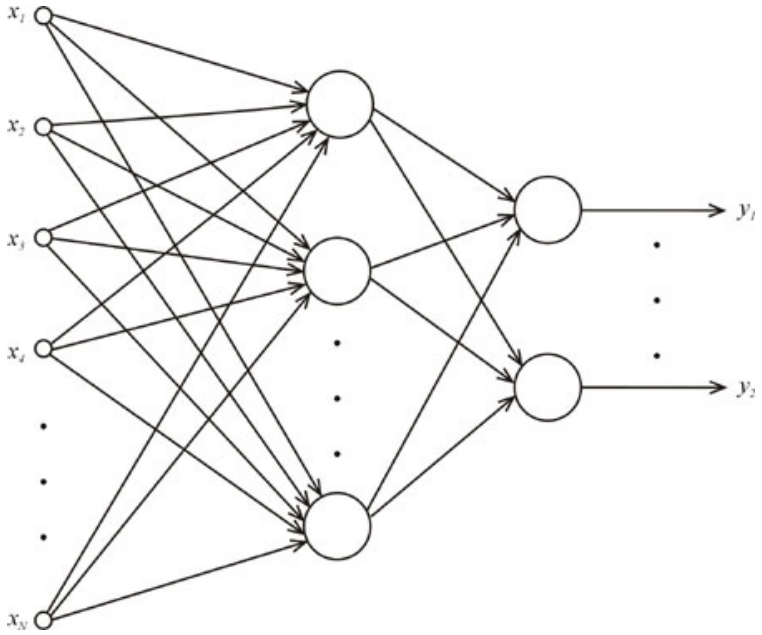
Metody Inżynierii Wiedzy
Systemy uczące się - podejście klasyczne - wykład 7

Adam Szmigielski
aszmigie@pjwstk.edu.pl
materiały: $ftp(public) : //aszmigie/MIW$

Sieć jednokierunkowa wielowarstwowa

Struktura sieci jednokierunkowej

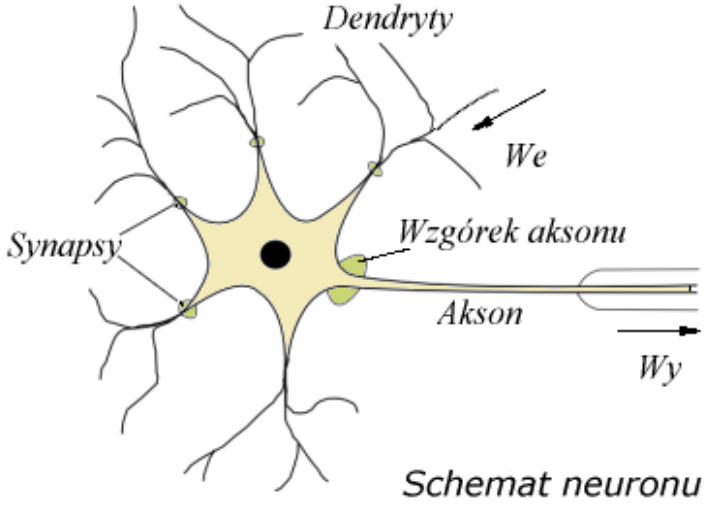


- Sieć szeregowo przetwarza informację z wejścia do wyjścia

Wyróżnia się następujące metody uczenia sieci:

- **uczenie z krytykiem** - na wejścia sieci podawane są dane. Wyjście sieci jest początkowo losowe. Jedyna informacja pochodząca od krytyka (binarna) dotyczy jedynie tego, czy sieć reaguje prawidłowo czy nie. Jeśli odpowiedź sieci jest prawidłowa wagi są wzmacniane, w przeciwnym wypadku wagi są osłabiane.
- **uczenie z nauczycielem** - w uczeniu z nauczycielem sieci podaje się parę wektorów uczących - wektor danych wejściowych oraz wektor danych wyjściowych. Wagi zmieniane są w kierunku minimalizacji błędu sieci.
- **uczenie samoorganizujące się** - sieci tego typu są układami dynamicznymi, zgodnie z którymi waga powiązań między dwoma neuronami wzrasta przy jednoczesnym stanie pobudzenia obu neuronów, w przeciwnym przypadku maleje.

