Laboratorium 1
Programowanie Sieciowe

Marek Bazan

III rok

Semestr letni 2021/2022

Plan zajęć

1. Literatura
2. Wstęp do Pythona
3. Matematyczny model nauronu.
4. Dane Iris,
5. Klasyfikatory binarne:
   - Perceptron prosty,
   - Adaline.
6. Uczenie za pomocą reguły delta.
7. Zadanie domowe.

Literatura

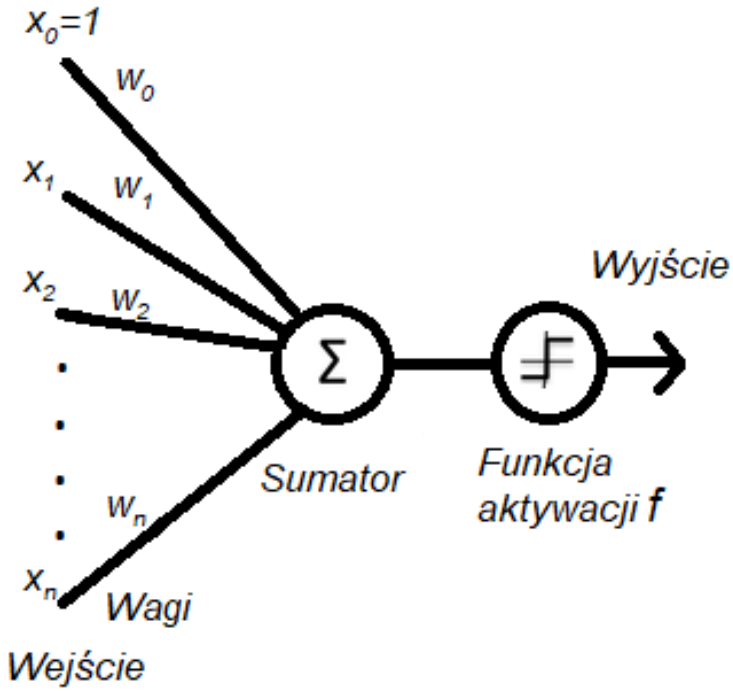
1. Wstęp do Pythona
   -  (Python początek) https://www.youtube.com/watch?v=BBu6ZoAHIwI&t=6833s
   -  (Python + numpy) https://cs231n.github.io/python-numpy-tutorial/
2. Dane Iris
   -  (Dane Iris) https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

Potrzebne podstawy Pythona

- (Python początek) `https://www.youtube.com/watch?v=BBu6ZoAHIwI&t=6833s`
- (Python + numpy) `https://cs231n.github.io/python-numpy-tutorial/`

1. Zmienne i podstawienie,
2. Instrukcja warunkowa,
3. Pętle `for`, `while`,
4. Typy danych,
5. Funkcje,
6. Operacja na listach,
7. Operacje na macierzech.

