Data Science
Metody Sztucznej Inteligencji

Paweł Wawrzyński

Uczenie maszynowe
Sztuczne sieci neuronowe

Plan na dziś

- Uczenie maszynowe
- Problem aproksymacji funkcji
- Sieci neuronowe

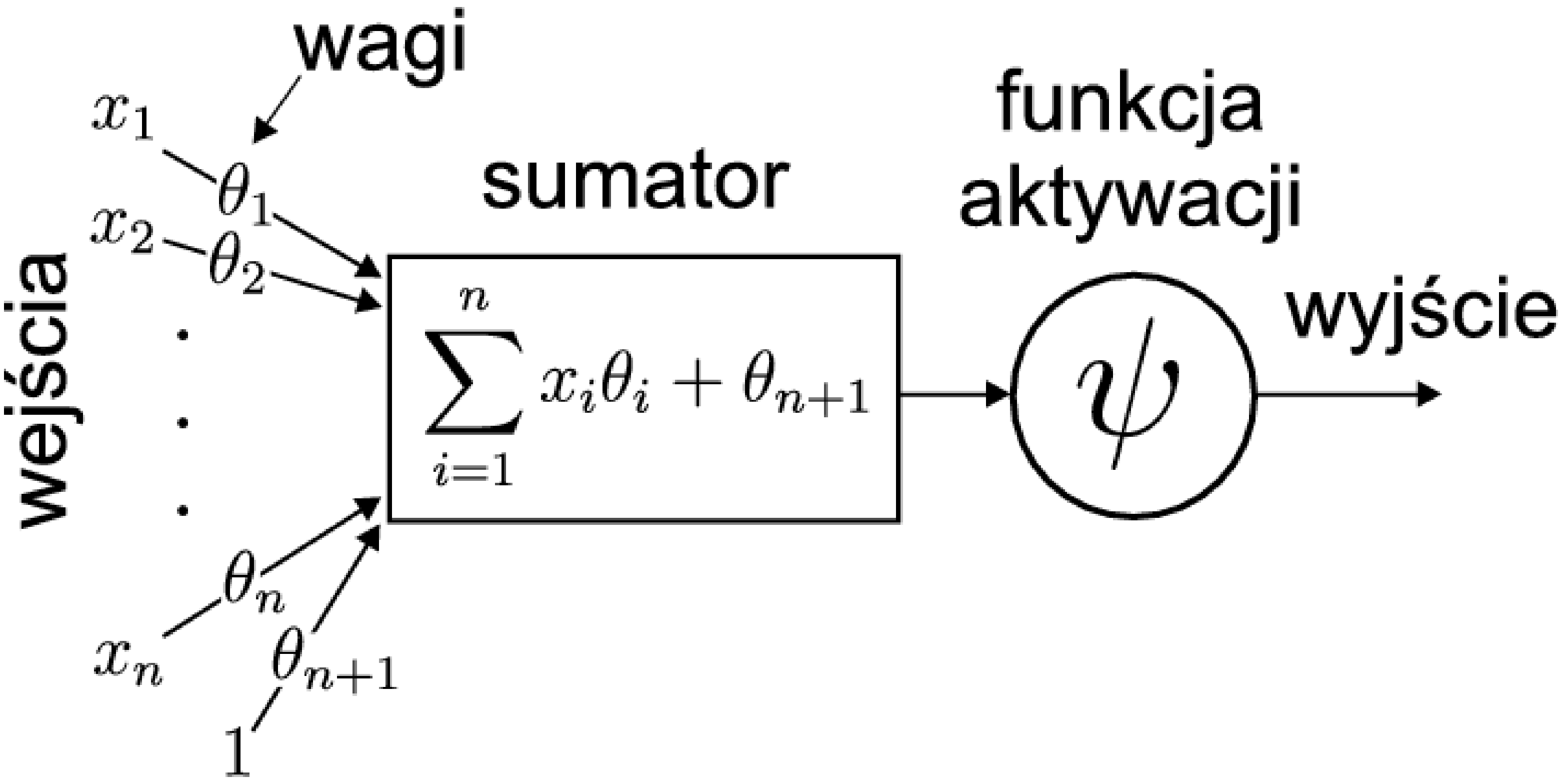
Naturalne sieci neuronowe

- Neuron
  - jądro komórkowe
  - dendryty
  - akson
  - zakończenia aksonu
  - połączenia synaptyczne
- Działanie
  - ładowanie się przez dendryty
  - strzelanie impulsami przez akson

Sztuczne sieci neuronowe

- Źródło inspiracji: naturalny mózg
- Rozmaite zastosowania:
  - aproksymacja funkcji
  - prognozowanie
  - klasyfikacja
  - pamięć asocjacyjna
- My zajmiemy się: *perceptronem dwuwarstwowym* ponieważ jest to dobry aproksymator nieliniowy

