METODY INŻYNIERII WIEDZY

SZTUCZNE SIECI NEURONOWE MLP

Adrian Horzyk

Akademia Górniczo-Hutnicza

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Katedra Automatyki i Inżynierii Biomedycznej

Laboratorium Biocybernetyki

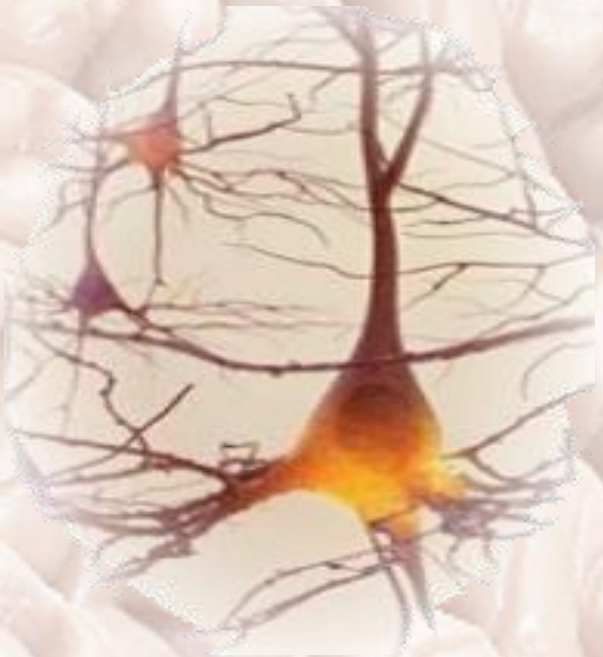
30-059 Kraków, al. Mickiewicza 30, paw. C3/205

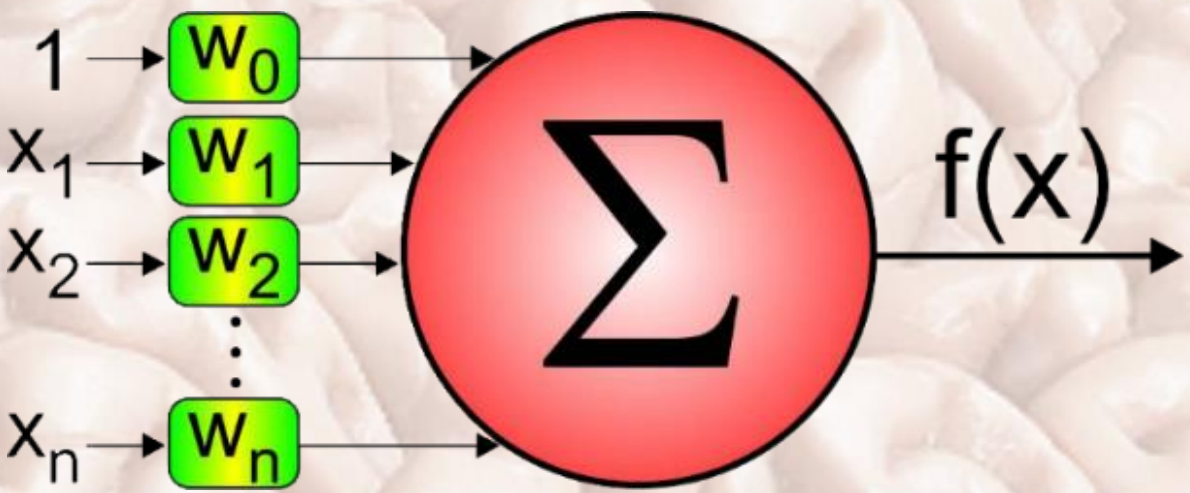
horzyk@agh.edu.pl, Google: Adrian Horzyk

MÓZG

- ✓ Plastyczny i dynamicznie zmienny twór
- ✓ Błyskawicznie reaguje na skomplikowane sytuacje
- ✓ Zdolny do adaptacji i samoorganizacji
- ✓ Zdolność do kompensacji po uszkodzeniach elementów
- ✓ Zdolny do równoległego przetwarzania danych
- ✓ Umożliwia uczenie się, adaptację i myślenie
- ✓ Uogólnia obiekty, fakty i reguły
- ✓ Kreatywnie reaguje w nowych kontekstach
- ✓ Aktywnie reprezentuje wiedzę
- ✓ Jest siedliskiem inteligencji

NEURONY BIOLOGICZNE I SZTUCZNE



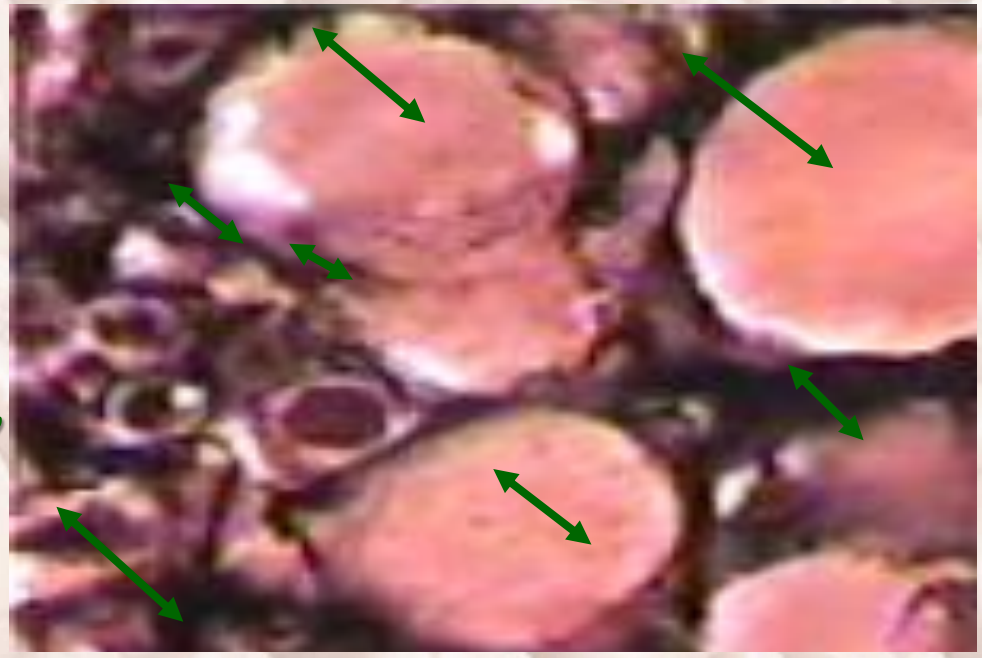


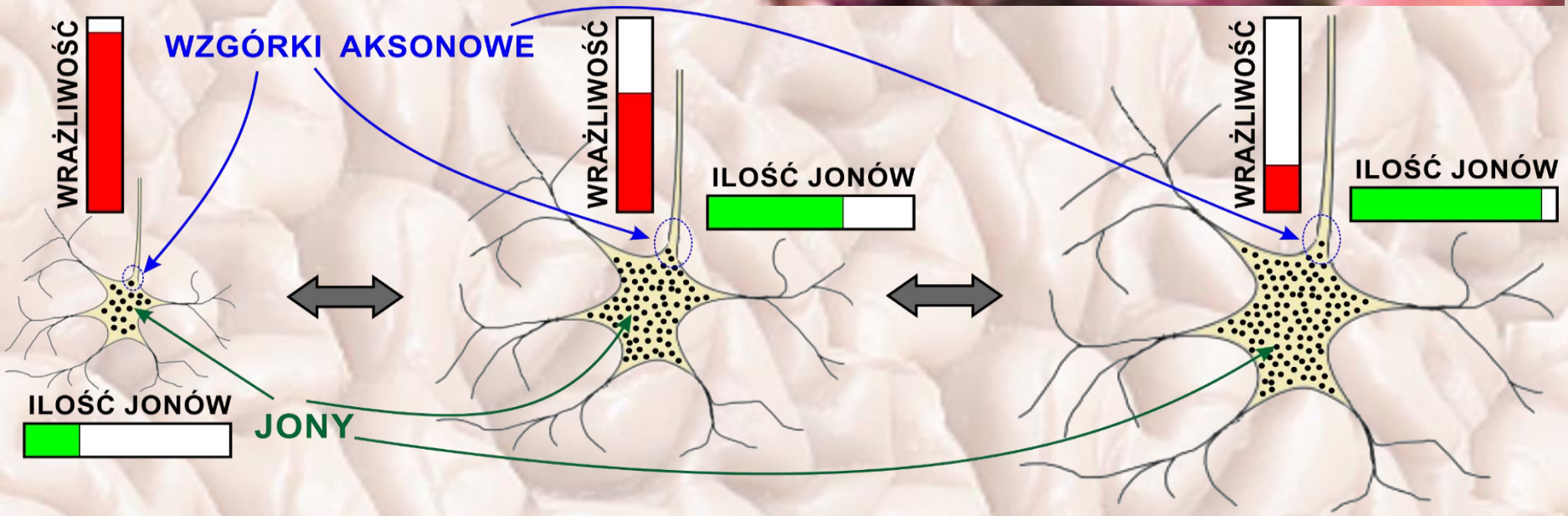
- Neurony nie tylko sumują ważone sygnały wejściowe.
- Podlegają procesom relaksacji i refrakcji w czasie.
- Neurony posiadają progi aktywacji i swoją wrażliwość.
- Reprezentują kombinacje, umożliwiające im selektywne oddziaływanie na inne neurony.

WIELKOŚĆ I PLASTYCZNOŚĆ NEURONÓW

Neurony biologiczne różnią się:

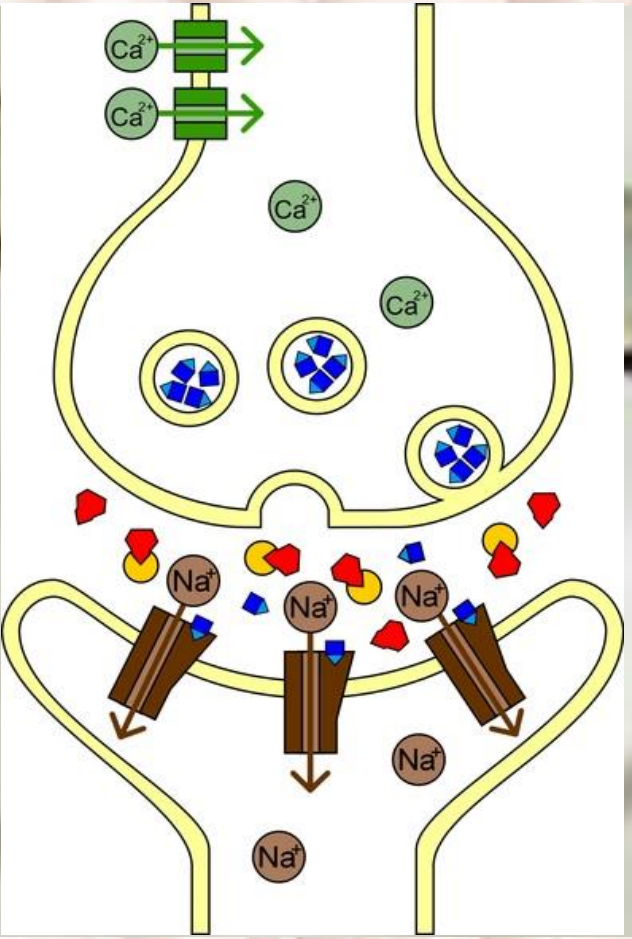
- wielkością i pojemnością,
- wrażliwością i reaktywnością,
- reprezentowanymi zbiorami kombinacji bodźców (danych),
- połączeniami
- i innymi cechami…