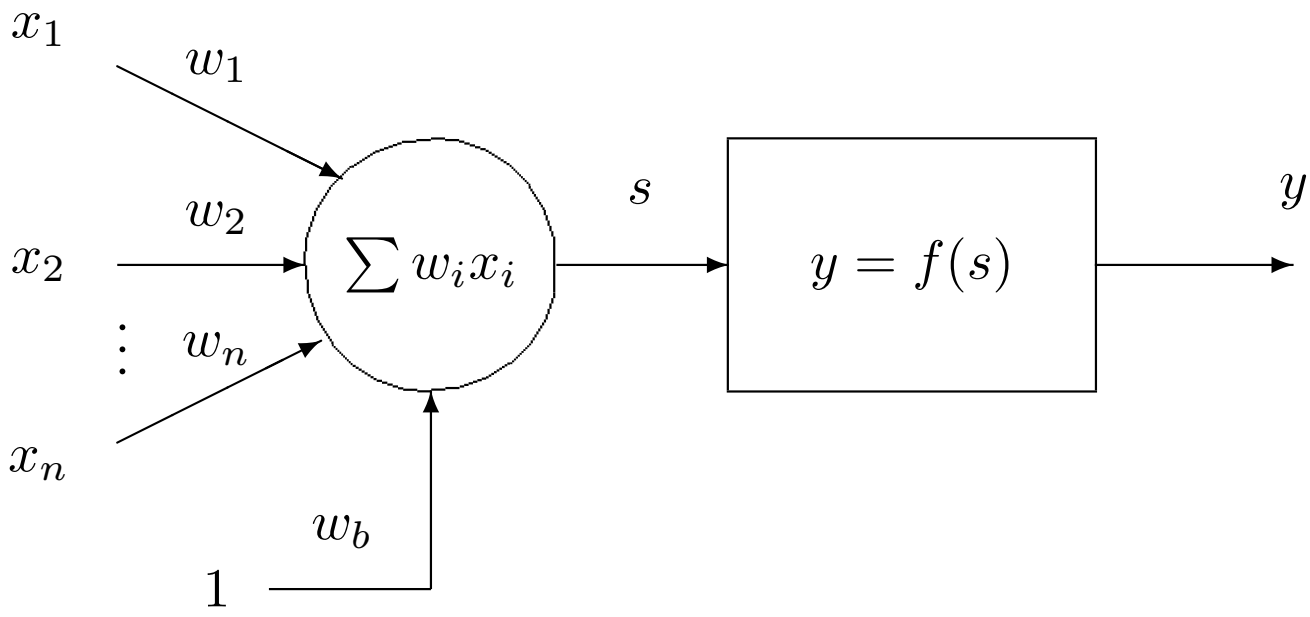
Na czym polega nauka sieci?



Schemat neuronu

- Na odpowiednim doborze struktury sieci
- Na doborze wag
  - w prostych przypadkach bezpośrednie określenie wartości wag,
  - zastosowanie algorytmów nauki sieci

