Obliczenia i systemy inspirowane biologicznie

WYKŁAD 1

Tematyka

- (Sztuczne) sieci neuronowe.
- Algorytmy genetyczne i obliczenia ewolucyjne.
- Inteligencja grupowa (stadna): algorytmy mrówkowe, optymalizacja rojem cząstek.
- Algorytmy komórkowe.
- Sztuczne systemy immunologiczne.
- Obliczenia niekonwencjonalne.

Kategorie metod inspirowanych Naturą

- **Metody odtwarzające zjawiska występujące w Naturze (m.in. biologiczne)**, np. automaty komórkowe.
- **Metody motywowane zjawiskami występującymi w Naturze (m.in. biologicznymi)**, np. algorytmy genetyczne, sieci neuronowe.
- **Metody korzystające ze zjawisk występujących w Naturze (m.in. biologicznych)** – obliczenia niekonwencjonalne.

Środowisko R

- R jest wolnym (na licencji GNU GPL - możliwość wykorzystywania także w zastosowaniach komercyjnych) zaawansowanym środowiskiem oraz językiem programowania przeznaczonym do wykonywania obliczeń statystycznych i numerycznych oraz analizy i wizualizacji danych.
- Język R powstał w 1997 r. na Uniwersytecie w Auckland (Nowa Zelandia).
- R jest językiem interpretowanym.

RStudio

- RStudio jest zintegrowanym środowiskiem programistycznym (IDE) dla instalacji R.
- RStudio umożliwia m.in.:
  - wygodne zarządzanie plikami źródłowymi oraz całymi projektami,
  - korzystanie z licznych rozszerzeń możliwości konsoli,
  - korzystanie z zintegrowanego systemu pomocy i narzędzi wspomagających zarządzanie generowanymi plikami graficznymi.

Język R

- Podstawowe typy danych:
  - wartości logiczne (*logical*)
  - bajty (*raw*)
  - liczby całkowite (*integer*)
  - liczby rzeczywiste (*double*)
  - liczby zespolone (*complex*)
  - łańcuchy znaków (*character*)
  - pusty (*NULL*)

Język R

- Typy złożone:
  - macierze (*matrix*)
  - tablice (*array*)
  - szeregi czasowe (*ts*)
  - czynniki (*factor*)
  - ramki danych (*data.frame*)
  - formuły (*formula*)

Język R

- Stałe logiczne

```
TRUE, FALSE
```

- Wybrane wartości specjalne

```
Inf, -Inf, NaN
```

- Sprawdzenie typu danej
  - przykład:

```
typeof("Język R");
```

Język R

- Operator przypisania
  - przykłady:

```
x<-5;
b<-TRUE
```

Język R

- Tworzenie wektora
  - przykłady:

```
x<-c(1, 2, 4, 2, 5);
y<-1:10
```

- Sprawdzenie długości wektora
  - przykład:

```
length(x);
```

Język R

- Indeksowanie wektorów zaczyna się od 1

- Dostęp do elementów wektora
  - przykłady:

```
x[2];
y[3:5];
```

- Tworzenie macierzy
  - przykłady:

```
x<-matrix(1:10, nrow=2, ncol=5);
y<-matrix(c('a','b','a','d'),
ncol=2, byrow=TRUE);
```

- Sprawdzenie rozmiarów macierzy
  - przykłady:

```
ile_kolumny<-ncol(x);
ile_wierszy<-nrow(x);
```

Język R

- Dostęp do elementów macierzy
  - przykłady:

```
x[2,2];
x[,1:2];
x[1,]
```

Język R

- Transpozycja macierzy
  - przykład:

```
t(x);
```

- Mnożenie macierzy
  - przykład:

```
x %*% y;
```

Język R

- Tablice (*array*):
  - są uogólnieniem macierzy,
  - mogą mieć wiele wymiarów.
- Macierze *(matrix)* są tablicami dwuwymiarowymi.