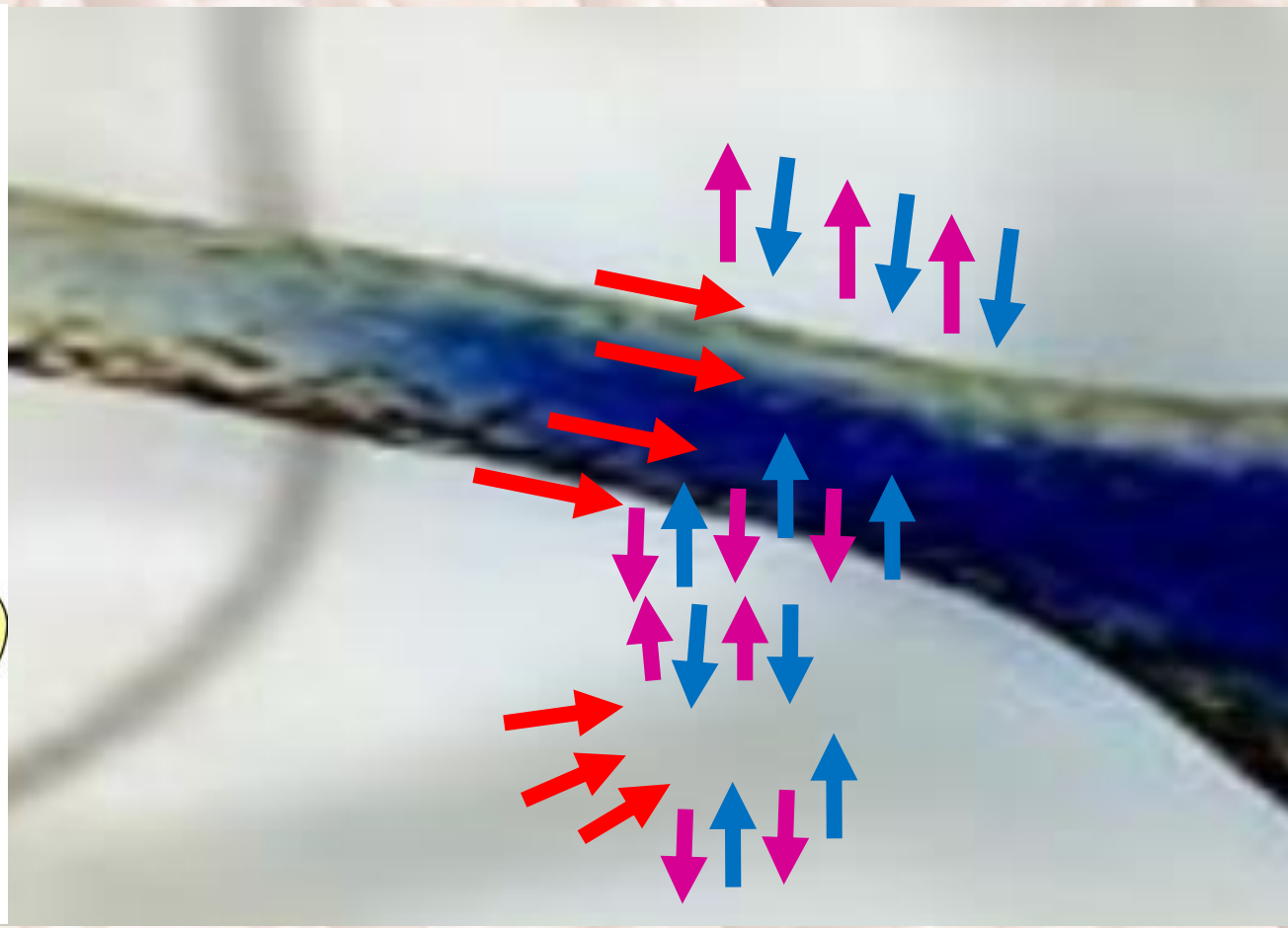




POŁĄCZENIA I SYNAPSY

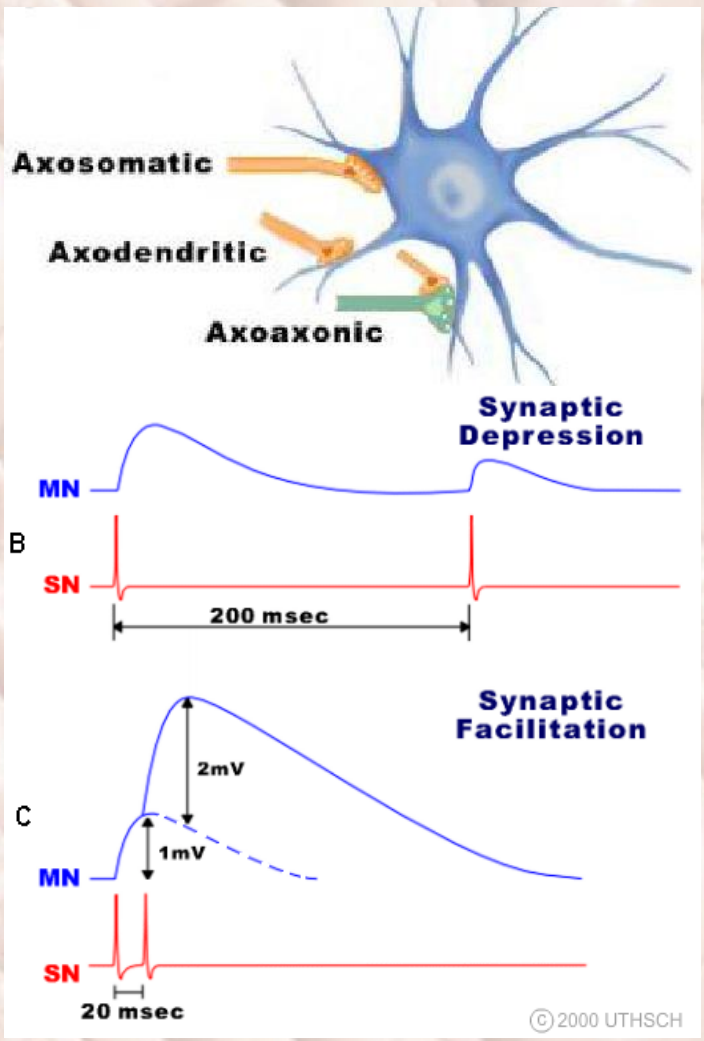
- Szybkie i wybiórcze przekazywanie informacji
- Adaptatywne reagowanie na nadchodzące sygnały
- Zróżnicowanie reprezentowanych kombinacji

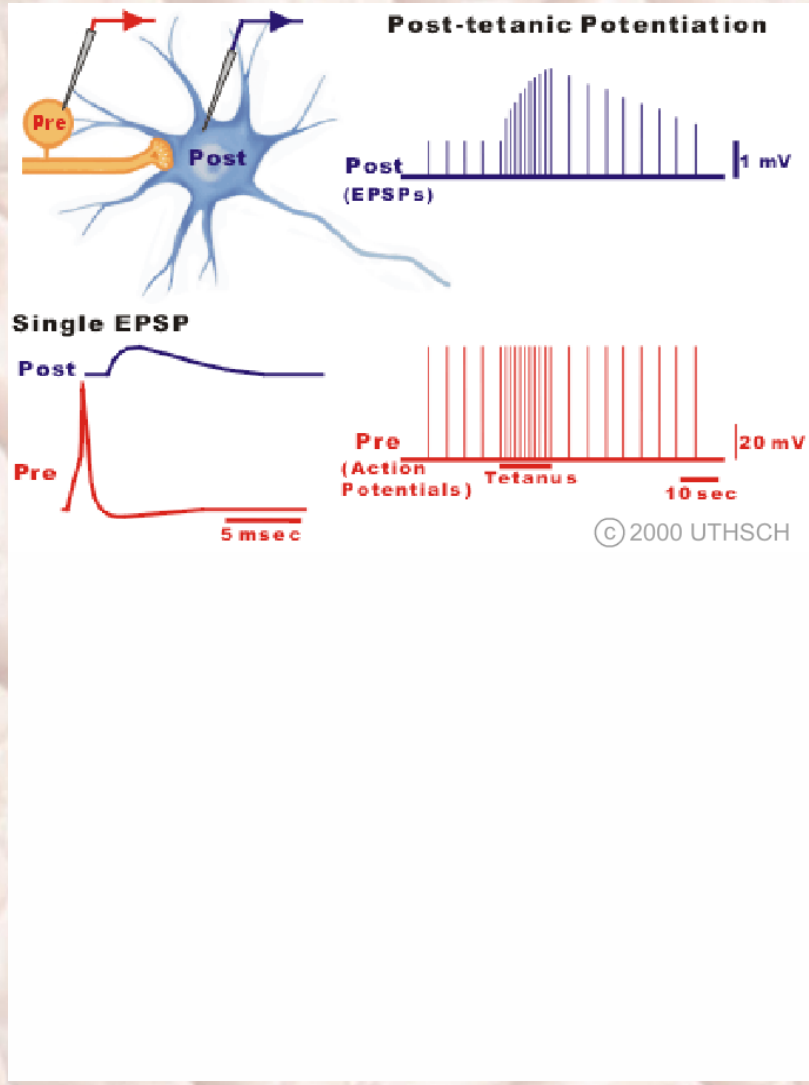




PLASTYCZNOŚĆ SYNAPS

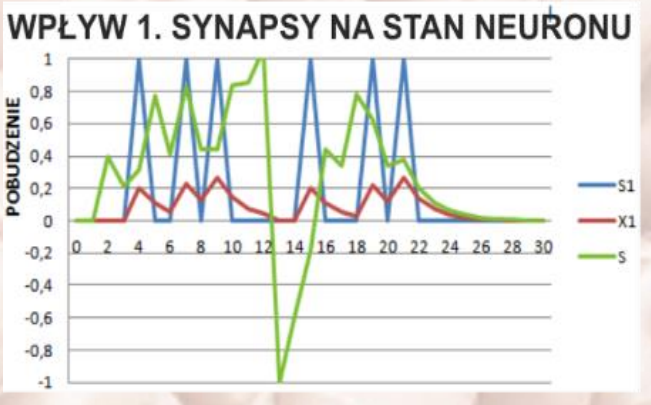
Adaptacyjne mechanizmy plastyczne zależne są od czasu na poziomie milisekund pomiędzy aktywacjami presynaptycznymi i postsynaptycznymi:

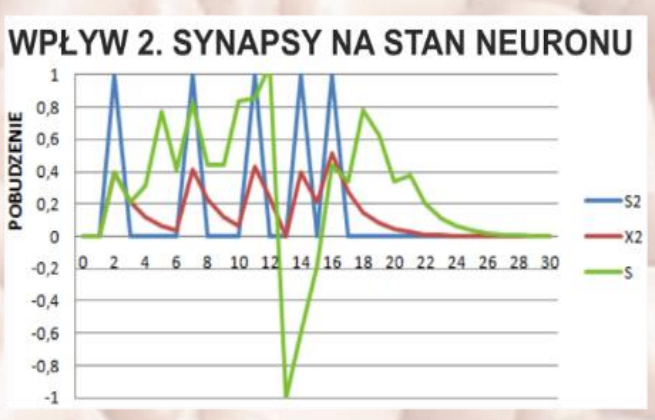


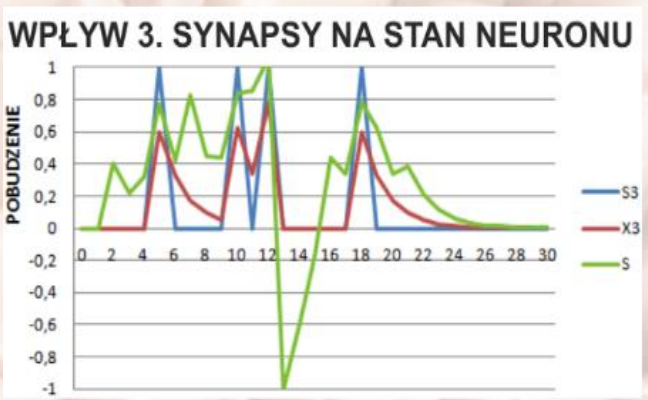


Materiały uzupełniające: http://neuroscience.uth.tmc.edu/s1/chapter07.html

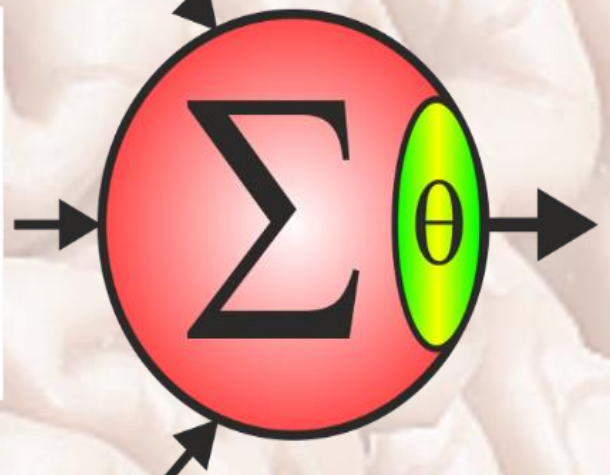
STAN NEURONU







Zmienia się z upływem czasu pod wpływem wewnętrznych procesów, a nie tylko na skutek zewnętrznych oddziaływań.