Wymagania stawiane funkcji aktywacji neuronu



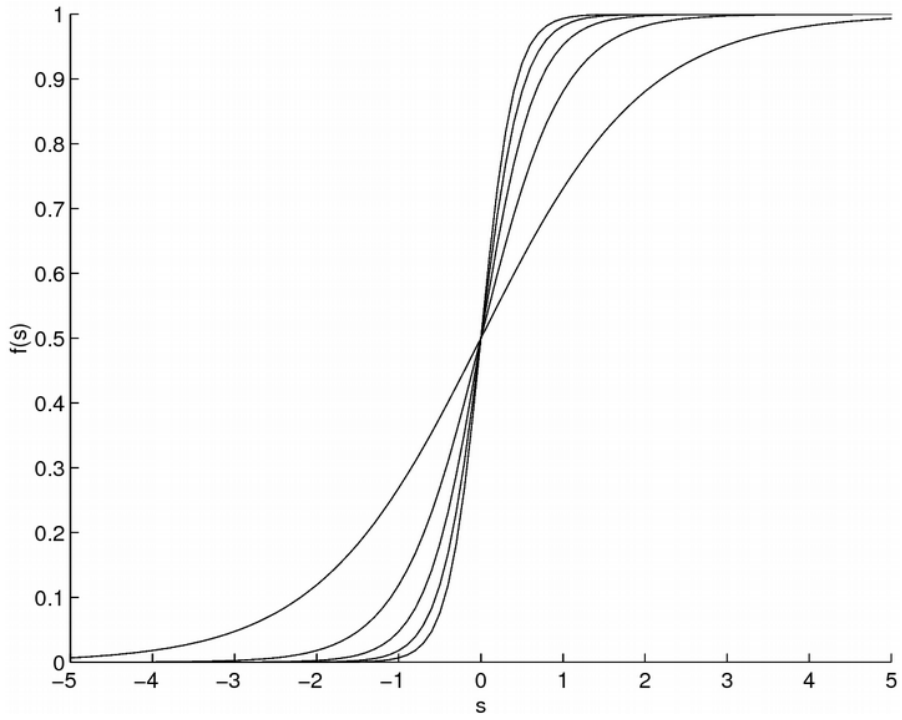
Rysunek 1: Schemat neuronu

- siągłość funkcji aktywacji
- posiadanie ciągłej pochodnej pochodną

Najczęściej stosowane funkcje aktywacji

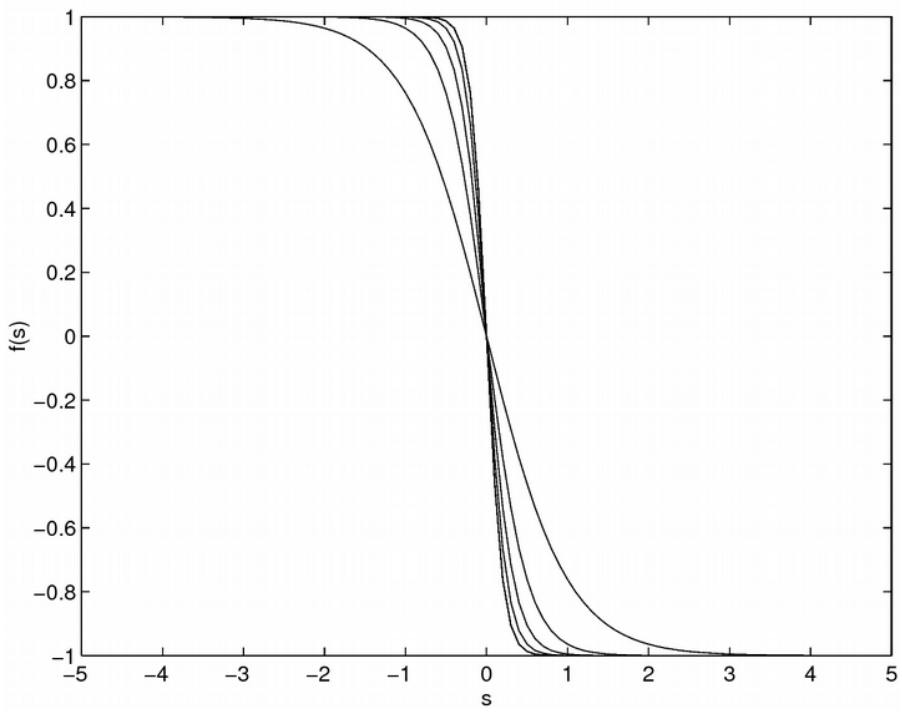
rodzaj	równanie	pochodna	zastosowanie
skokowa	$\mathrm{f(s)} = \begin{cases} 1 & s > 0 \\ 0 & s \leq 0 \end{cases}$	–	klasyfikacja
liniowa	$f(s) = s$	$f'(s) = 1$	skalowanie
unipolarna	$f(s) = \frac{1}{1+e^{-s}}$	$f'(s) = f(s)(1 - f(s))$	mod. nielin.
bipolarna	$f(s) = \frac{e^{s}-e^{-s}}{e^{s}+e^{-s}}$	$f'(s) = (1 - f(s) \cdot f(s))$	mod. nielin.
arc tang.	$f(s) = atan(s)$	$f'(s) = \frac{1}{1+s^2}$	mod. nielin.