Język R

- Komentarze
  - przykład:

```
#komentarz do końca linii
```

- Konkatenacja i wyświetlanie łańcuchów znaków
  - przykład:

```
cat("Wartość zmiennej x", x, "\n");
```

Język R

- System pomocy
  - przykład:

```
?cat
```

Język R

- Operatory arytmetyczne
  - dodawanie +
  - odejmowanie -
  - mnożenie *
  - dzielenie /
  - potęgowanie ^
  - reszta z dzielenia %%
  - dzielenie całkowite %/%
- Operatory arytmetyczne są zwektoryzowane (operacje wykonywane są element po elemencie)

Język R

- Operatory logiczne
  - negacja !
  - suma logiczna |
  - iloczyn logiczny &
- Operatory logiczne są zwektoryzowane (operacje wykonywane są element po elemencie)

Język R

- Operatory relacyjne
  - mniejszy <
  - większy >
  - mniejszy lub równy <=
  - większy lub równy >=
  - równy ==
  - różny !=
- Operatory relacyjne są zwektoryzowane (operacje wykonywane są element po elemencie)

Język R

- Tworzenie funkcji:
  - przykład:

```
kwadrat <- function(x)
{
   x^2;
}
```

Język R

- Funkcje mogą być zapisywane w oddzielnych plikach źródłowych (skryptach), np.:

```
kw.R
```

- Aby skorzystać z funkcji należy wywołać funkcję **source**, np.:

```
source('kw.R');
```

Język R

- Instrukcja warunkowa IF:
  - przykład:

```
if(a>0)
{
    b=5;
}
```

- Instrukcja warunkowa IF-ELSE:
  - przykład:

```
if(a>0)
{
   b=5;
}
else
{
   b=10;
}
```

Język R

- Instrukcja pętli WHILE:
  - przykład:

```
while(i<10)
{
   cat("i: ", i, "\n");
   i<-i+2;
}
```

Język R

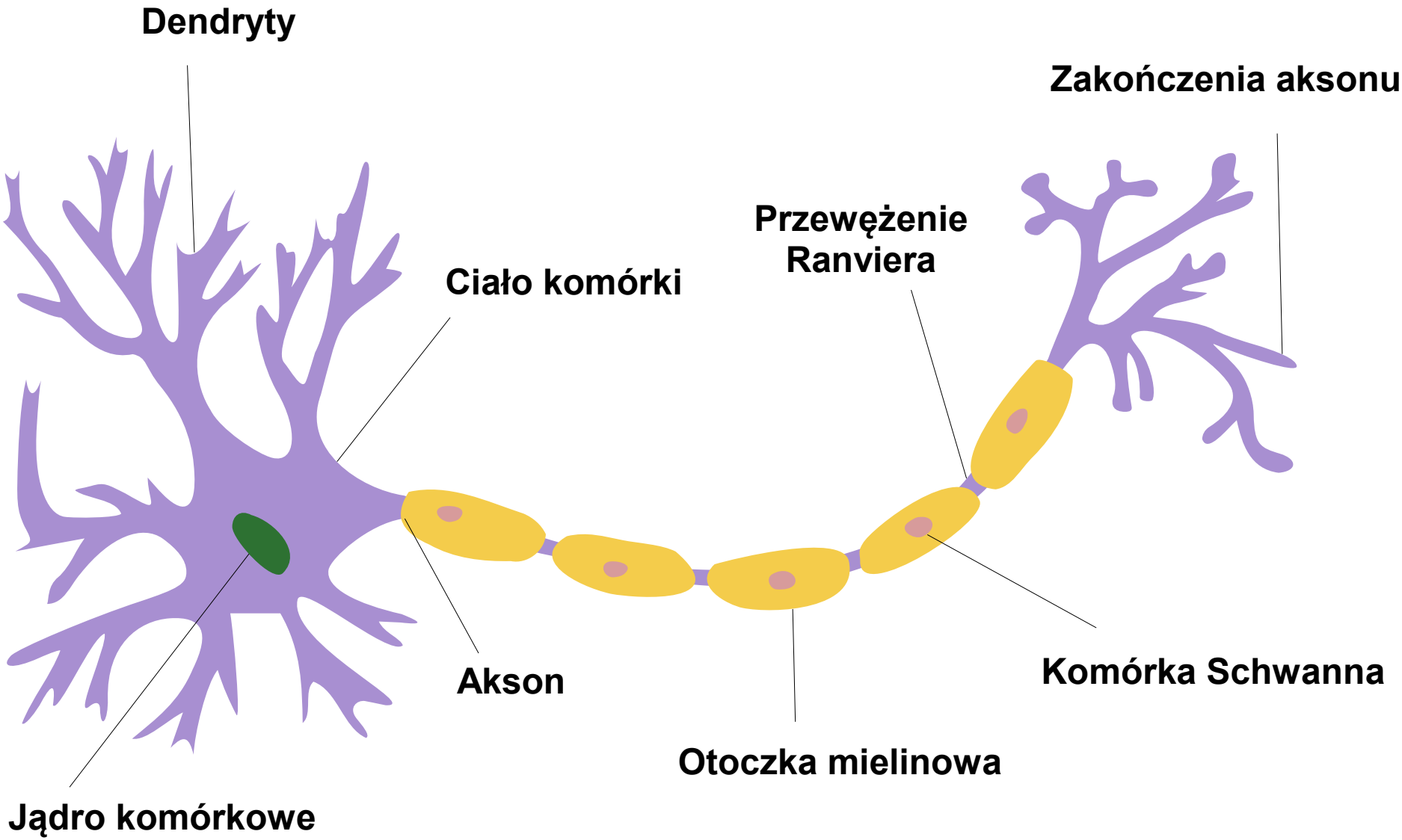
- Instrukcja pętli FOR:
  - przykład:

```
for(i in 1:100)
{
   cat("i: ", i, "\n");
}
```