Czas wespół z innymi czynnikami decyduje o stanie neuronów oraz ich zwiększonej lub zmniejszonej reaktywności.

SZTUCZNE SIECI NEURONOWE

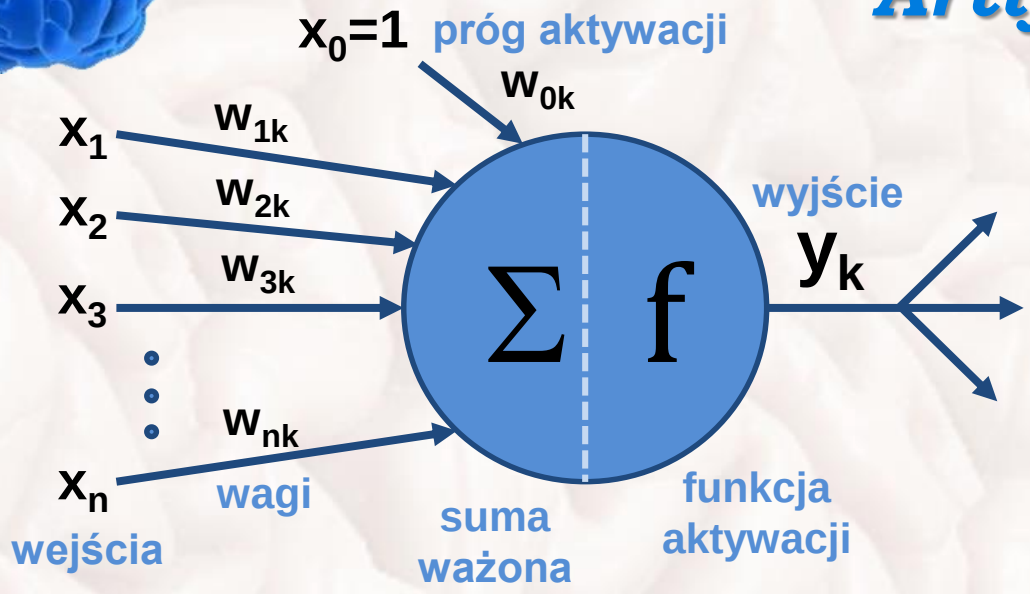
Artificial Neural Networks - ANN

Sztuczne sieci neuronowe – to nazwa przypisywana matematycznym modelom naturalnych sieci neuronowych opierających się na idei neuronu McCullocha-Pittsa oraz jego wariacjom bazującym na różnych:

- **funkcjach aktywacji sztucznych neuronów,**
- **architekturach połączeń sztucznych neuronów,**
- **metodach adaptacji/uczenia.**

W sztucznych sieciach neuronowych traktuje się sztuczny neuron jako jednostkę obliczającą na swoim wyjściu pewną wartość (niczym funkcja) na podstawie zwykle ważonych danych wejściowych, z których obliczana jest najczęściej suma ważona wejść, na podstawie której wyznaczają jest wartość tzw. funkcji aktywacji neuronów.

SZTUCZNY NEURON
Artificial Neuron



$$y_k = f\left(\sum_{i=0}^{n} w_{ik}x_i\right)$$

- Dane z wszystkich wejść x_1 … x_n równocześnie oddziałują na sztuczny neuron.
- Poprzednie stany sztucznego neuronu nie mają żadnego wpływu na jego aktualny stan, liczy się tylko aktualne pobudzenie oraz wagi w_{0k}, w_{1k} … w_{nk}.
- Nie istnieją żadne zależności czasowe pomiędzy jego stanami.
- Reakcja sztucznego neuronu następuje natychmiast i powoduje obliczenie wartości wyjściowej ewaluując wybraną funkcję aktywacji sztucznego neuronu f, której wartość zależna jest od sumy ważonej wejść oraz aktualnych wartości wag w_{0k}, w_{1k} … w_{nk}.

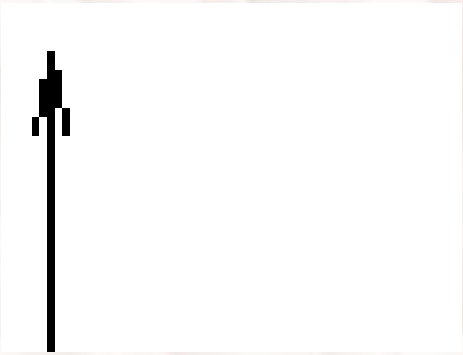
FUNKCJE AKTYWACJI SZTUCZNYCH NEURONÓW

Funkcje aktywacji sztucznych neuronów – to zwykle funkcje progowe/schodkowe, liniowe lub nieliniowe.

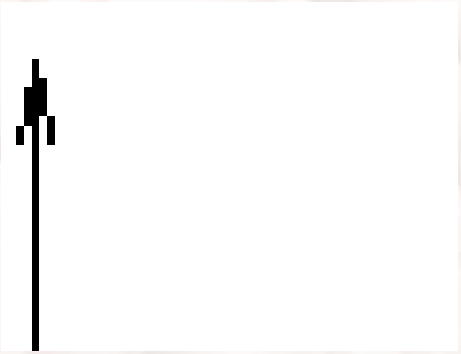
Wśród funkcji nieliniowych najczęściej występują funkcje sigmoidalne (ew. tangens hiperboliczny) oraz radialne (np. funkcja Gaussa).

Funkcje ciągłe umożliwiają aproksymację oraz wykorzystanie metod gradientowych, chętnie stosowanych do adaptacji sztucznych sieci neuronowych.

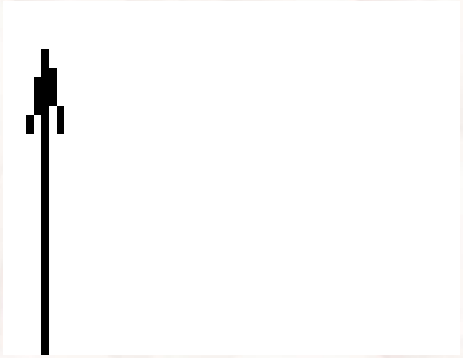
Funkcje progowe/schodkowe f
z wyjściem binarnym {0;1} lub {-1;+1}



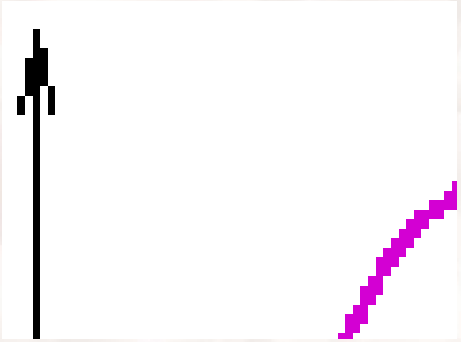
Funkcje liniowe f
z wyjściem ciągłym



Funkcje sigmoidalne f
z wyjściem ciągłym



Funkcje Gaussowskie f
z wyjściem ciągłym



PERCEPTRON

model neuronu o progowej funkcji aktywacji

Perceptron zdefiniowany jest jako pewna funkcja schodowa:

$$y = f\left(\sum_{k=1}^{n} w_k x_k + \theta\right)$$

θ - próg aktywacji neuronu

Funkcja schodkowa unipolarna:

$$f(s) = \begin{cases} 1 & gdy\ s > \theta \\ 0 & gdy\ s \leq \theta \end{cases}$$

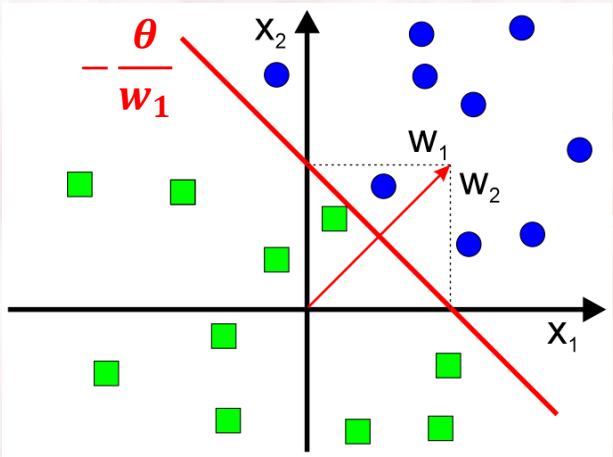
Funkcja schodkowa bipolarna:

$$f(s) = \begin{cases} +1 & gdy\ s > \theta \\ -1 & gdy\ s \leq \theta \end{cases}$$

Perceptron posiadający n wejść dzieli n-wymiarową przestrzeń wektorów wejściowych x na dwie półprzestrzenie, które są podzielone (n-1)-wymiarową hiperpłaszczyzną, zwaną granicą decyzyjną określoną wzorem:

$$\sum_{k=1}^{n} w_k x_k + \theta = 0$$

Prosta wyznaczająca podział przestrzeni jest zawsze prostopadła do wektora wag:



ALGORYTM UCZENIA PERCEPTRONU

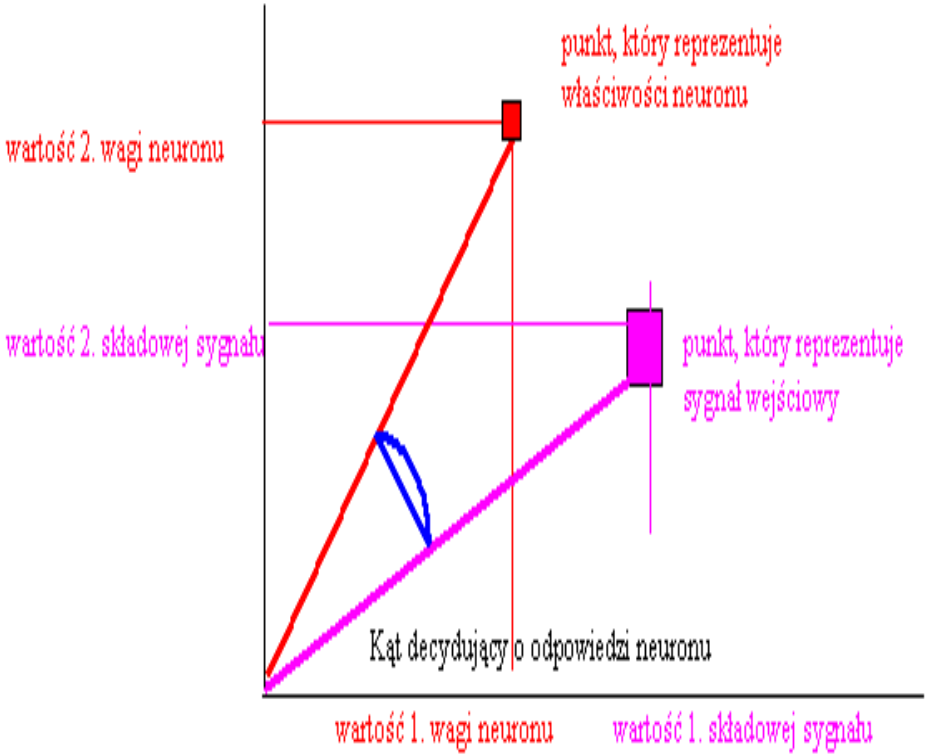
Dla określonego zbioru uczącego składającego się z par wektorów uczących (x, d(x)), gdzie d(x) to uczona wartość wyjściowa przez nauczyciela (w uczeniu nadzorowanym) wykonaj następujące kroki:

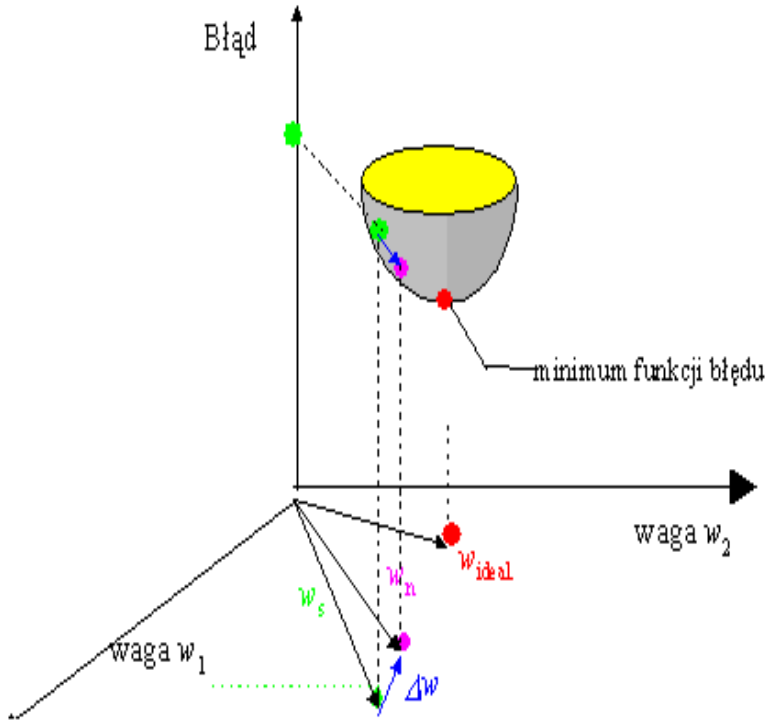
1. W sposób losowy wybierz wagi początkowe perceptronu w_i.
2. Na wejścia perceptronu podaj kolejny wektor uczący x.
3. Oblicz wartość wyjściową perceptronu y(x).
4. Porównaj uzyskaną wartość wyjściową y(x) z wartością wzorcową d dla wektora x.
5. Dokonaj modyfikacji wag według zależności:

   Jeżeli $y(x) \neq d(x)$, to $\theta = \theta + d(x)$ oraz $w_i = w_i + d(x) \cdot x_i$
   w przeciwnym przypadku waga się nie zmienia.
6. Oblicz średni błąd dla wszystkich wzorców uczących.
7. Jeśli błąd jest mniejszy od założonego lub osiągnięto maksymalną ilość powtórzeń zbioru uczącego przerwij algorytm.
8. W odwrotnym przypadku przejdź do punktu 2.

SPOSÓB DZIAŁANIA PERCEPTRONU

Działanie sieci neuronowej jest wypadkową działania poszczególnych neuronów oraz zachodzących pomiędzy nimi interakcji. Pojedynczy neuron w typowych przypadkach realizuje (z matematycznego punktu widzenia) operację iloczynu skalarnego wektora sygnałów wejściowych oraz wektora wag. W efekcie odpowiedź neuronu zależy od wzajemnych stosunków geometrycznych pomiędzy wektorami sygnałów i wektorami wag.





PIERWSZE MODELE SZTUCZNYCH NEURONÓW

Model McCullocha-Pittsa (1943 r.) odzwierciedla tylko proste sumowanie statycznie ważonych sygnałów wejściowych $x_1,\ldots,x_n$, jakie do niego docierają, próg aktywacji w_0 oraz pewną funkcję aktywacji f zależną od wartości sumy ważonej sygnałów wejściowych i wartości progu:

$$x = w_0 + \sum_{i=1}^{n} x_i w_i \qquad y = f(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Mimo dużej prostoty, model ten znalazł ogromną ilość zastosowań w bardzo wielu współcześnie stosowanych metodach inteligencji obliczeniowej, pokazując swoją uniwersalność i możliwość jego zastosowania w różnych zadaniach aproksymacji, regresji, klasyfikacji, pamięciach skojarzeniowych i wielu innych.

Pierwsza generacja neuronów McCullocha-Pittsa zbudowana była z wykorzystaniem dyskretnych funkcji aktywacji, które zwracają wartości unipolarne {0; 1} lub bipolarne {-1; 1}.

FUNKCJE CIĄGŁE AKTYWACJI SZTUCZNYCH NEURONÓW

Druga generacja neuronów wywodzących się modelu McCullocha-Pittsa stosuje ciągłe funkcje aktywacji z zakresu [0;1] lub [-1;1], wykorzystując najczęściej sigmoidalne lub radialne funkcje aktywacji. Istnieją **neurony sigmoidalne**, neurony oparte o funkcję tangens hiperboliczny lub **neuronami radialnymi**, np. **neuron Gaussowski**, **neuron Hardy'ego**, **neuron wielomianowy**. Są one stosowane obecnie najpowszechniej w różnego rodzaju modelach sztucznych sieci neuronowych:

$$y = f(x) = \frac{1}{\sqrt{\|x-c\|^2 + \delta^2}}$$

$$y = f(x) = \frac{1}{1+e^{-\beta x}}$$

$$y = f(x) = \tanh(\beta x)$$

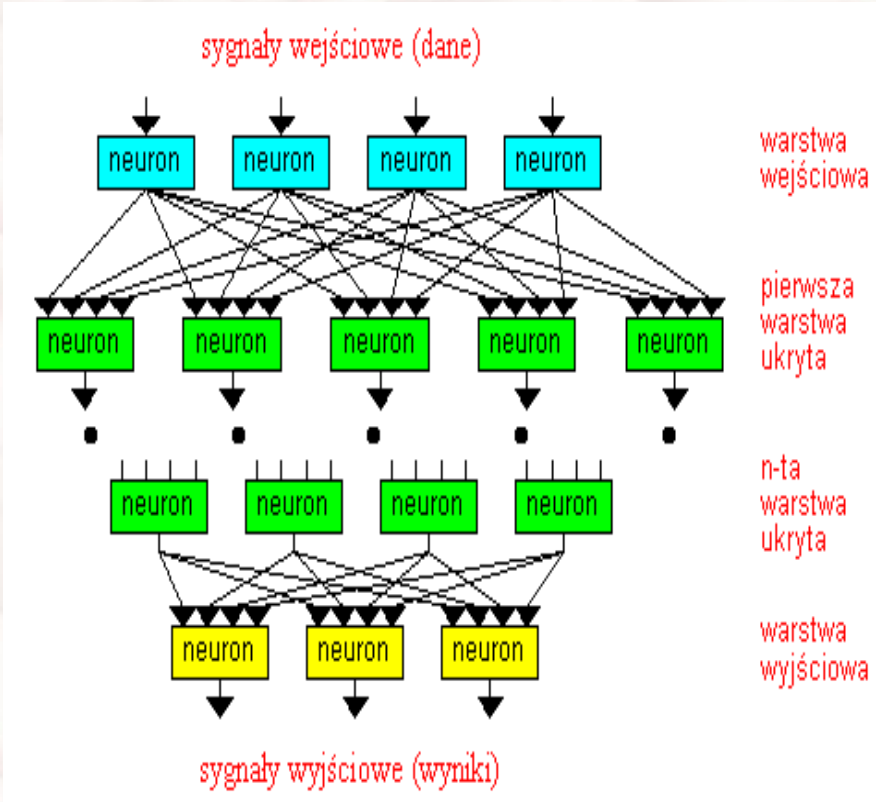
$$y = f(x) = \|x-c\|^2 + \delta^2$$

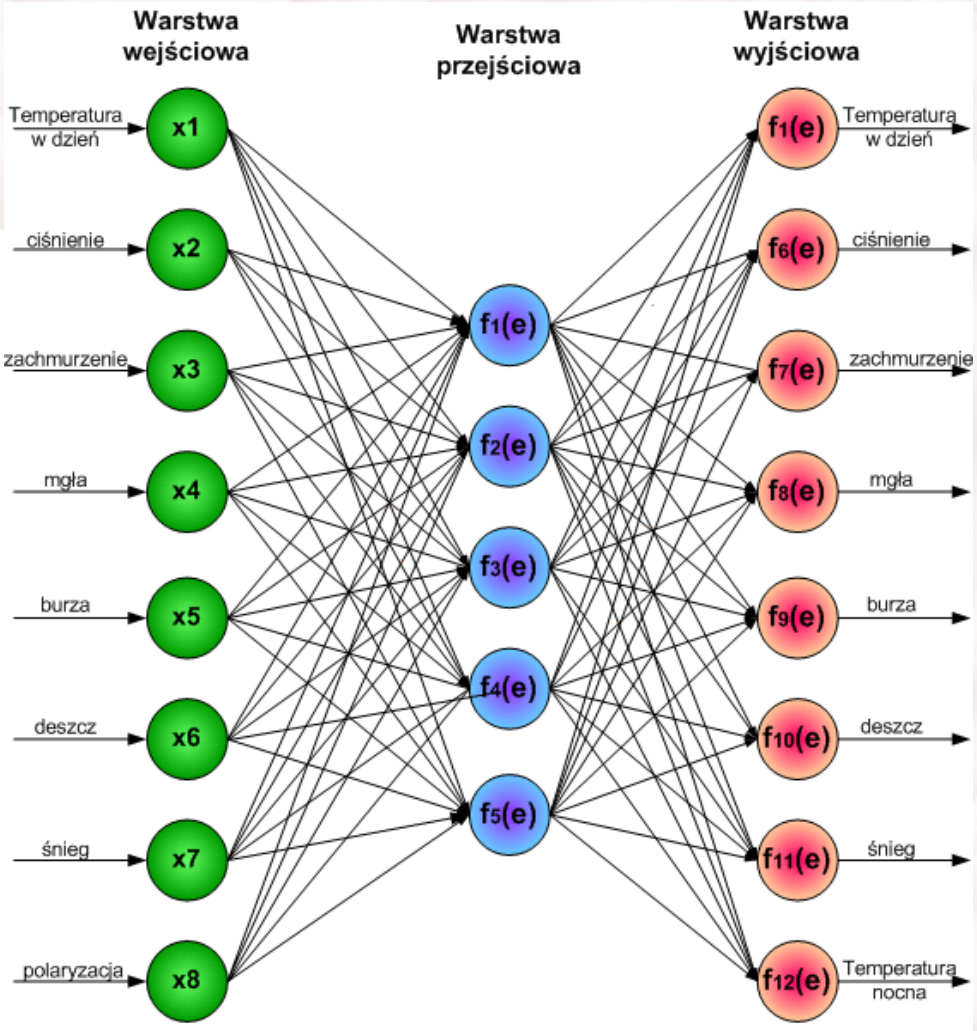
$$y = f(x) = e^{\frac{\|x-c\|^2}{-2\sigma^2}}$$

gdzie β, σ, δ i c są parametrami, z których niektóre mogą być adaptowane w trakcie procesu nauki sieci, wykorzystującej takie neurony.

TOPOLOGIA SZTUCZNYCH SIECI NEURONOWYCH

Sztuczne sieci neuronowe są bardzo uproszczonymi i zmodyfikowanymi modelami biologicznych sieci neuronowych o topologii warstwowej spotykanej w korze mózgowej istot żywych.