Neuron



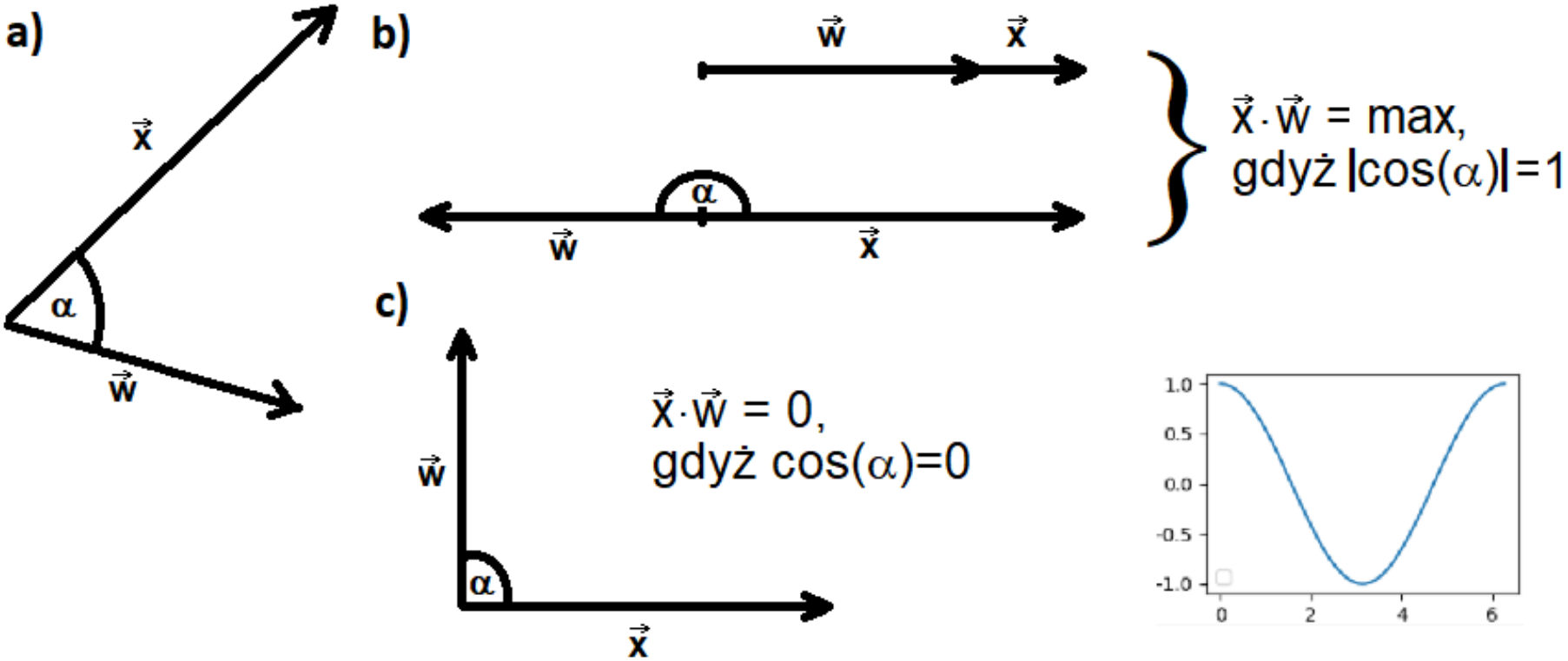
Rysunek: Model matematyczny neuronu.

1. Aktywacja neuronu to iloczyn skalarny wektora wag $\vec{w} = (w_0, w_1, \ldots, w_n)$ i wejścia $\vec{x} = (1, x_1, \ldots, x_n)$ czyli $\vec{x} \cdot \vec{w}$,
2. Wyjście neuronu to $f(\vec{x} \cdot \vec{w})$,

Co mierzy neuron?

Co mierzy neuron?

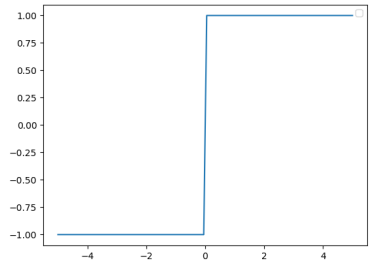
3. Ponieważ $\vec{x} \cdot \vec{w} = |\vec{x}||\vec{w}| \cos(\alpha)$ gdzie α jest kątem pomiędzy wektorami $\vec{x}$ i $\vec{w}$



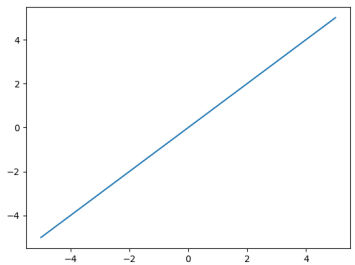
Rysunek: Aktywacja neuronu w zależności od kąta α pomiędzy wektorem wag $\vec{w}$ i wektorem wejścia $\vec{x}$. Maksymalną aktywację (co do wartości bezwzględnej) uzyskujemy, gdy $\alpha = 0$ i $\alpha = \pi$ czyli, gdy wektory $\vec{w}$ i $\vec{x}$ są współliniowe. Natomiast, gdy $\alpha = \frac{\pi}{2}$ aktywacja jest równa 0.