Język R

- Wybrane funkcje wejścia/wyjścia:
  - **read.table** - wczytywanie danych tabelarycznych z plików tekstowych
  - **write.table** - zapis danych tabelarycznych do plików tekstowych

Neuron biologiczny



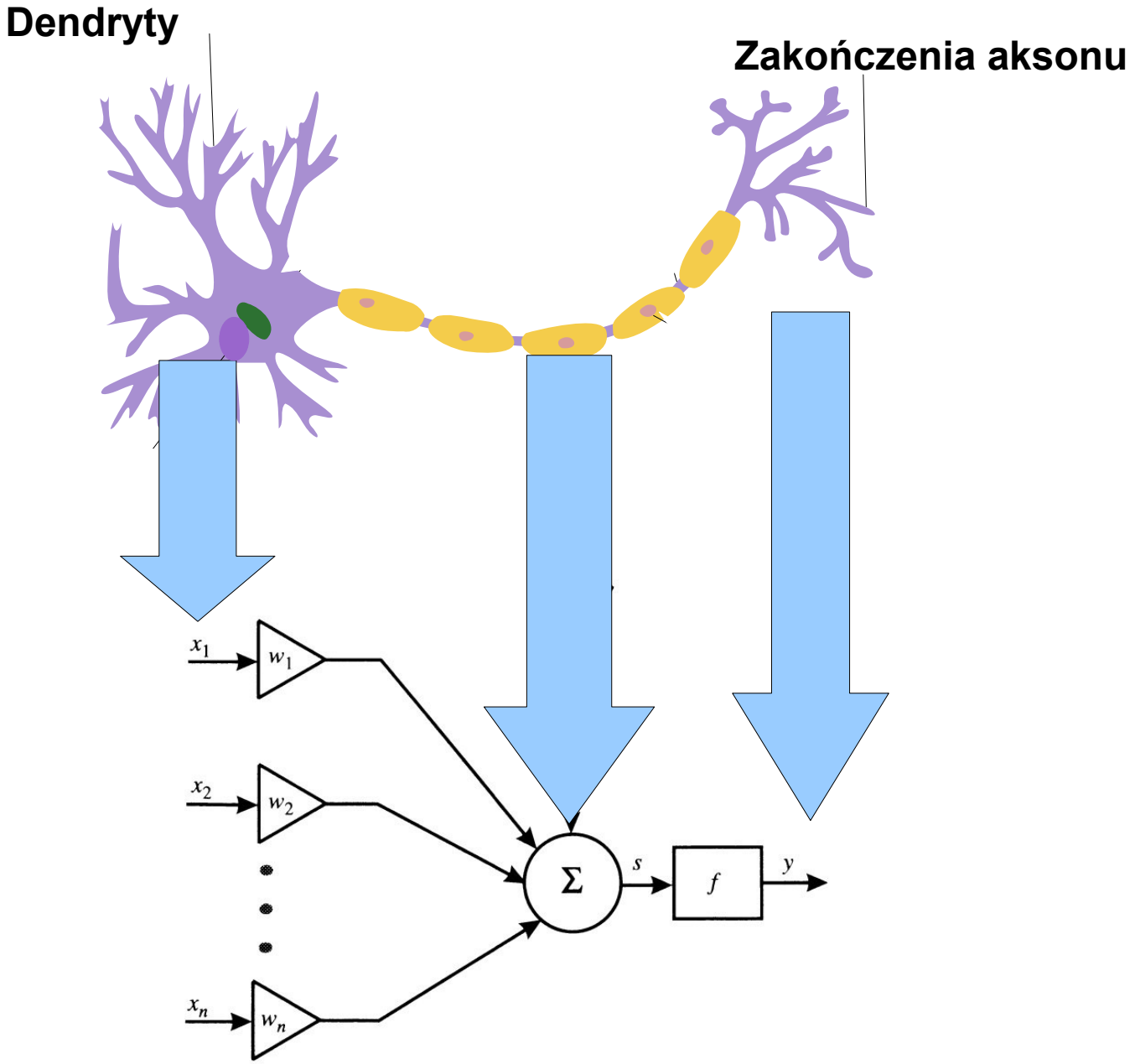
Źródło: https://pl.wikipedia.org/wiki/Neuron#/media/File:Neuron,_LangNeutral.svg