UCZENIE I ADAPTACJA SIECI NEURONOWYCH

Do najbardziej znanych i powszechnie wykorzystywanych metod uczenia sieci neuronowych należą:

Reguła Hebba polegająca na wzmacnianiu tych połączeń synaptycznych (A~B), w których aktywność jednego neuronu A powoduje aktywność drugiego połączonego z nim neuronu B. Odpowiada to empirycznym badaniom nad LTP (long term potentiation).

Metoda wstecznej propagacji błędów (back propagation) umożliwiająca uczenie sieci wielowarstwowych poprzez propagację różnicy pomiędzy pożądanym a otrzymanym sygnałem na wyjściu sieci.

Samoorganizacja w sieci neuronowej (np. SOM) umożliwiają uczenie sieci bez nauczyciela (unsupervised), którego celem jest wykształcenie w sieci neuronów, które by reagowały na pewne powtarzające się kombinacje bodźców z przestrzeni danych, którym można później przypisać pewne znaczenie. W tych sieciach wzmacniany jest neuron, który osiąga największą wartość pobudzenia oraz ew. również otaczające go neurony, co zapewnia sytuowanie podobnych kombinacji bodźców blisko siebie.

Sieci rekurencyjne (np. sieci Hoppfielda), których bodźce krążą przez pewien określony czas aż do osiągnięcia pewnego stanu stabilności, który jest traktowany jako odpowiedź sieci.

PODSTAWY UCZENIA SIECI NEURONOWYCH

Sztuczne sieci neuronowe można adaptować do rozwiązywania różnych zadań na wiele różnych sposobów.

Istnieje wiele reguł ich adaptacji, spośród których większość wykorzystuje uczenie się oparte o tzw. wzorce uczące.

W sztucznych sieciach neuronowych w trakcie uczenia adaptacji podlegają wagi reprezentujące siłę połączeń pomiędzy neuronami.

Reguła Widrowa-Hoffa

Dotyczy uczenia nadzorowanego sieci jednokierunkowych, gdzie minimalizuje się błąd pomiędzy pożądaną a aktualną odpowiedzią.

$$\delta^j = z^j - y^j = z^j - \mathbf{w}^T \mathbf{x}^j$$

Korekta wag jest następująca (Widrow,Hoff 1962):

$$\Delta w_i = \eta \cdot \delta^j \cdot x_i^j$$

MODEL ADALINE

Adaptive Linear Neuron

Neuron Adaline jest bardzo podobny do Perceptronu. Różni się algorytm uczenia:

Porównuje się sygnał wzorcowy *d* z sygnałem *s* na wyjściu liniowej części neuronu (sumatora), a więc błąd opisany jest wzorem: $\delta = d - s$

Uczenie neuronu sprowadza się do minimalizacji funkcji błędu średniego kwadratowego, nazywany też **błędem średniokwadratowym**:

$$Q(w) = \frac{1}{2}\,\varepsilon^2 = \frac{1}{2}\left[d - \left(\sum_{i=0}^{n} w_i x_i\right)\right]^2$$

Ze względu na to, iż funkcja jest różniczkowalna,
możemy użyć metody największego spadku gradientu:

$$w_i(t+1) = w_i(t) - \eta\frac{\partial Q(w_i)}{\partial w_i} = w_i(t) + \eta\delta x_i = w_i(t) + \eta(d-s)x_i$$

Reguła ta nosi nazwę **reguły delta**: $\delta = d - s$

REGUŁA DELTA UCZENIA SIECI NEURONOWYCH

Jedną z najbardziej znanych i często stosowanych reguł adaptacji sztucznych sieci neuronowych w procesie uczenia jest reguła delta.

Reguła delta

Obowiązuje dla neuronów z ciągłymi funkcjami aktywacji i nadzorowanego trybu uczenia. Regułę delta wyprowadza się jako wynik minimalizacji kryterium błędu średniokwadratowego Q.

$$Q = \frac{1}{2}\sum_{j=1}^{N}\left(z^j - y^j\right)^2 = \sum_{j=1}^{N} Q^j, \qquad Q^j = \frac{1}{2}\left(\delta^j\right)^2$$

Korekta wag:

$$\Delta w_i = \eta \cdot \delta^j \cdot f'(e^j) \cdot x_i^j$$

gdzie f'() oznacza pochodną funkcji aktywacji. W przypadku funkcji sigmoidalnej:

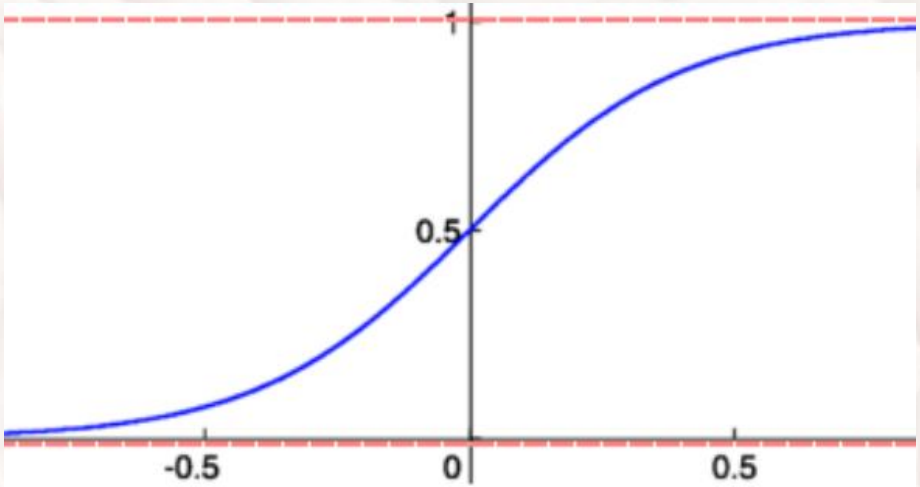
$$\Delta w_i = \eta \cdot \delta^j \cdot (1 - y^j) \cdot y^j \cdot x_i^j$$

Stosowana jest do uczenia wielowarstwowych sieci neuronowych wraz z algorytmem wstecznej propagacji błędów (Rumelhart, McClelland 1986)

SIGMOIDALNY MODEL NEURONU

Model ten niewiele różni się od Perceptronu i modelu Adaline. Różnica polega na zastosowaniu funkcji sigmoidalnej na wyjściu neuronu, która oblicza sygnał wyjściowy na podstawie sumy ważonej sygnałów wejściowych:

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



Miarę błędu definiuje się wtedy jako:

$$Q(w) = \frac{1}{2}\left[d - f\left(\sum_{i=0}^{n} w_i x_i\right)\right]^2$$

Korzystając z metody największego spadku gradientu otrzymujemy:

$$w_i(t+1) = w_i(t) - \eta\frac{\partial Q(w_i)}{\partial w_i} = w_i(t) - \eta\delta x_i$$

$$= w_i(t) - \eta\big(d - f(s)\big)f'(s)x_i$$

SIECI NEURONOWE WIELOWARSTWOWE

MLP – Multilayer Perceptron

Sieci wielowarstwowe to odpowiednio połączone warstwy neuronów zwykle o nieliniowych funkcjach aktywacji (np. neurony sigmoidalne, radialne), aczkolwiek czasami w warstwie wyjściowej pojawiają się neurony liniowe.

Sieci wielowarstwowe muszą posiadać minimalnie dwie warstwy neuronów:

warstwę wejściową i **warstwę wyjściową**

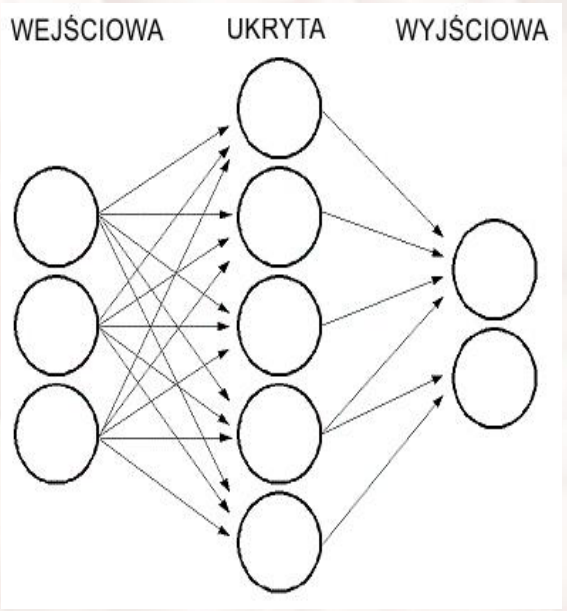
pomiędzy którymi może być jedna lub więcej **warstw ukrytych**.

W każdej warstwie może występować różna ilość neuronów:

W warstwie wejściowej odpowiada ona zwykle ilości parametrów opisujących dane wejściowe, w warstwie wyjściowej np. ilości klas, natomiast ilość neuronów w warstwach ukrytych odpowiada za możliwości modelu.

Neurony łączą się tylko pomiędzy (zwykle) sąsiednimi warstwami, zaś wewnątrz warstw nie występują połączenia pomiędzy neuronami.

Neurony zwykle łączymy pomiędzy sąsiednimi warstwami na zasadzie każdy z każdym.



UCZENIE SIECI WIELOWARSTWOWYCH

Sieci wielowarstwowe uczymy zwykle metodami nadzorowanymi (tzn. z nauczycielem):

1. **Podajemy na wejście sygnał wejściowy (zwykle w postaci wektora lub macierzy danych)**
2. **Obliczamy wartości wyjściowe neuronów w 1. warstwie i poprzez ich połączenia z neuronami w 2. warstwie podajemy te wartości jako sygnały wejściowe dla neuronów w 2. warstwie.**
3. **Obliczamy wartości wyjściowe w kolejnych warstwach tym samym sposobem.**
4. **Wartości wyznaczone w ostatniej warstwie stanowią zarazem odpowiedź sieci na podany sygnał wejściowy.**
5. **Sygnał ten porównujemy z wzorcowym (określonym przez nauczyciela) i wyznaczamy błąd.**
6. **Korzystając metod gradientowych propagujemy błąd wstecz przez sieć i dokonujemy korekty wartości wag połączeń synaptycznych.**

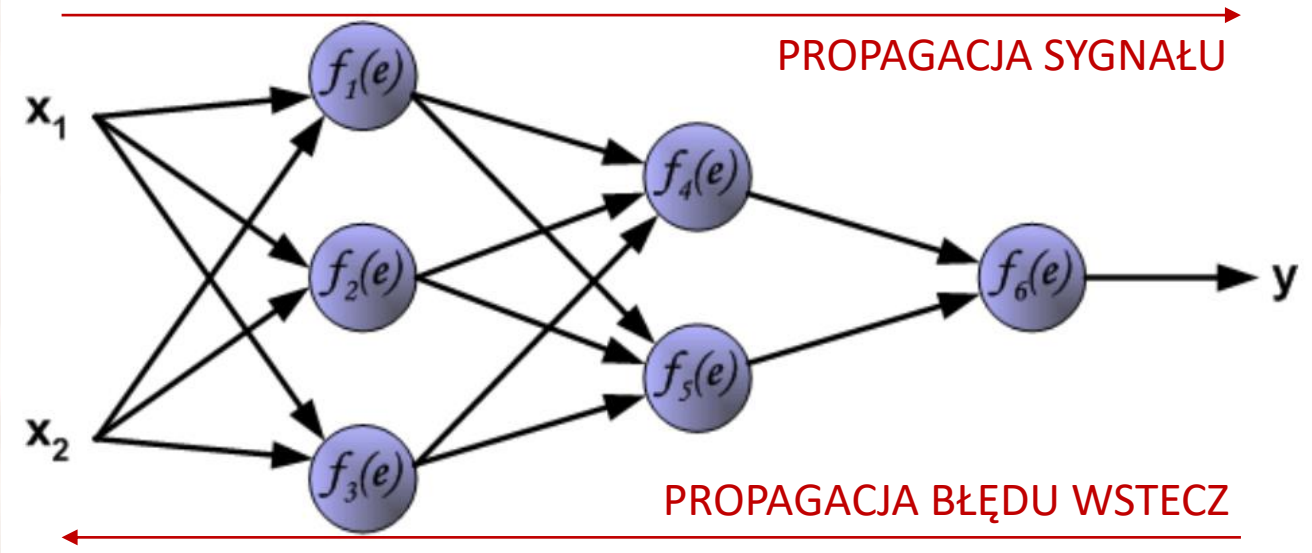
ALGORYTM WSTECZNEJ PROPAGACJI BŁĘDÓW

Pierwszym i zarazem najbardziej popularnym algorytmem do uczenia sieci wielowarstwowych jest algorytm wstecznej propagacji błędów (*backpropation algorithm*), którego działanie oparte jest na regule delta.