Funkcje aktywacji

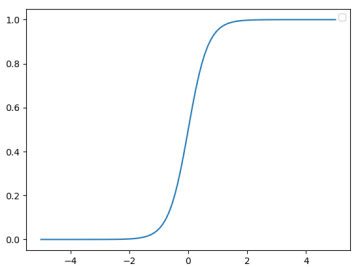
1. Funkcja Heaviside'a $f(x) = \begin{cases} -1 & \text{dla} \quad x \leqslant 0 \\ 1 & \text{dla} \quad x > 0 \end{cases}$



2. Funkcja liniowa $f(x) = x$

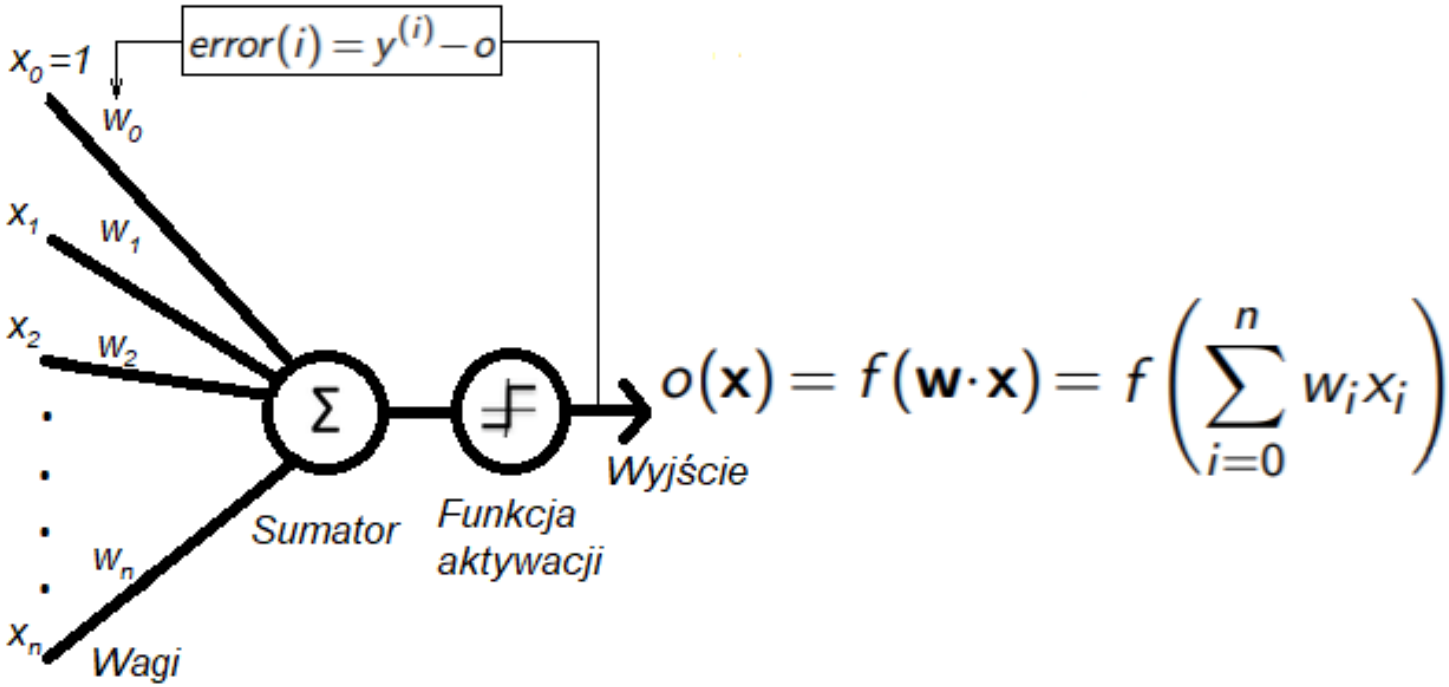


3. Funkcja sigmoidalna $f(x) = \frac{1}{1+e^{(-\alpha x)}}$



Perceptron - uczenie za pomocą reguły perceptronu

Dla perceptronu f jest funkcją Heaveside'a.
Niech $Z = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$ oznacza zbiór danych.



1. Dla ustalonej liczby epok $\mathtt{n_{epoch}}$ iterujemy się po zbiorze Z