Neuron biologiczny

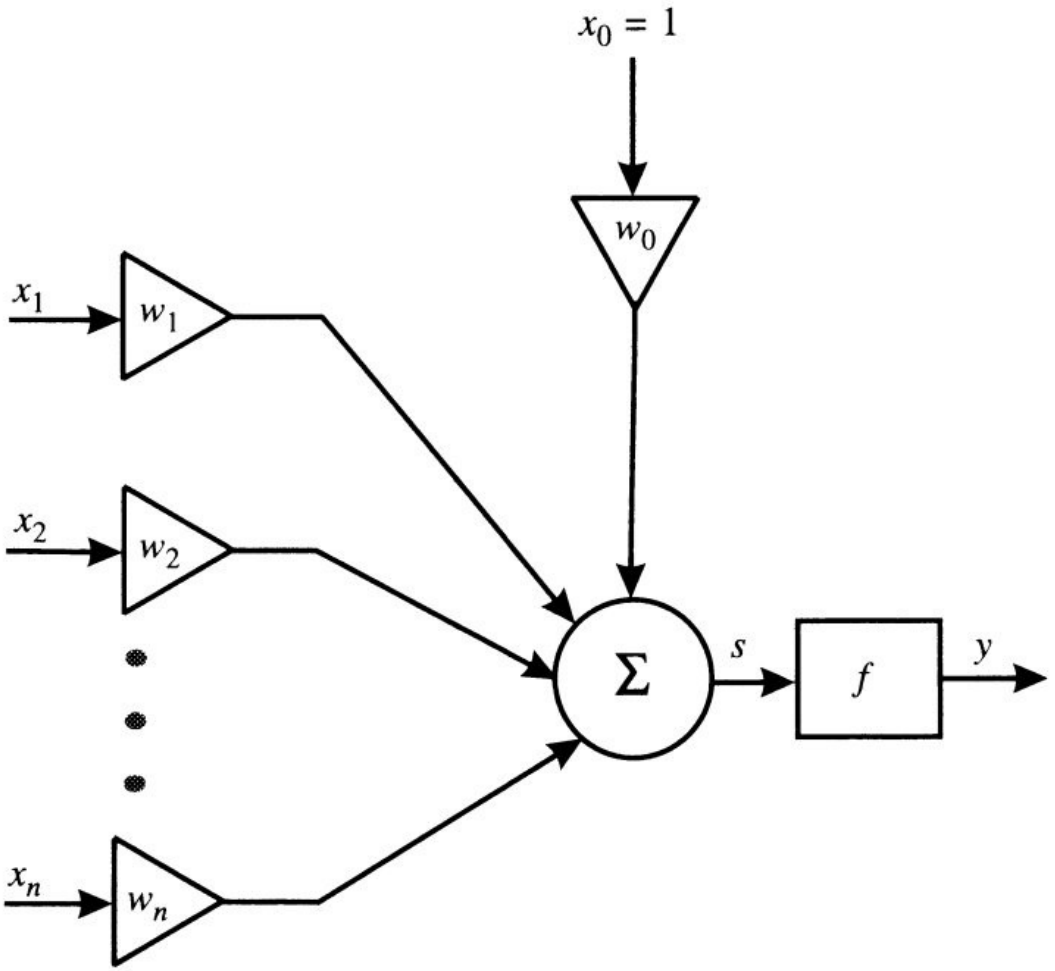
- Neurony kontaktują się między sobą poprzez synapsy tworząc sieci neuronowe.
- Nośnikiem informacji są sygnały elektryczne.

