również formują się proste formy umiejętności ruchowych i percepcyjnych.

Samouczenie c.d.

Przenosząc teorie Hebba na grunt sieci neuronowych informatycy zaprojektowali metodę samouczenia polegającą na tym, że sieci pokazuje się kolejne przykłady sygnałów wejściowych, **nie podając żadnych informacji o tym, co z tymi sygnałami należy zrobić.** Sieć po prostu obserwuje otoczenie i odbiera różne sygnały, nikt jej jednak nie mówi, jakie znaczenie mają pokazujące się obiekty i jakie są pomiędzy nimi zależności. Sieć na podstawie obserwacji występujących sygnałów stopniowo sama odkrywa, jakie jest ich znaczenie i również sama ustala zachodzące między sygnałami zależności. W ten sposób sieć nie tylko **nabywa** wiedzę - ona ją poniekąd sama **tworzy!**

Samouczenie c.d.

Po podaniu do sieci neuronowej każdego kolejnego zestawu sygnałów wejściowych tworzy się w tej sieci pewien **rozkład** sygnałów wyjściowych - niektóre neurony sieci są pobudzone bardzo silnie, inne słabiej, a jeszcze inne mają sygnały wyjściowe wręcz ujemne. Interpretacja tych zachowań może być taka, że niektóre neurony "rozpoznają" podawane sygnały jako "własne" (czyli takie, które są skłonne akceptować), inne traktują je "obojętnie", zaś jeszcze u innych neuronów wzbudzają one wręcz "awersję". Po ustaleniu się sygnałów wyjściowych wszystkich neuronów w całej sieci - wszystkie wagi wszystkich neuronów są zmieniane, przy czym wielkość odpowiedniej zmiany wyznaczana jest na podstawie **iloczynu sygnału wejściowego**, wchodzącego na dane wejście (to którego wagę zmieniamy) **i sygnału wyjściowego** produkowanego przez neuron, w którym modyfikujemy wagi.

Samouczenie c.d.

Dokładniejsza analiza procesu samouczenia metodą Hebba pozwala nam stwierdzić, że w wyniku konsekwentnego stosowania opisanego algorytmu początkowe, najczęściej przypadkowe "preferencje" neuronów ulegają systematycznemu wzmacnianiu i dokładnej polaryzacji. Jeśli jakiś neuron miał "wrodzoną skłonność" do akceptowania sygnałów pewnego rodzaju - to w miarę kolejnych pokazów nauczy się te sygnały rozpoznawać coraz dokładniej i coraz bardziej precyzyjnie. Po dłuższym czasie takiego samouczenia w sieci powstaną zatem (całkiem spontanicznie!) **wzorce** poszczególnych **typów** występujących na wejściu sieci sygnałów. W wyniku

Samouczenie c.d.

tego procesu sygnały podobne do siebie będą w miarę postępu uczenia coraz skuteczniej **grupowane** i **rozpoznawane** przez pewne neurony, zaś inne typy sygnałów staną się "obiektem zainteresowania" innych neuronów. W wyniku tego procesu samouczenia sieć - całkiem sama, tylko obserwując podawane do niej sygnały - nauczy się, ile klas podobnych do siebie sygnałów pojawia się na jej wejściach oraz sama przyporządkuje tym klasom sygnałów neurony, które nauczą się je rozróżniać, rozpoznawać i sygnalizować. **Tylko** tyle i **aż** tyle!

Samouczenie - formalnie

Metoda samouczenia jest oparta na regule Hebba, która zakłada wzmacnianie połączeń między źródłami silnych sygnałów a neuronami, które na nie silnie reagują. W wyniku uczenia sygnały podobne do siebie będą rozpoznawane przez pewne neurony, zaś przez inne będą ignorowane (ujemne wartości wyjściowe). Po ustaleniu się sygnałów wyjściowych wszystkich neuronów w całej sieci wszystkie wagi wszystkich neuronów są zmieniane, przy czym wielkość odpowiedniej zmiany wyznaczana jest na podstawie iloczynu sygnału wejściowego, wchodzącego na dane wejście (to którego wagę zmieniamy) i sygnału wyjściowego neuronu. Opisując

Ciąg uczący

Jest to zbiór przykładów na podstawie których sieć uczy się. Ciąg ten ma postać:

$$U = \{X^{(1)}, X^{(2)}, \ldots, X^{(N)}\}$$

gdzie

$X^{(j)}$	jest wektorem danych wejściowych w j-tym kroku
N	jest liczbą posiadanych na etapie uczenia pokazów