Funkcja sigmoidalna (unipolarna)



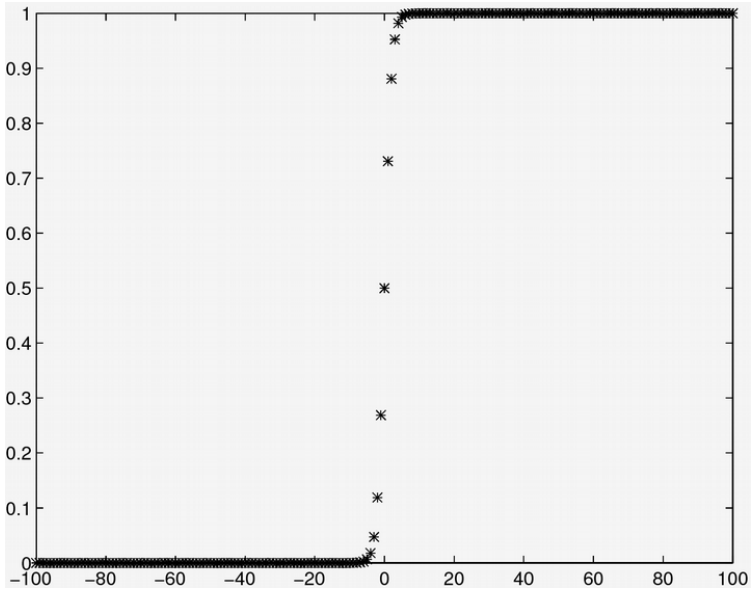
$f(s) = \frac{1}{1+e^{-\beta \cdot s}}$ dla $\beta = \{1, 2, 3, 4, 5\}$

Funkcja sigmoidalna (bipolarna)



$$f(s) = tanh(\beta \cdot s) = \frac{e^{\beta \cdot s} - e^{-\beta \cdot s}}{e^{\beta \cdot s} + e^{-\beta \cdot s}} \text{ dla } \beta = \{1, 2, 3, 4, 5\}$$

Inicjalizacja wag



Wykres funkcji sigmoidalnej dla zakresu sumy ważonej $s \in (-100, 100)$. Wagi powinny być tak zainicjalizowane aby:

- Suma ważona $s = \sum_i x_i \cdot w_i + w_b$ wypadała na nieliniowej części charakterystyki funkcji aktywacji,
- Wartość wag była różna od zera.

Wpływ danych wejściowych

Wektor wejściowy $X = [x_1, \ldots, n_n]$ wpływa na sumę zgodnie ze wzorem

$$s = \sum_{i}^{n} x_i \cdot w_i + w_b. \tag{1}$$

Może to powodować następujące problemy

- duże wartości x_i zwiększają wartość sumy s do nieliniowej części charakterystyki,
- Jeden, lub więcej x_i dominuje w sumie - co może powodować do niestabilności numerycznej (np. wielkości x_i są rzędu wielkości 10^{-6}, a x_j są rzędu wielkości 10^6, $i, j \in \{1, \ldots, n\}$)