- Dendryty → Synapsy położone na dendrytach służą do odbierania informacji z innych neuronów ↔ **Wejścia**.
- Zakończenia aksonu → Synapsy położone na na zakończeniach aksonu służą do przekazywania informacji do innych neuronów ↔ **Wyjścia**.

Od neuronu biologicznego do sztucznego neuronu



Model neuronu



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

f - funkcja aktywacji

$x_1, x_2, \ldots, x_n$ – sygnały wejściowe

$w_0, w_1, \ldots, w_n$ – wagi

y – sygnał wyjściowy

Źródło: L. Rutkowski „Metody i techniki sztucznej inteligencji". PWN, Warszawa, 2005.

Przykładowe funkcje aktywacji

- Unipolarna funkcja skokowa:

$$f(s)=\begin{Bmatrix} 1 & dla & s>0 \\ 0 & dla & s\leqslant 0 \end{Bmatrix}$$

- Bipolarna funkcja skokowa:

$$f(s)=\begin{Bmatrix} 1 & dla & s>0 \\ -1 & dla & s\leqslant 0 \end{Bmatrix}$$

- Bipolarna funkcja sigmoidalna:

$$f(s)=\tanh(\beta\, s)$$

Przykładowe funkcje aktywacji

- Obcięta funkcja liniowa:

$$f(s)=\begin{Bmatrix} 1 & dla & s>1 \\ v & dla & -1\leqslant v\leqslant 1 \\ -1 & dla & s<-1 \end{Bmatrix}$$

Perceptron

- Perceptron to neuron z unipolarną skokową funkcją aktywacji.
- Perceptron najlepiej opisuje rzeczywisty neuron.
- Perceptron dokonuje jedynie liniowego podziału przestrzeni. Przy jego pomocy nie można rozdzielić przypadków, które nie są liniowo separowalne.

Uczenie neuronu

Uczenie nadzorowane (z nauczycielem)

Ciąg uczący *(k* przykładów)*:*

$$(\mathbf{x}(t), d(\mathbf{x}(t))$$

$\mathbf{x}(t)=[x_1(t),x_2(t), ..., x_n(t)]$ – wektor sygnałów wejściowych

$d(\mathbf{x}(t))$ – wartość wzorcowa sygnału wyjściowego dla wektora sygnałów wejściowych

$t=1,2, ..., k$

Uczenie neuronu

1) Wybieramy w sposób losowy wagi początkowe neuronu. Ustawiamy t=1.

2) Na wejścia neuronu podajemy wektor uczący:

$$\mathbf{x}(t)=[x_1(t),x_2(t), ..., x_n(t)]$$

3) Obliczamy wartość wyjściową y neuronu.

4) Porównujemy wartość wyjściową y z wartością wzorcową $d(\mathbf{x}(t))$ znajdującą się w ciągu uczącym.