Aproksymacja neuronowa, model neuronu (1/2)



Aproksymacja neuronowa, model neuronu (2/2)

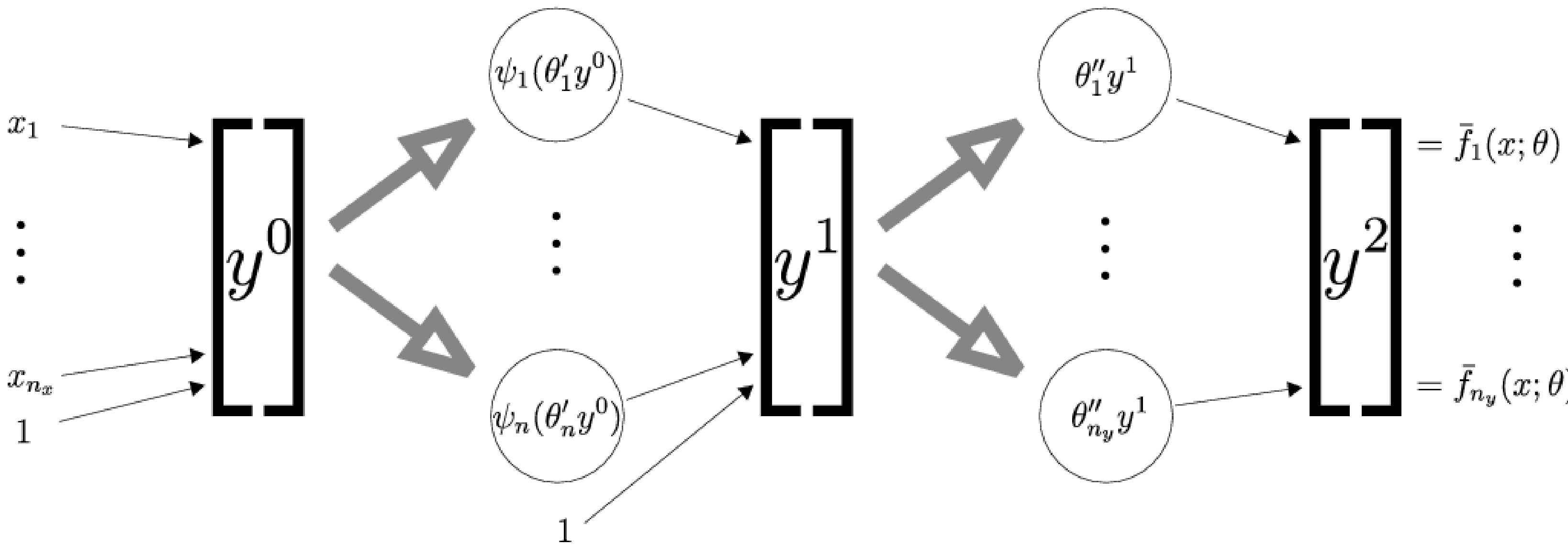
- Funkcja aktywacji jest ciągła i niemalejąca
- Neuron jest liniowy, jeśli $\psi(z)=z$
- Neuron jest sigmoidalny, jeśli f-cja aktywacji jest ciągła, różniczkowalna, rosnąca i ograniczona

$$\psi(z)=\arctan(z),\quad \psi(z)=\frac{\exp(z)}{1+\exp(z)},\ \text{itp.}$$

- Neuron jest ReLU, jeśli

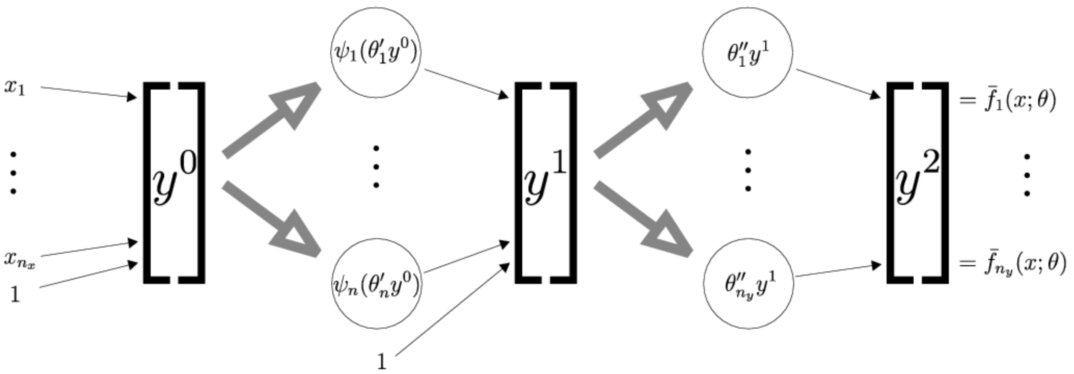
$$\psi(z)=\max\{0,z\},\quad \psi(z)=\min\{\max\{0,z\},M\}$$