Wyznaczamy średniokwadratową funkcję błędu dla sieci Q(w), a następnie dążymy do znalezienia minimum tej funkcji względem wektora w.

Uczenie składa się z dwóch naprzemiennych faz:

1. **Fazy propagacji sygnału od wejść (wektora x) do wyjścia.**
2. **Fazy wstecznej propagacji błędu od wyjścia y w kierunku wejść sieci.**

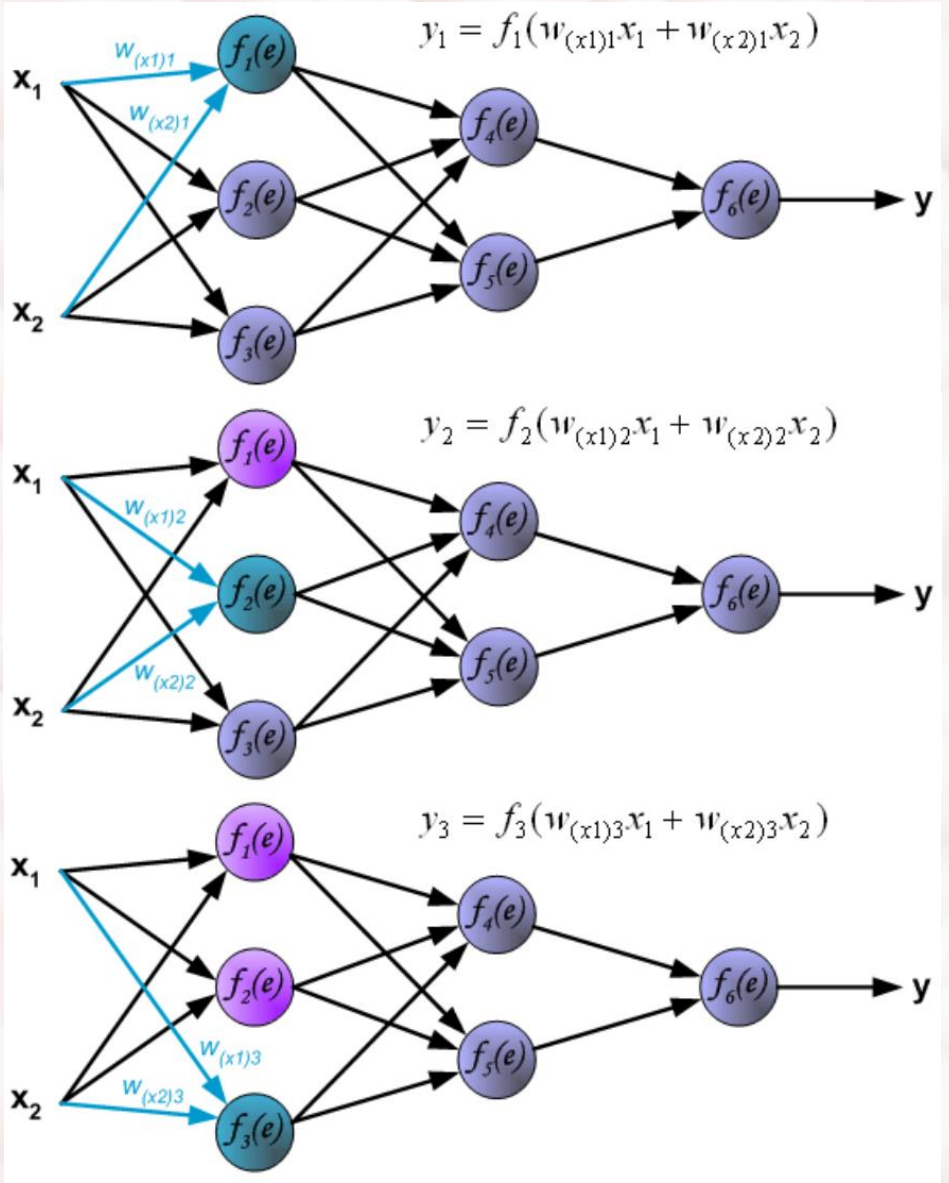


PROPAGACJA SYGNAŁU PRZEZ PIERWSZĄ WARSTWĘ SIECI

Sieć wielowarstwową pobudzamy sygnałami wejściowymi (x_1, x_2) kolejno warstwami neuronów.

Najpierw pobudzane są neurony w 1. warstwie i obliczana jest ich wartość wejściowa y_1, y_2 i y_3.

Wartości wyjściowe następnie są wykorzystane jako sygnały wejściowe w kolejnej warstwie neuronów.

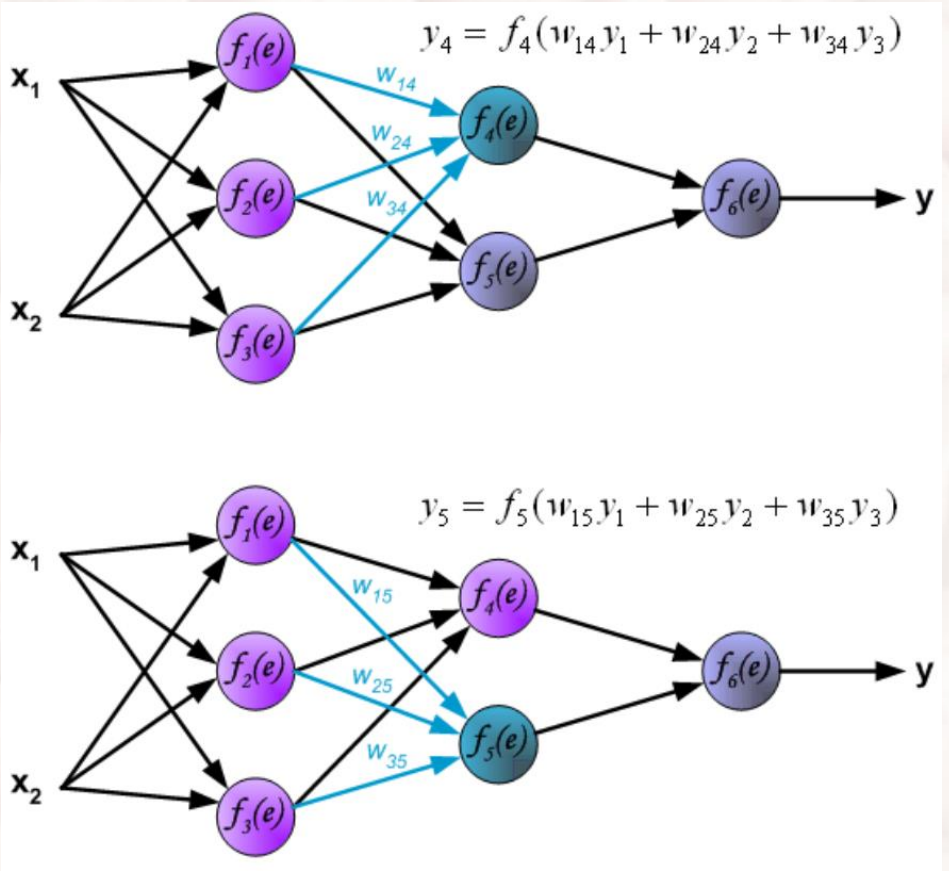


PROPAGACJA SYGNAŁU PRZEZ DRUGĄ WARSTWĘ SIECI

Drugą (ukrytą) warstwę neuronów sieci wielowarstwowej pobudzamy sygnałami wyjściowymi warstwy pierwszej (y_1, y_2 i y_3) obliczonymi w poprzednim kroku.

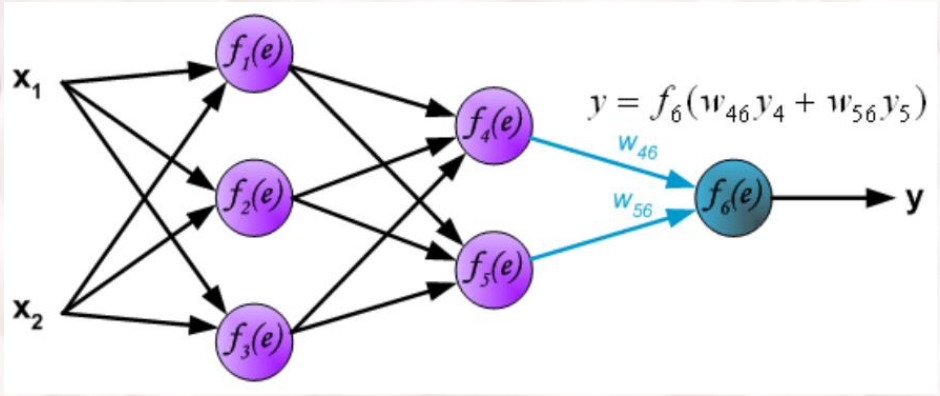
Na tej podstawie wyznaczane są wartości wyjściowe neuronów w warstwie drugiej (y_4 i y_5).

Obliczone wartości wyjściowe następnie są wykorzystane jako sygnały wejściowe w ostatniej warstwie neuronów, tzw. Warstwie wyjściowej sieci, która w naszym przypadku zawiera tylko 1 neuron.



PROPAGACJA SYGNAŁU PRZEZ TRZECIĄ WARSTWĘ SIECI

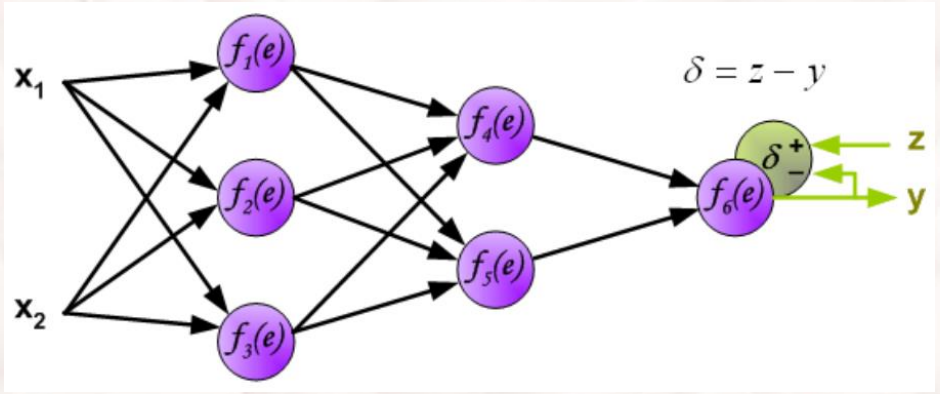
Trzecia warstwa neuronów sieci wielowarstwowej zawierająca pojedynczy neuron pobudzamy sygnałami wyjściowymi warstwy drugiej (y_4 i y_5) obliczonymi w poprzednim kroku.



Na tej podstawie wyznaczana jest wartość wyjściowa neuronu w warstwie trzeciej (y_6).

Obliczona wartość wyjściowa jest porównywana z wartością pożądaną (z) wyznaczoną w zbiorze uczącym przez nauczyciela, a różnica pomiędzy wartością pożądaną i uzyskaną ($\delta = z - y$) stanowią o dalszych krokach działania algorytmu.