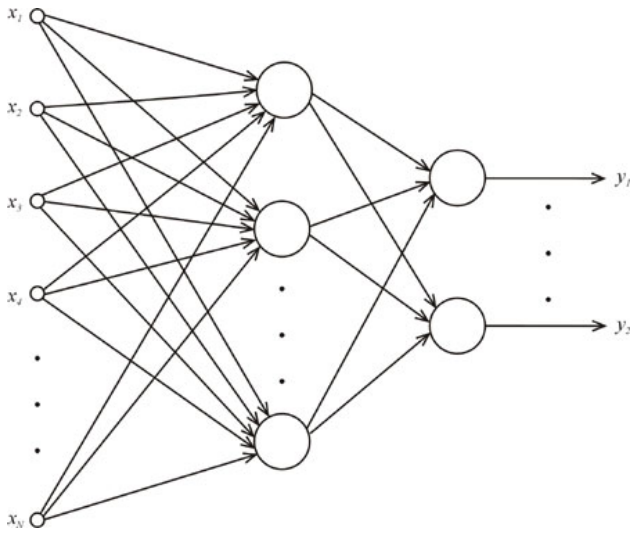
Sposoby eliminacji wpływu danych wejściowych

- analiza statystyczna z odpowiednim przeskalowaniem danych,
- możliwość doboru parametru β w funkcji aktywacji,
- zastosowanie pierwszej warstwy z liniową funkcją aktywacji

Wzorce uczące

- Sieć neuronowa, ze względu na nieliniowość funkcji aktywacji, może pełnić rolę uniwersalnego aproksymatora, odwzorowującego przestrzeń n-wymiarową w m-wymiarową $R^n \rightarrow R^m$,
- Parę wektorów danych $X = [x_1, x_2, \ldots, x_n]$ oraz $Y = [y_1, y_2, \ldots, y_m]$ nazywać będziemy **wzorcem**. Dane do nauki sieci neuronowej będziemy traktować jako zbiór wzorców,
- Nauczona sieć neuronowa powinna umieć odwzorowywać przestrzeń R^n w R^m zgodnie ze wzorcami.

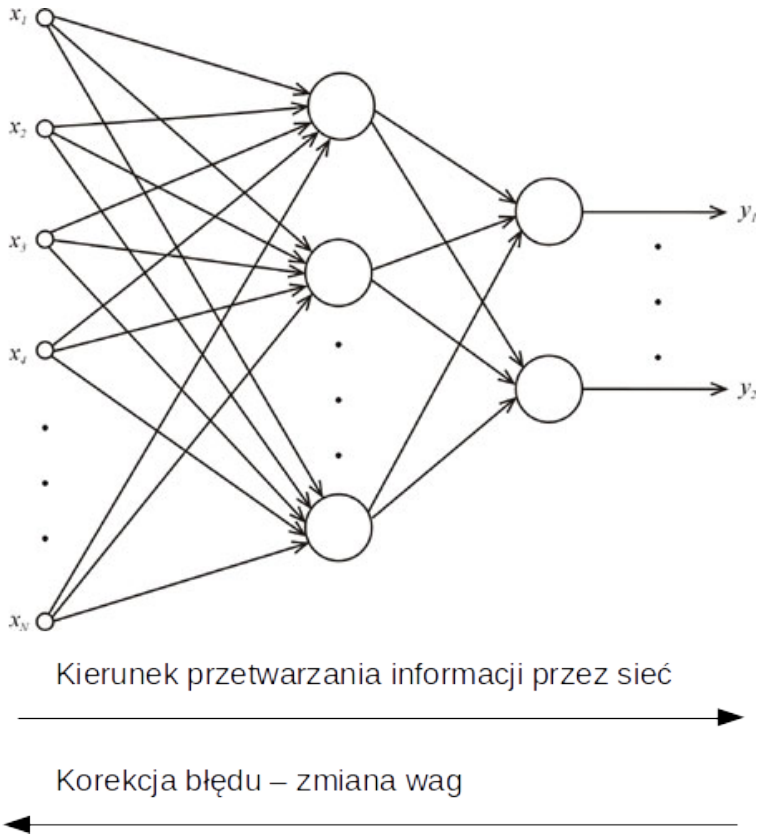
Sieć neuronowa jako czarna skrzynka



sieć neuronowa jako czarna skrzynka

- Znając wzorce sieć neuronową możemy traktować jako czarną skrzynkę,
- Wzorce uczące nic nie mówią o tym jaka powinna być odpowiedź neuronów w warstwie uczącej.

Wsteczna propagacja błędu



Kierunek przetwarzanej informacji jest przeciwny do kierunku propagacji błędu.