5) Dokonujemy modyfikacji wag wg. zależności:

$$\textit{jeżeli}\ \ y \neq d(\boldsymbol{x}(t)): \quad w_i(t+1)=w_i(t)+d(\boldsymbol{x}(t))\cdot x_i(t)$$
$$\textit{jeżeli}\ \ y = d(\boldsymbol{x}(t)): \quad w_i(t+1)=w_i(t)$$

6) Zwiększamy t o 1 i jeśli t nie jest większe od k wracamy do pkt. 2.

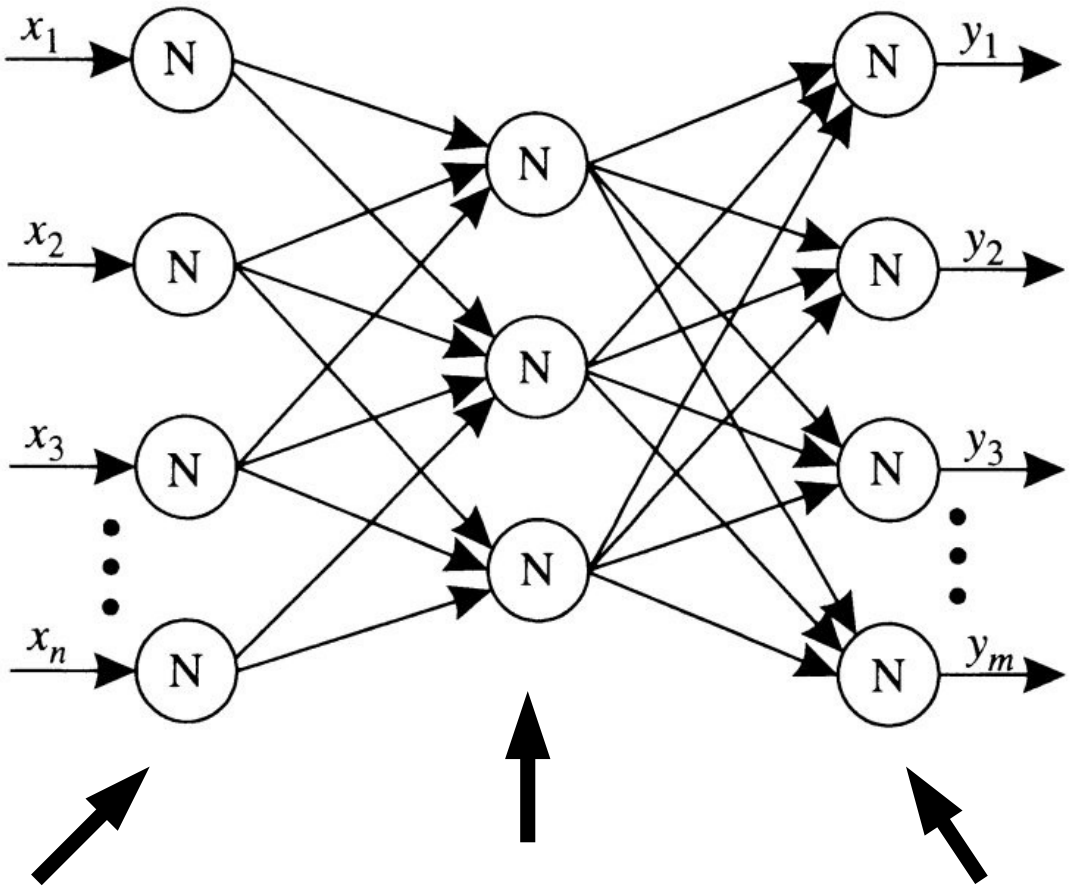
Uczenie neuronu

Algorytm uczenia powtarza się tak długo, aż dla wszystkich wektorów wejściowych wchodzących w skład ciągu uczącego błąd na wyjściu będzie mniejszy od założonej tolerancji.

Wykonanie jednej pętli algorytmu dotyczy tzw. jednej epoki, na którą składają się dane tworzące ciąg uczący.