2. Dla $i = 1, \ldots, N$ modyfikujemy wagi obliczając

$$\begin{aligned} error(i) &= y^{(i)} - o(x^{(i)}) \\ \Delta w &= \eta \cdot error(i) \cdot x^{(i)} \\ w &= w + \Delta w \end{aligned}$$

Perceptron - uczenie za pomocą reguły perceptronu (2)

W powyższym algorytmie

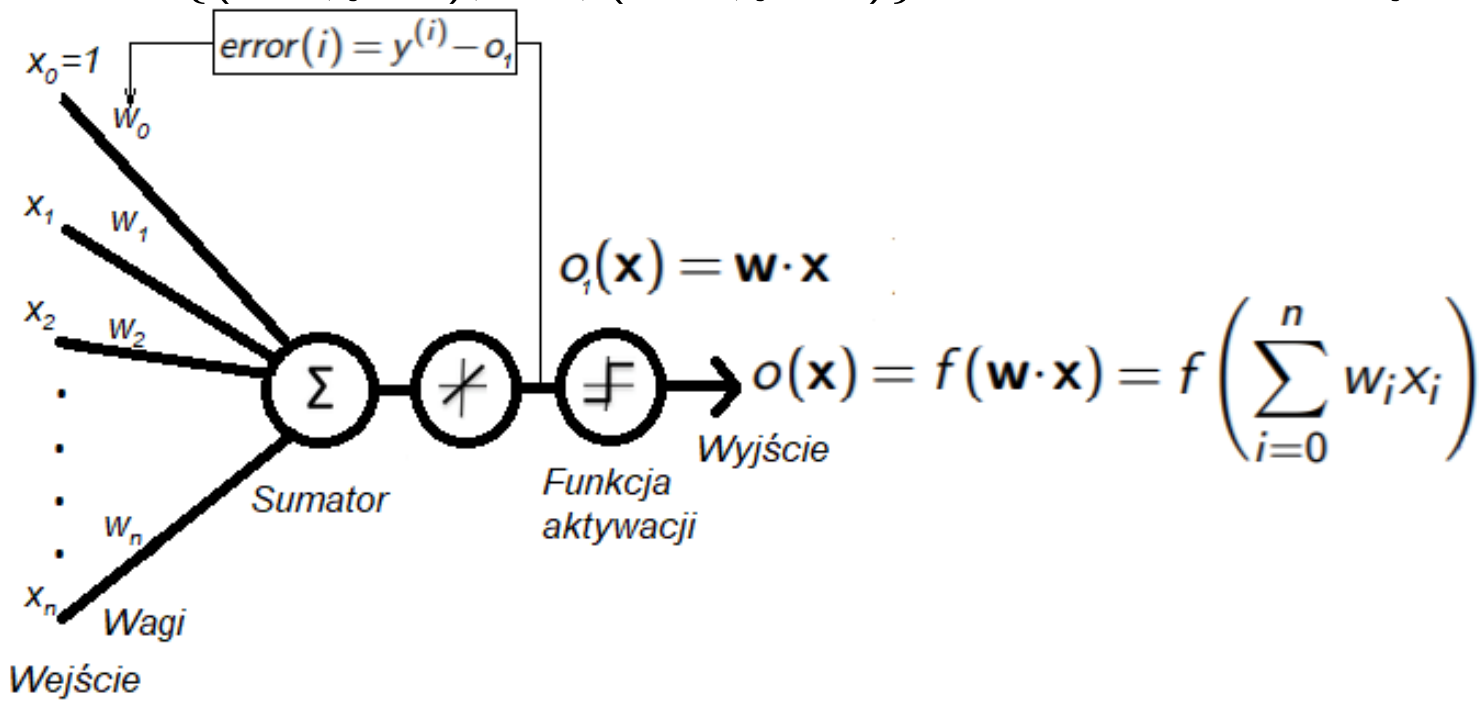
- η jest współczynnikiem uczenia z przedziału $(0, 1)$ np równym $0.1, 0.01$.
- Zbiór Z konstruujemy w ten sposób, że wybieramy tylko 80% danych z każdej klasy. Pozostałe 20% danych umieszczamy w zbiorze testowym i na nich sprawdzamy preceptron po zakończeniu uczenia.
- Działa tylko dla dwóch klas.