Reguła Hebba

Regułę uczenia dla m-tego neuronu w j-tym kroku uczenia definiujemy w sposób następujący:

$$w_i^{(m)(j+1)} = w_i^{(m)(j)} + \eta x_i^{(m)} y_m^{(j)}$$

gdzie

$$y_m^{(j)} = \sum_{i=1}^{n} w_i^{(m)(j)} x_i^{(j)}$$

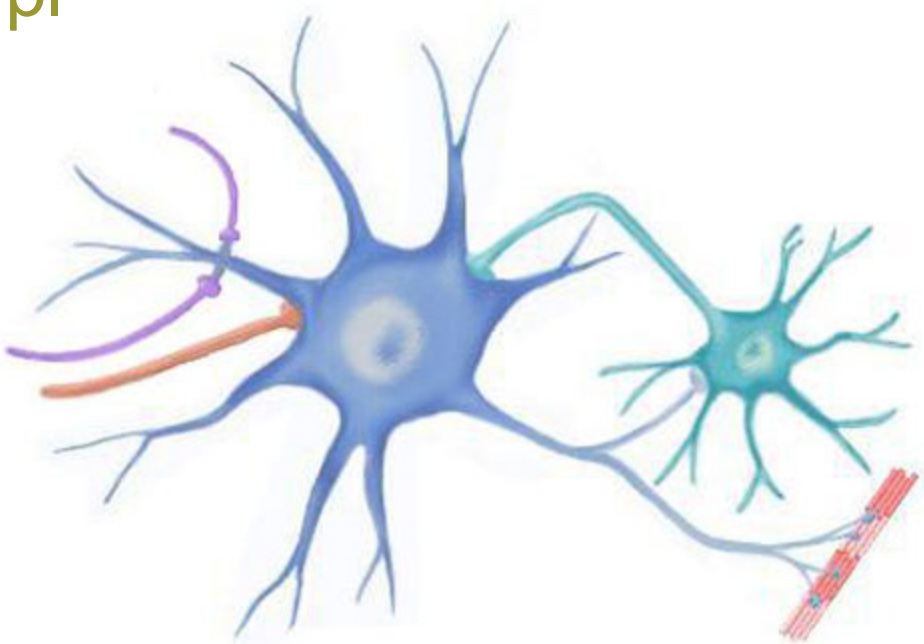
Inaczej uczenie wg. reguły Hebba nazywa się uczeniem korelacyjnym (*correlation learning*), gdyż zmierza ona do takiego dopasowania wag, aby uzyskać najlepszą korelację między sygnałami wejściowymi, a zapamiętanym w formie wartości wag "wzorcem" sygnału, na który określony neuron ma reagować.

Zalety z używania SSN

- Brak konieczności jawnego formułowania jakichkolwiek założeń dla modeli
- Możliwość rozwiązywania problemów nie mających dobrej definicji formalnej
- Możliwość zastępowania procesu programowania oraz „ręcznego” tworzenia modelu - procesem uczenia sieci
- Możliwość pracy współbieżnej

Literatura dotycząca SSN

- Bilski J., Świąć A. (2000): Metoda wstecznej propagacji błędów i jej modyfikacje. Biocybernetyka i inżynieria biomedyczna 2000 - sieci neuronowe, tom VI, Akademicka Oficyna Wydawnicza
- Findeisen W., Szymanowski J., Wierzbicki A. (1977): Teoria i metody obliczeniowe optymalizacji. PWN
- Herz J., Krogh A., Palmer R.G. (1995): Wstęp do teorii obliczeń neuronowych. WNT,
- Jędruch W. (1996): Sztuczne sieci neuronowe, Wydawnictwo Naukowe PWN
- Korbicz J., Obuchowicz A., Uciński D. (1994): Sztuczne sieci neuronowe - Podstawy i zastosowania. Akademicka Oficyna Wydawnicza PLJ,
- Master T., (1996): Sieci neuronowe w praktyce, WNT
- Niederliński A. (1983): Systemy i sterowanie: Wstęp do automatyki i cybernetyki technicznej. PWN
- Osowski S. (1996): Sieci neuronowe w ujęciu algorytmicznym, WNT,
- Osowski S. (2000): Sieci neuronowe do przetwarzania informacji, Oficyna Wydawnicza Politechniki Warszawskiej,
- Rutkowska D., Piliński M., Rutkowski L. (1997): Sieci neuronowe, algorytmy genetyczne i systemy rozmyte, Wydawnictwo Naukowe PWN
- Tadeusiewicz R. (1993): Sieci neuronowe. Akademicka Oficyna Wydawnicza RM,
- **Tadeusiewicz R. (1998): Elementarne wprowadzenie do techniki sieci neuronowych z przykładowymi programami, Akademicka Oficyna Wydawnicza PLJ,**
- Żurada J., Barski M., Jędruch W. (1996): Sztuczne sieci neuronowe. Podstawy teorii i zastosowania, Wydawnictwo PWN,

Dziękuję za uwagę

- Zapraszam na kolejny wykład ;-)
  - dr inż. Jacek Czerniak
  - jczerniak@ukw.edu.pl