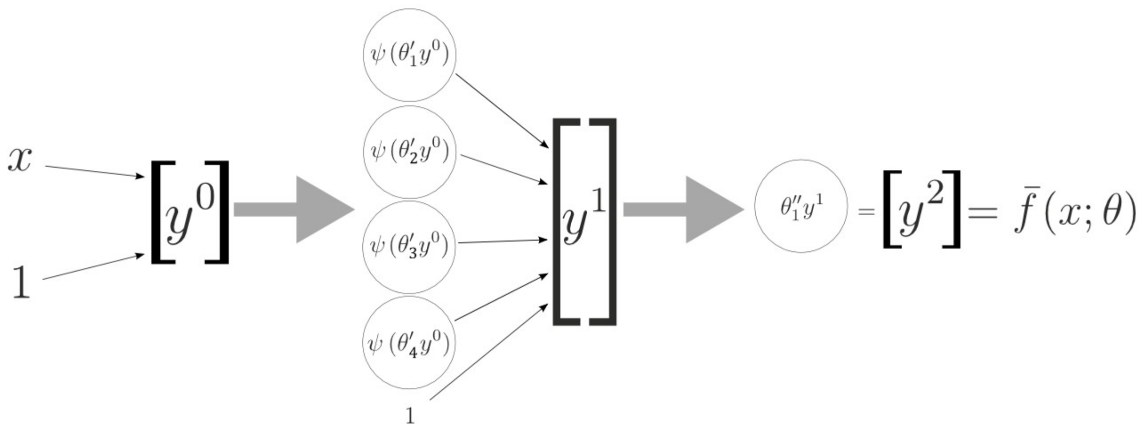
Aproksymacja neuronowa, perceptron 2-warstwowy

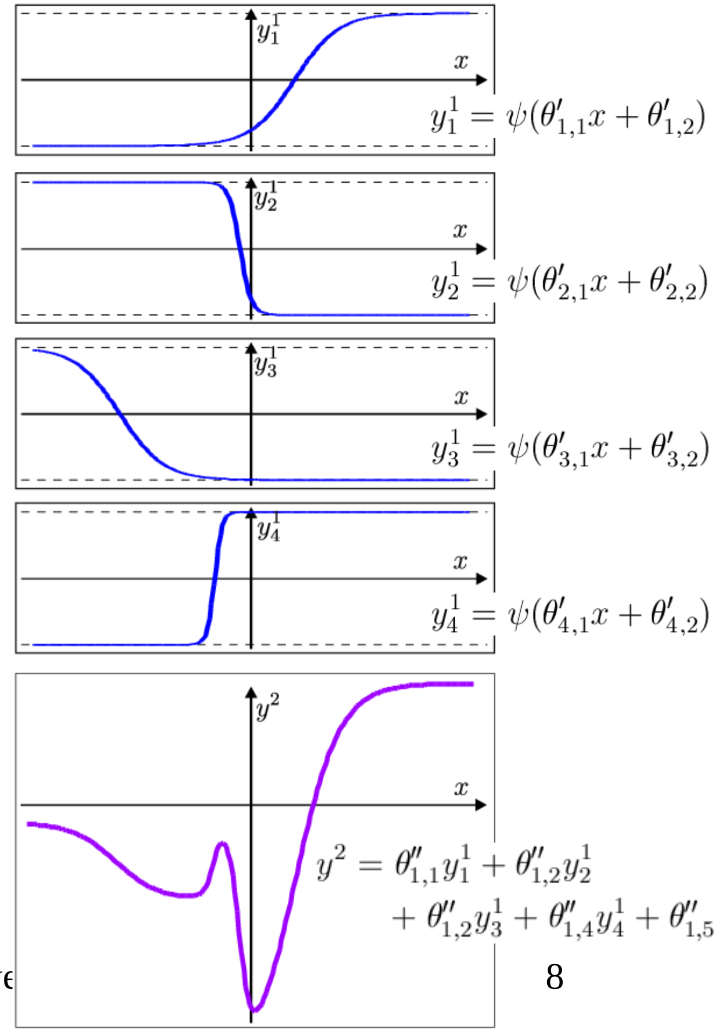


$$\bar{f}_k(x;\theta) = \sum_{j=1}^{n} \theta''_{k,j}\psi_j \left(\sum_{i=1}^{n_x} \theta'_{j,i} x_i + \theta'_{j,n_x+1} \right) + \theta''_{k,n+1}$$