Algorytm propagacji wstecznej błędu

- **krok 1** Wybierz w losowy sposób wagi sieci neuronowej,
- **krok 2** Oblicz odpowiedź sieci - jeśli błąd sieci jest akceptowalny, zakończ naukę sieci,
- **krok 3** Dla każdego neuronu oblicz deltę

$$\delta = \frac{df(s)}{ds} \cdot e, \tag{2}$$

gdzie $\frac{df(s)}{ds}$ jest pochodną funkcji aktywacji sumy ważonej s, a e błędem neuronu. Może on być obliczony bezpośrednio jedynie dla warstwy wyjściowej sieci jako różnicę odpowiedzi sieci a żądanej wartości. W przypadku warstw ukrytych błąd jest estymowany w oparciu o warstwy poprzednie jako sumę iloczynów ważonych błędów w poprzednich neuronach e_p i wag połączeń synaptycznych pomiędzy

obecnym i poprzedzającym neuronem w_p.

$$e = \sum_p w_p \cdot e_p, \tag{3}$$

Dla każdego wejścia neuronu koryguje się wagę w zgodnie ze wzorem

$$w = w + \eta \cdot \delta \cdot we, \tag{4}$$

gdzie η jest współczynnikiem uczenia sieci, a we wartością wejścia biasu ($we = 1$). Przejdź do **kroku 2**.

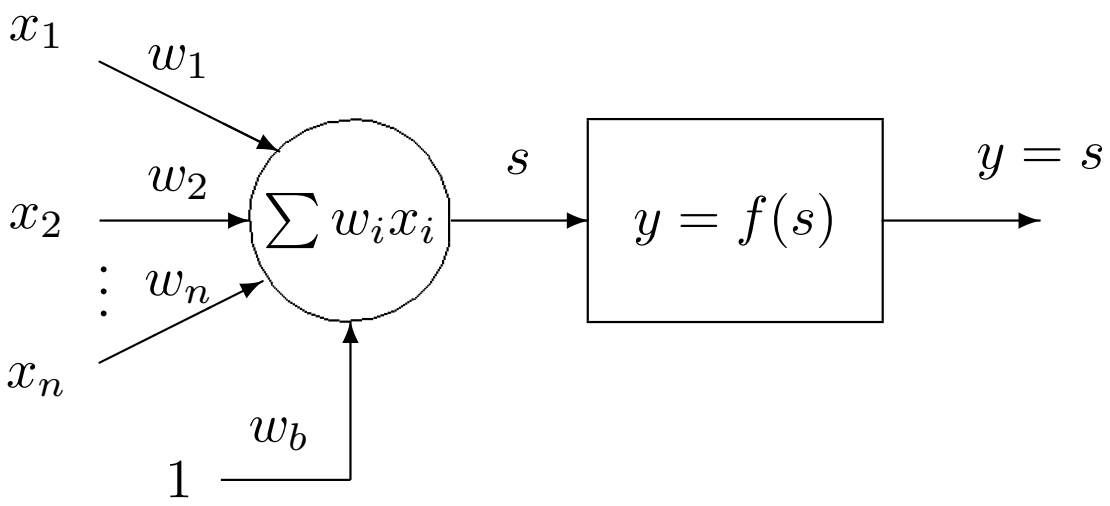
sposoby podawania wzorców

Ze względu na sposób podawania wzorców istnieją dwie metody:

1. **wsadowa -** korekcji wag dokonuje się dla wszystkich wzorców jako sumę poprawek.
2. **on line -** korekcji wag dokonuje się dla każdego wzorca osobno.