Klasyfikacja 2 klas z Iris Data

1. Bazując na stronie `https://github.com/rasbt/pattern_classification/blob/master/machine_learning/singlelayer_neural_networks/singlelayer_neural_networks.ipynb` zaimplementować klasyfikację osobników dowolnej klasy dla danych Iris za pomocą modelu perceptronu.

Adaline - uczenie za pomocą reguły Delta

Dla modelu Adeline f jest również funkcją Heaveside'a. Natomiast do sprzężenia zwrotnego podajemy tylko aktywację.
Niech $Z = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$ oznacza zbiór danych.



1. Dla ustalonej liczby epok $\mathtt{n_{epoch}}$ iterujemy po zbiorze Z.
2. Dla $i = 1, \ldots, N$ modyfikujemy wagi obliczając

$$\begin{aligned} error(i) &= y^{(i)} - o_1(x^{(i)}) \\ \Delta w &= \eta \cdot error(i) \cdot x^{(i)} \\ w &= w + \Delta w \end{aligned}$$

Adaline - uczenie za pomocą reguły Delta (2)

W powyższym algorytmie

- η jest współczynnikiem uczenia z przedziału $(0, 1)$ np równym $0.1, 0.01, 0, 001$, *itp*.
- Zbiór Z konstruujemy w ten sposób, że losujemy 80% danych z każdej klasy. Pozostałe 20% danych umieszczamy w zbiorze testowym i na nich sprawdzamy Adaline'a po zakończeniu uczenia wyliczając $o(x)$ dla każdego x ze zbioru walidacyjnego.
- Działa również tylko dla dwóch klas.

Klasyfikacja 2 klas z Iris Data

1. Bazując na stronie `https://github.com/rasbt/pattern_classification/blob/master/machine_learning/singlelayer_neural_networks/singlelayer_neural_networks.ipynb` zaimplementować klasyfikację osobników dowolnej klasy dla danych Iris.

Klasyfikacja 3 klas z Iris Data

2. Wykorzystując implementacje z poprzedniego punktu zbudować klasyfikator dla 3 klas z danych Iris.
   - 2.1 Budujemy trzy klasyfikatory bazując na poprzednim punkcie
   - 2.2 Każdy z trzch klasyfikatorów ma klasyfikować jako pozytywną inną klasę oraz jako negatywną pozostałe dwie klasy.
   - 2.3 Trenujemy wszystkie klasyfikatory na odpowiednio przygotowanych zbiorach
   - 2.4 Klasyfikacja danych z trzech klas polega na podaniu danej do wszystkich trzch klasyfikatorów i wybraniu tej klasy gdzie wskazanie pozytywne jest największe.