Wartość błędu jest propagowana wstecz, ważona zgodnie z wagą połączenia między neuronami i sumowana w tych neuronach celem wyznaczenia ich błędu.



PROPAGACJA WSTECZNA BŁĘDU DO DRUGIEJ WARSTWY

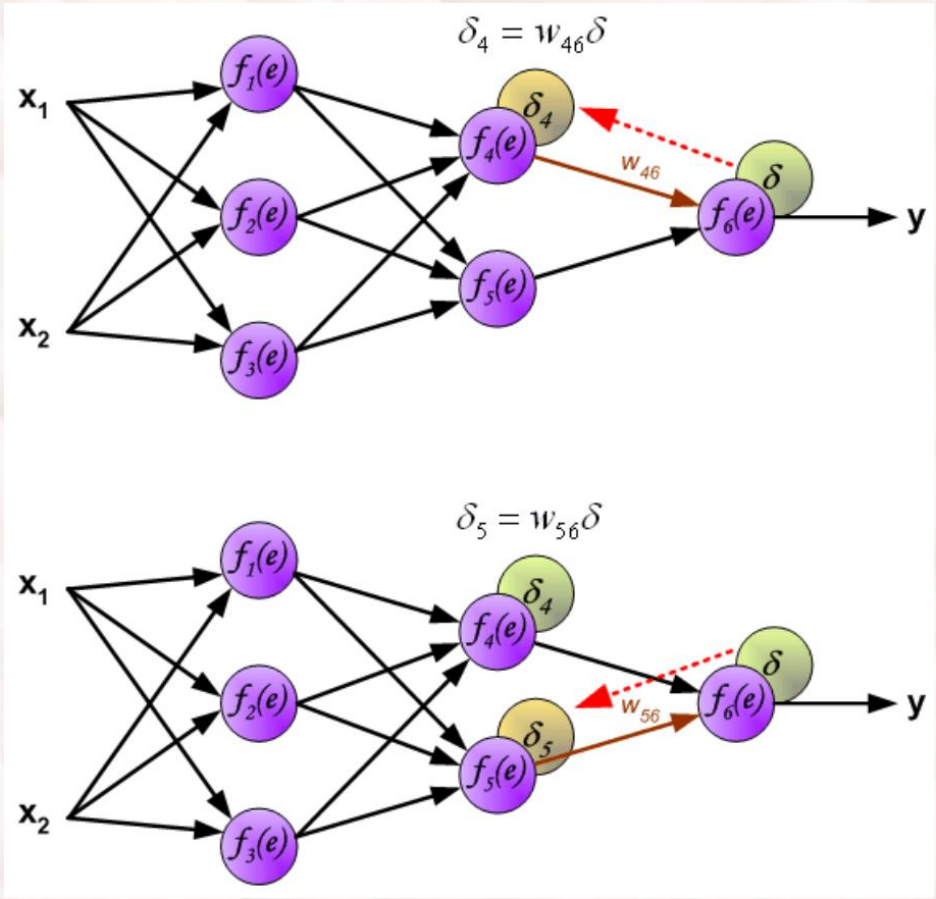
Przechodząc z błędem wstecz z trzeciej warstwy do drugiej, błąd jest ważony zgodnie z aktualną wartością wagi połączenia pomiędzy neuronem warstwy 3 i odpowiednim neuronem warstwy drugiej.

Neurony w warstwie drugiej sumują ważone sygnały błędów dochodzących do nich z warstwy trzeciej. Tutaj ze względu na to, iż w warstwie 3 występuje tylko jeden neuron, sumy składają się tylko z jednego członu:

$$\delta_i = \sum_{j=1}^{k} w_{ij}\delta_j$$

$$\delta_4 = w_{46}\delta_6$$

$$\delta_5 = w_{56}\delta_6$$

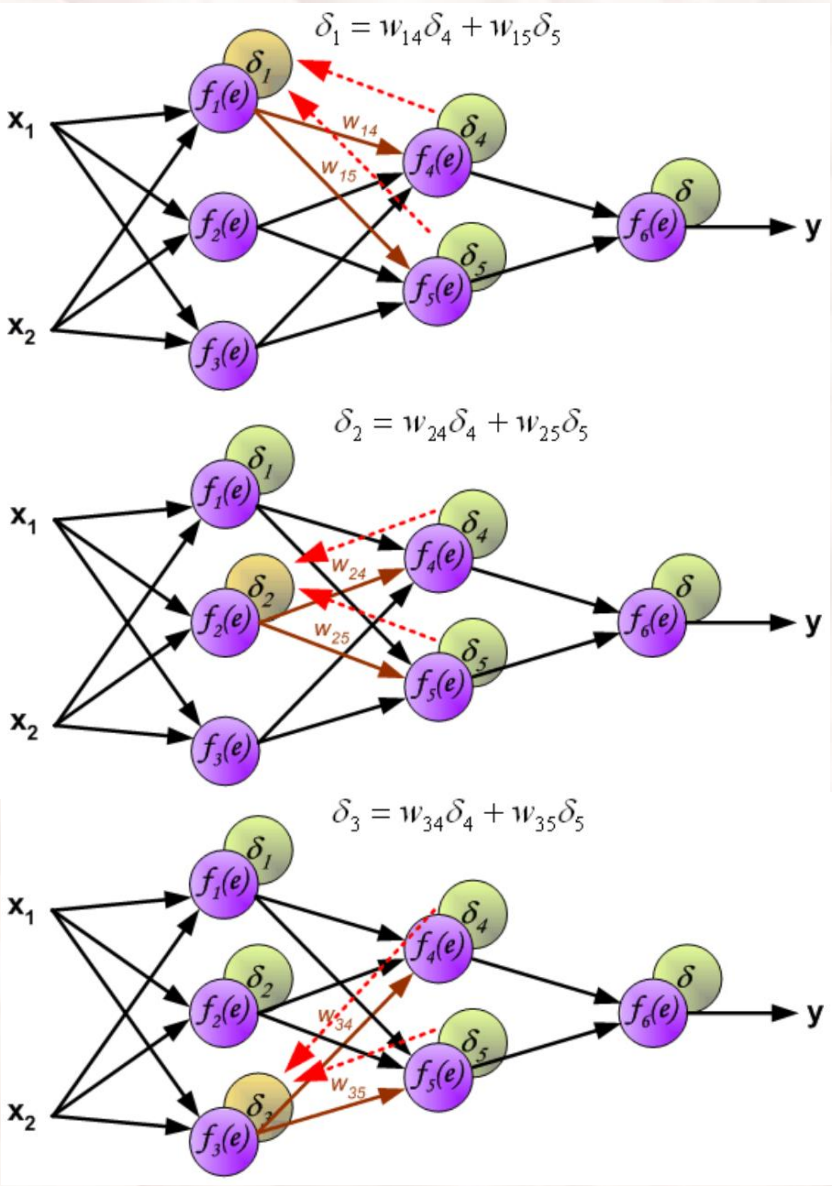


PROPAGACJA WSTECZNA BŁĘDU DO PIERSZWEJ WARSTWY

Przechodząc z błędem wstecz z drugiej do pierwszej warstwy sieci błędy obliczone dla drugiej warstwy (δ_4 i δ_5) są ważone zgodnie z aktualną wartością wag połączeń pomiędzy neuronami warstwy drugiej i pierwszej, a następnie sumowane neuronach warstwy pierwszej zgodnie z następującą zależnością:

$$\delta_i = \sum_{j=1}^{k} w_{ij}\delta_j$$

Następnie dokonywana jest korekta wartości wag sieci.



KOREKTA WAG SIECI W TRAKCIE PROPAGACJI WSTECZNEJ BŁĘDU

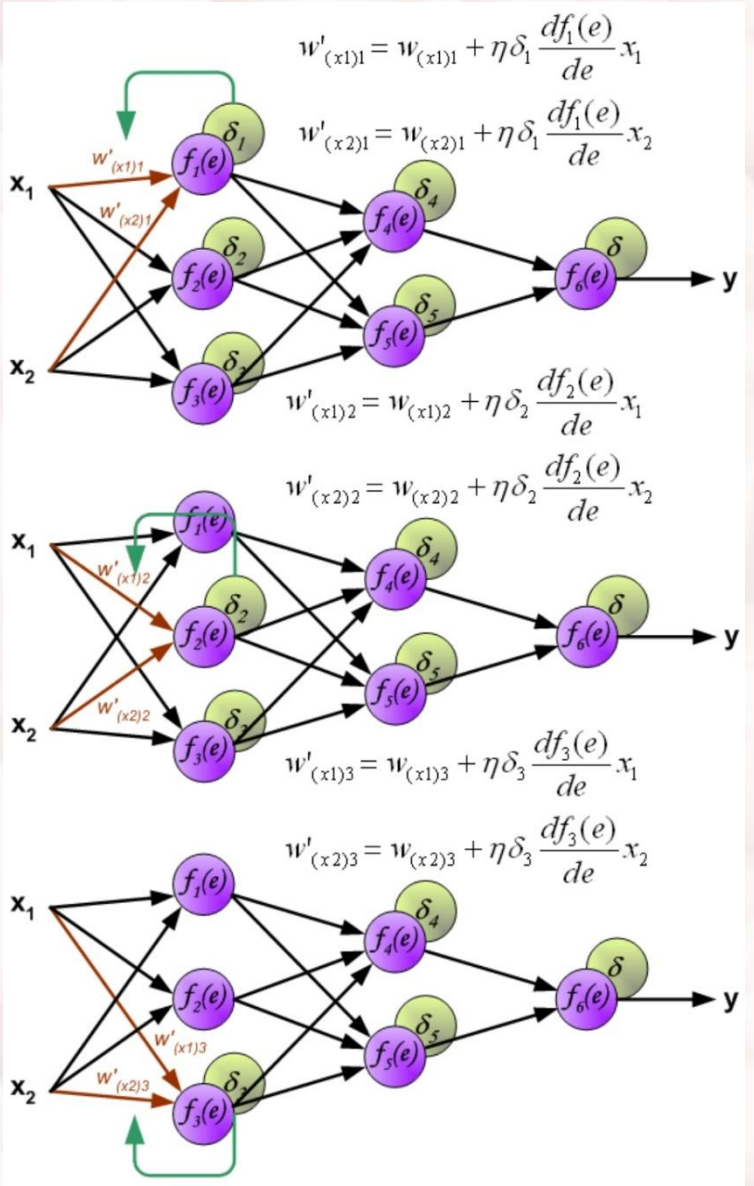
Dokonujemy korekty wag sieci tak, żeby zmniejszyły błąd średniokwadratowy, jaki był obliczony na wyjściu sieci.

W tym celu korzystamy z uogólnionej reguły delta, korzystając z pochodnej cząstkowej funkcji aktywacji oznaczonej jako $df_i(e) / de$.

Korekty wag możemy dokonywać:

od razu dla każdego wzorca uczącego (tzw. *on-line training*)

dopiero po zakończeniu propagacji błędów dla całego zbioru uczącego, sumując je dla wszystkich wzorców, a na końcu obliczając ich średnią (tzw. *off-line training* lub *batch training*).



KOREKTA WAG SIECI W TRAKCIE PROPAGACJI WSTECZNEJ BŁĘDU

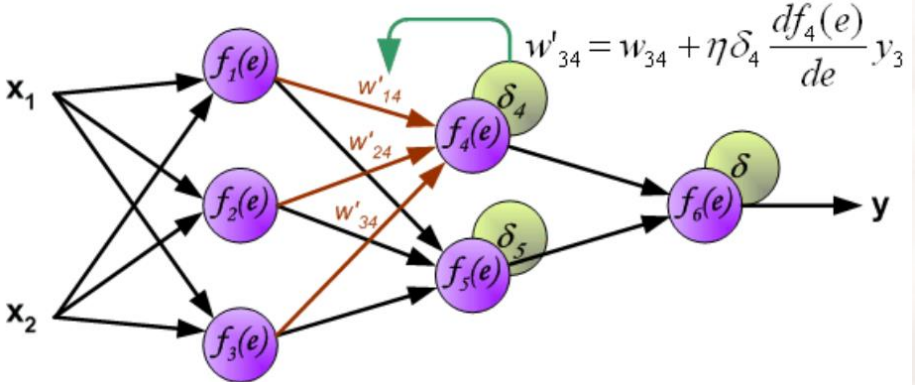
Podobnie dokonujemy korekty wag w drugiej i trzeciej warstwie sieci.