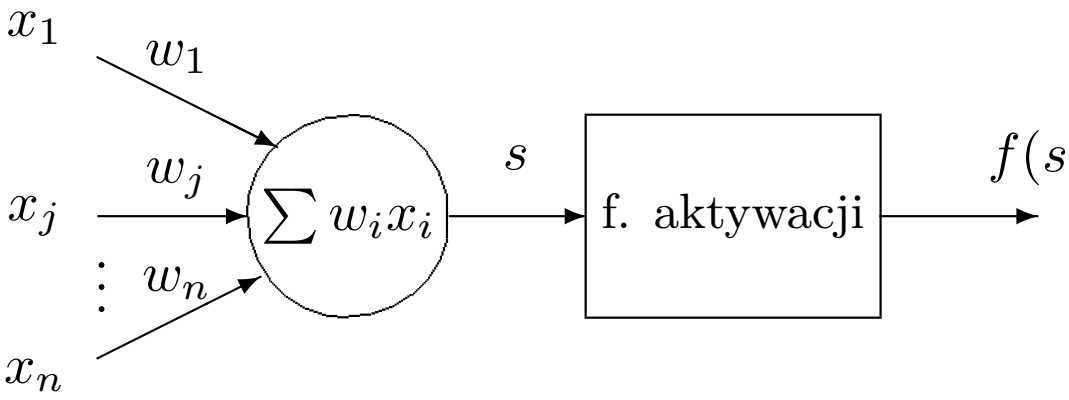
Cykl, w którym do nauki sieci zostały użyte wszystkie wzorce nosi nazwę *epoki*.

Perceptron z liniową funkcją aktywacji



- Wyjście opisane jest równaniem $y = w_1 \cdot x_1 + \ldots + w_n \cdot x_n + w_b$
- Błąd kwadratowy wynosi $e = \sum_i^n (w_i \cdot x_i - y)^2$

Minimalizacja błędu w neuronie



$$\nabla e = \begin{bmatrix} \frac{\partial e}{\partial w_1} \\ . \\ \frac{\partial e}{\partial w_j} \\ . \\ \frac{\partial e}{\partial w_n} \end{bmatrix}$$

$$e = (y - f(\sum_{i=1}^{n} w_i x_i))^2$$

$$\frac{\partial e}{\partial w_j} = 2 \cdot (y - f(\sum_{i=1}^{n} w_i x_i)) \cdot (-\frac{\partial}{\partial w_j} f(\sum_{i=1}^{n} w_i x_i))$$

$$\frac{\partial e}{\partial w_j} = -2 \cdot \underbrace{(y - f(\sum_{i=1}^{n} w_i x_i))}_{e} \cdot \underbrace{f'(\sum_{i=1}^{n} w_i x_i)}_{f'(s)} \cdot \underbrace{(\frac{\partial}{\partial w_j} \sum_{i=1}^{n} w_i x_i)}_{x_j}$$

Minimalizacja błędu w sieci jednokierunkowej

Dla sieci neuronowej w celu minimalizacji błądu kadratowego należy

- dla każdego neuronu należy obliczyć:
  - błąd odpowiedzi danego neuronu
- dla każdej wagi:
  - dokonać korekty w kierunku minimalizaji gradientu

Sieć neuronowa z liniową funkcją aktywacji

- W przypadku neuronu z liniową funkcją aktywacji model neuronu jest tożsamy z liniwym modelem autoregresyjnym
- Model liniowy daje podobne wyniki co neuron (bias pełni wówczas rolę drugiego współczynnika prostej

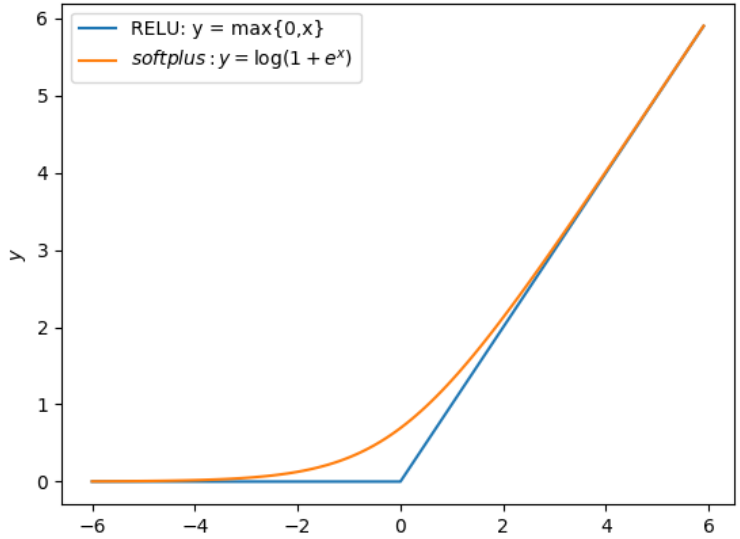
Co daje nieliniowość funkcji aktywacji?