Algorytm może być wielokrotnie stosowany dla tego samego ciągu uczącego, aż do spełnienia warunku zakończenia (konieczne może więc być podanie wielu epok).

Wielowarstwowe sieci neuronowe



WARSTWA WEJŚCIOWA

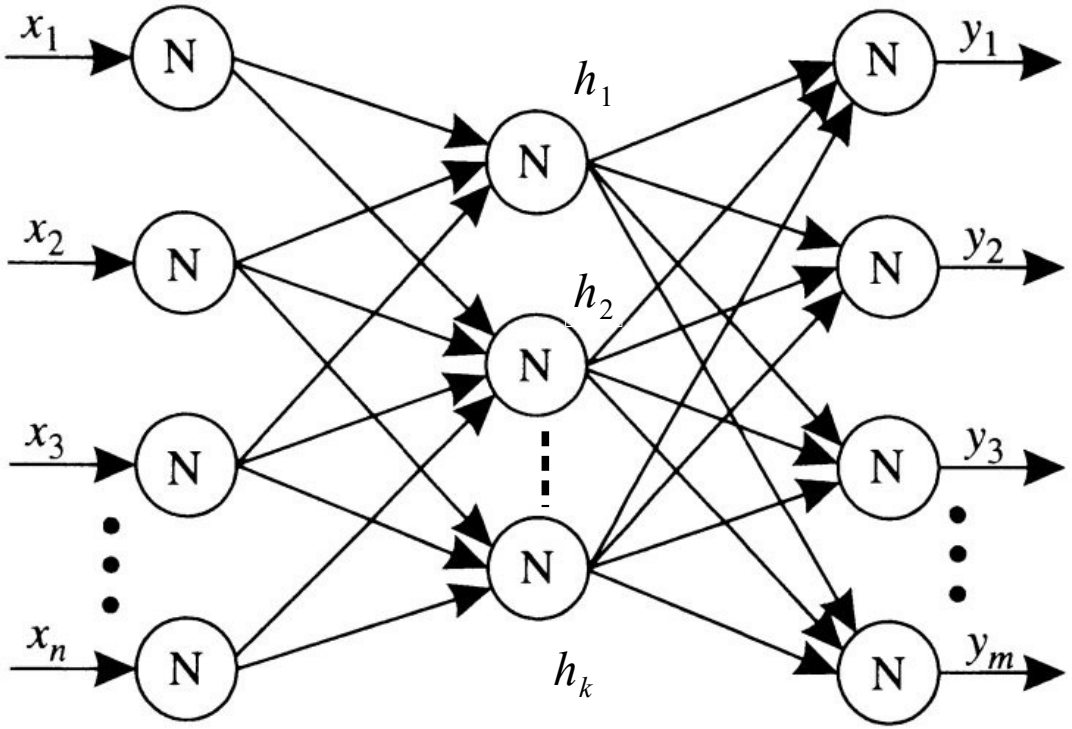
WARSTWA UKRYTA

WARSTWA WYJŚCIOWA

Źródło: L. Rutkowski „Metody i techniki sztucznej inteligencji”. PWN, Warszawa, 2005.

Wielowarstwowe sieci neuronowe

- Modyfikacja wag metodą wstecznej propagacji błędów



Błąd neuronu wyjściowego:

$$e_i = y_i(1-y_i)(d_i-y_i)$$

Nowe wagi neuronu wyjściowego:

$$w'_{ji} = w_{ji} + e_i \cdot h_j$$

Błąd neuronu w warstwie ukrytej:

$$\varepsilon_j = h_j(1-h_j)\sum e_i w_{ji}$$

Nowe wagi neuronu wyjściowego:
analogicznie jak dla neuronu wyjściowego

Inne modele sieci neuronowych

- Sieci rekurencyjne, np. sieci Hopfielda, sieci Jordana, sieci Elmana.
- Sieci pamięci asocjacyjnej, np. sieci typu BAM, sieci Hamminga.
- Sieci samoorganizujące się, np. mapy samorganizujące się (SOM).

Przykład sieci rekurencyjnej

- Sieć Elmana