Współczynnik η służy stopniowej adaptacji sieci oraz określa szybkość uczenia się. Na początku jest zwykle duży, a następnie jego wartość jest stopniowo zmniejszana. Warunkuje on możliwość przejścia od minimów lokalnych do minimum globalnego.

$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2$$

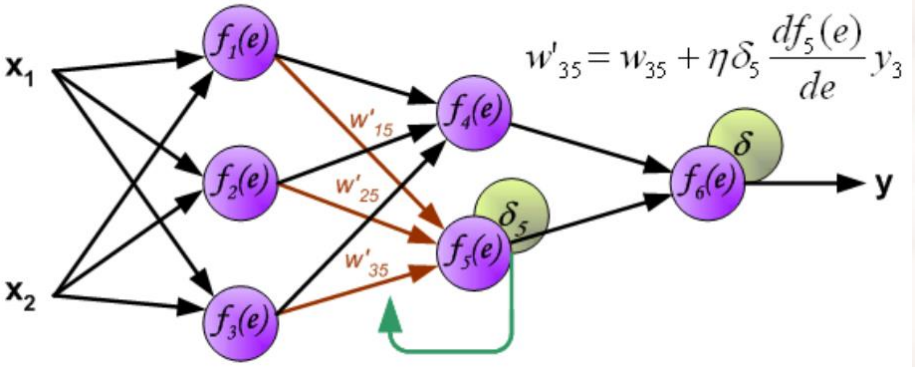
$$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$$



$$w'_{15} = w_{15} + \eta\delta_5 \frac{df_5(e)}{de} y_1$$

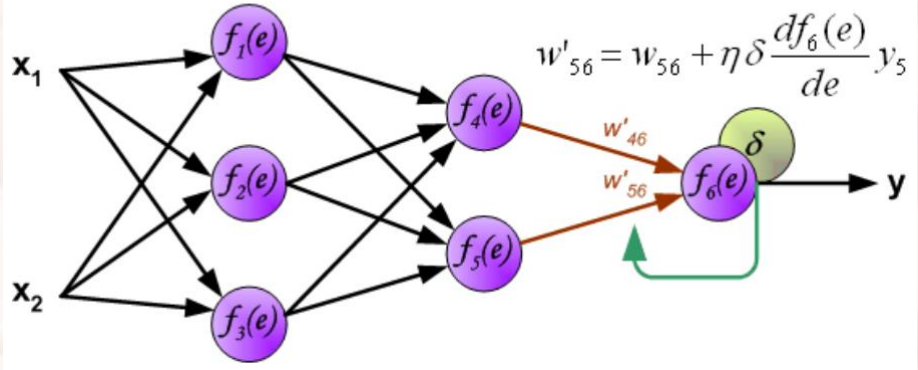
$$w'_{25} = w_{25} + \eta\delta_5 \frac{df_5(e)}{de} y_2$$

$$w'_{35} = w_{35} + \eta\delta_5 \frac{df_5(e)}{de} y_3$$



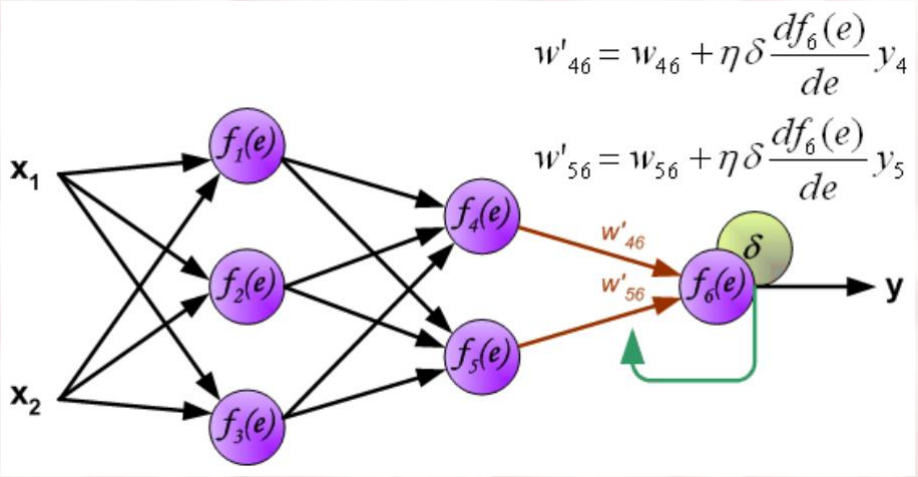
$$w'_{46} = w_{46} + \eta\delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta\delta \frac{df_6(e)}{de} y_5$$



KOREKTA WAG SIECI W TRAKCIE PROPAGACJI WSTECZNEJ BŁĘDU

Wyznaczenie wartości pochodnej we wzorach na aktualizację wag zależne jest od postaci funkcji aktywacji f.



Dla najczęściej stosowanych funkcji:

sigmoidalnej:

$$f(x) = \frac{1}{1 + e^{-\beta * x}}$$

$$f'(x) = \beta * f(x) * \left(1 - f(x)\right) = \beta * y * (1 - y)$$

tangensa hiperbolicznego:

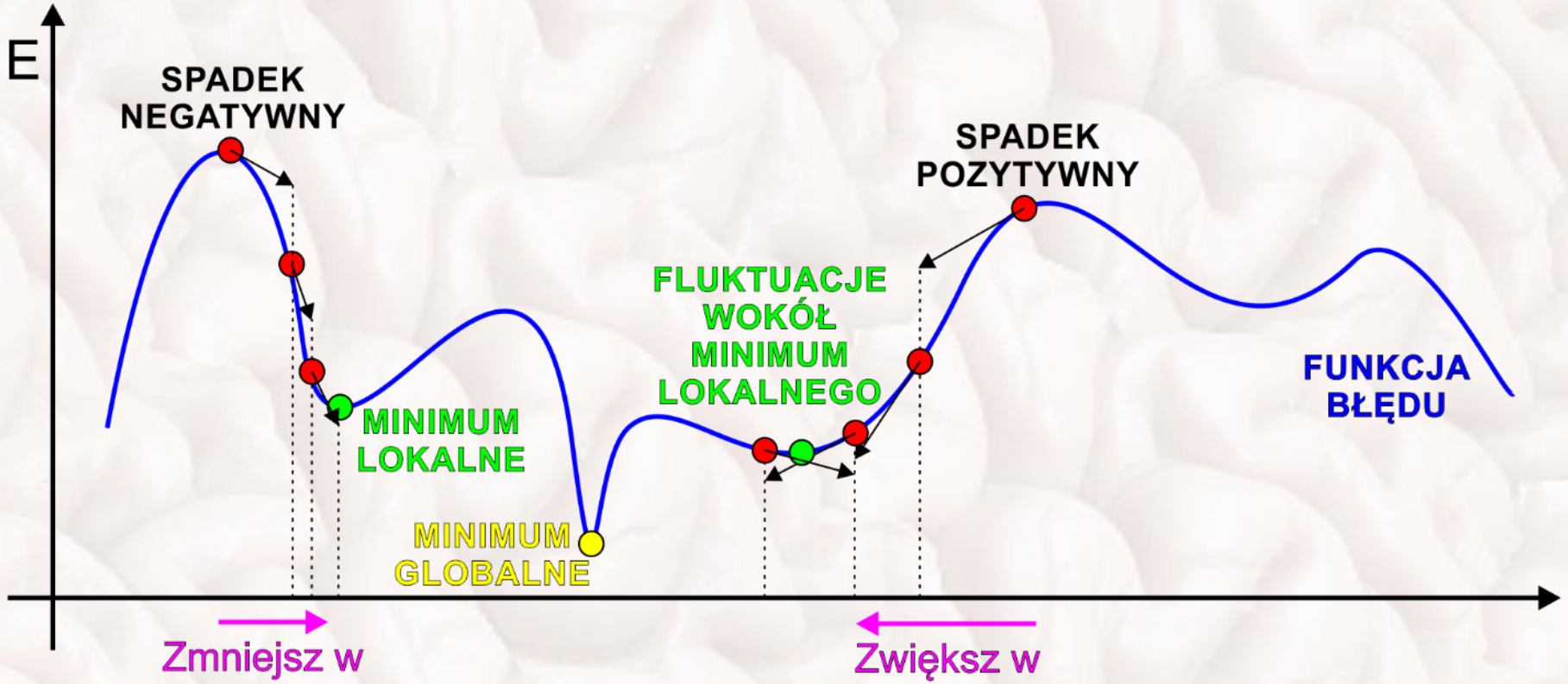
$$f(x) = tgh(\beta * x)$$

$$f'(x) = \beta * \left(1 - tgh^2(\beta * x)\right) = \beta * y\left(1 - y^2\right)$$

TRUDNOŚCI ZWIĄZANE Z ADAPTACJĄ SIECI NEURONÓW

Najczęściej spotykane metody adaptacji sieci neuronowych wykorzystują metody optymalizacji gradientowej, wyznaczając kierunek spadku gradientu funkcji błędu.

Metody te jednak są narażone na utknięcie w lokalnych minimach, co jest podstawową piętą Achillesa tych metod, nie dając gwarancji znalezienia globalnego minimum – a więc optymalnego rozwiązania z punktu widzenia określonej architektury sieci neuronowej oraz przyjętych funkcji aktywacji neuronów.



ALGORYTM Levenberga-Marquardta

Algorytm Levenberga-Marquardta jest obecnie jednym z najczęściej stosowanych do uczenia jednokierunkowych sieci neuronowych ze względu na jego szybką zbieżność, niewielką złożoność obliczeniową oraz prostą implementację. Oparty jest on na algorytmie rozwiązania nieliniowego problemu najmniejszych kwadratów.

ZASTOSOWANIA SZTUCZNYCH SIECI NEURONOWYCH

- ✓ Klasyfikacji (obrazów, mowy, procesów,…)
- ✓ Regresji (funkcji matematycznych,…)
- ✓ Rozpoznawania (obiektów, ludzi, gestów, pisma…)
- ✓ Identyfikacji (obiektów, ludzi, gestów, pisma…)
- ✓ Przewidywania i prognozowania (np. szeregów czasowych, kursów walut,…)
- ✓ Sterowania i optymalizacji (np. różnych urządzeń ze sprzężeniem zwrotnym, …)
- ✓ Analizy i grupowania (np. w marketingu, sprzedaży, produkcji,…)
- ✓ Analizy ryzyka i opłacalności (np. kredytów i pożyczek bankowych,…)
- ✓ Doboru (np. surowców, składników, dla których nie jest znana technologia)
- ✓ … i wielu innych zagadnień, gdzie nie znamy algorytmu lub jego założenia są rozmyte albo złożoność obliczeniowa klasycznych rozwiązań zbyt duża.

Obecnie rynek rozwiązań sztucznej inteligencji (w tym sieci neuronowych) liczy sobie 20 000 000 000 USD rocznie i z roku na rok wykładniczo rośnie.

INSPIRACJE BIOLOGICZNE I ICH MODELE MATEMATYCZNE

- ✓ Sztuczne sieci neuronowe
- ✓ Algorytmy genetyczne
- ✓ Algorytmy ewolucyjne
- ✓ Logika rozmyta i systemy rozmyte
- ✓ Liczby niepewne
- ✓ Metody roju
- ✓ Teoria chaosu
- ✓ Inżynieria i reprezentacja wiedzy
- ✓ Kojarzenie, inteligencja i sztuczna świadomość

ANN

CZY SZTUCZNE SIECI NEURONOWE PROWADZĄ DO AI?