- Czy sieć neuronową jest modelem parametrycznym ?
- Czy sieć neuronowa jest “białą”, “szarą” czy “czarną” skrzynką ?

Problem znikającego gradientu

- Dla typowych funkcji aktywacji $\sigma(x)$ i $tgh(x)$ wartości pochodnych są zawsze w $< 0, 1 >$.
- Jest to zjawisko zanikającego gradientu (ang. vanishing gradient) i powoduje problem z wyuczeniem cech na niższych warstwach.
- Był to jeden z głównych powodów spowolnienia rozwoju sieci neuronowych w latach 90.

Funkcja (ang. Rectified Linear Unit) RELU



- Funkcja RELU: $y = max\{x, 0\}$
- Funkcja RELU można aproksymować funkcją ciągłą *softplus*:

$$\zeta = \log(1 + e^x)$$

- Pochodną funkcji *softplus* jest funkcja *sigmoidalna*:

$$\frac{d}{dx}\zeta(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

Propagacja wsteczna dla funkcji RELU

- Ponieważ funkcja RELU w punkcie $(0, 0)$ nie posiada pochodnej, nie można użyć algorytmu propagacji wstecznej
- Odpowiedź sieci liczy się dla funkcji RELU, a pochodną oblicza się dla funkcji softplus.
- Uczymy się bardziej odpornych cech dzięki rzadkiej aktywacji jednostek,
- Uczenie trwa dużo szybciej.

Ile neuronów nieliniowych powinna mieć sieć

- Zależy od danych,
- Czym dane są bardziej skorelowane sieć wymaga zastosowania większej ilości neuronów z nieliniową funkcją aktywacji.

Twierdzenie o uniwersalnej aproksymacji

Niech $f : R^N \to R$ będzie ciągła i skończona. Dodatkowo niech $y(x) = \sum_i g(w_i^T x)$, gdzie g jest ciągła, ograniczona i monotoniczna.

Wówczas dla każdego $\varepsilon > 0$ istnieje taki zbiór wag $\{w_i\}$, że dla każdego $x \in R^N$ zachodzi:

$$|y(x) - f(x)| < \varepsilon$$

Wystarczy zatem tylko jedna warstwa ukryta, aby z dowolną precyzją aproksymować każdą funkcję ciągłą.

Identyfikacja parametryczna i strukturalna

Dzięki nieliniowości sieć potrafi dokonać aproksymacji strukturalnej.

Zadania na laboratoria

1. Wczytaj dane z pliku *daneXX.txt*[a]. Zaproponuj i zrealizuj podział tych danych na *dane treningowe* i *dane testowe*,
2. Zaproponuj optymalną sieć neuronową aproksymującą wczytane dane. Użyj nieliniowej funkcji aktywacji w warstwie ukrytej (np. tanh lub sigmoidalna). Stosując propagację wsteczną błędu wytrenuj sieć metodą wsadową.
3. Oceń działanie sieci pod kątem dopasowania (zbyt małe, optymalne, zbyt duże dopasowanie).
4. Zmień sposób podawania sieci z wsadowej no on line. Wytrenuj sieć. Oceń działanie sieci.
5. Zaproponuj sieć z funkcją aktywacji RELU. Stosując propagację wsteczną błędu wytrenuj sieć metodą wsadową.

[a]gdzie XX jest numerem zestawu. W każdej linii pliku pierwsza liczba określa wejście a druga wartość wyjścia

Proszę nie używać dostępnych bibliotek sieci neuronowych. Do implementacji sieci proszę użyć macierzy stosując bibliotekę *numpy*