Perceptron dwuwarstwowy, Przykład, funkcja R → R







Perceptron dwuwarstwowy, własność uniwersalnej aproksymacji

- Niech $\mathcal{X}$ będzie zbiorem ograniczonym i domkniętym
- f jest ciągła na $\mathcal{X}$
- dla każdego $\epsilon > 0$ istnieją n, θ, t.że:

$$\max_{x \in \mathcal{X}} ||f(x) - \bar{f}(x;\theta)|| \leq \epsilon$$

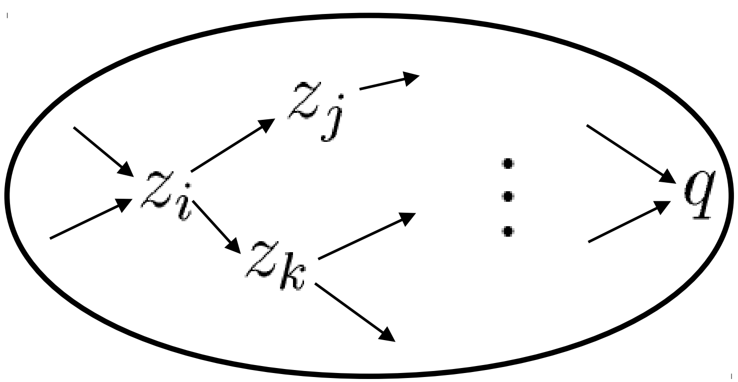
Gradient

- Funkcja straty $q : \Re^{n_y} \mapsto \Re$
- Typowo $q(y) = \|y - y^d\|^2$
- zawsze funkcja straty jest zdefiniowana przez przykład trenujący
- Interesuje nas $\dfrac{\mathrm{d}q(\bar{f}(x;\theta))}{\mathrm{d}\theta^T}$
- czyli wektor kolumnowy złożony z pochodnych

$$\frac{\mathrm{d}q(\bar{f}(x;\theta))}{\mathrm{d}\theta'_{j,i}} \text{ oraz } \frac{\mathrm{d}q(\bar{f}(x;\theta))}{\mathrm{d}\theta''_{k,j}}$$

Wsteczna propagacja gradientu

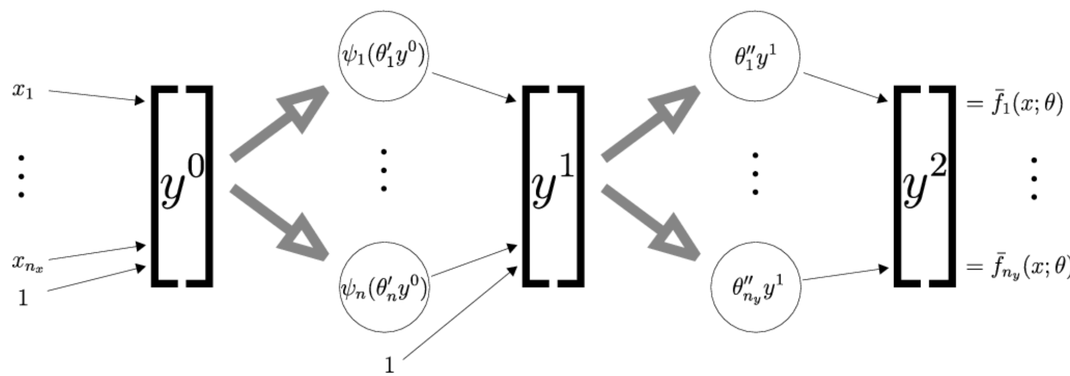
- Acykliczny graf działań obliczający q
- Zmienna w tym grafie oddziałuje na q poprzez zmienne, na które oddziałuje bezpośrednio



$$q = q\big(z_i, z_{k_1}(z_i), \ldots, z_{k_n}(z_i)\big)$$

$$\frac{\mathrm{d}q}{\mathrm{d}z_i} = \frac{\partial q}{\partial z_i} + \sum_{k: z_i \to z_k} \frac{\mathrm{d}q}{\mathrm{d}z_k} \frac{\partial z_k}{\partial z_i}$$

Pochodne po wagach warstwy wyjściowej



- $q = q(y_k^2(\theta''_{k,j}))$

$$\frac{\mathrm{d}q(\bar{f}(x;\theta))}{\mathrm{d}\theta''_{k,j}} = \frac{\partial q}{\partial y_k^2}\frac{\mathrm{d}y_k^2}{\mathrm{d}\theta''_{k,j}} = \frac{\partial q}{\partial y_k^2} y_j^1$$

- dla $q(y) = \|y - y^d\|^2$ mamy $\frac{\mathrm{d}q(\bar{f}(x;\theta))}{\mathrm{d}\theta''_{k,j}} = 2\,(y_k^2 - y_k^d)y_j^1$

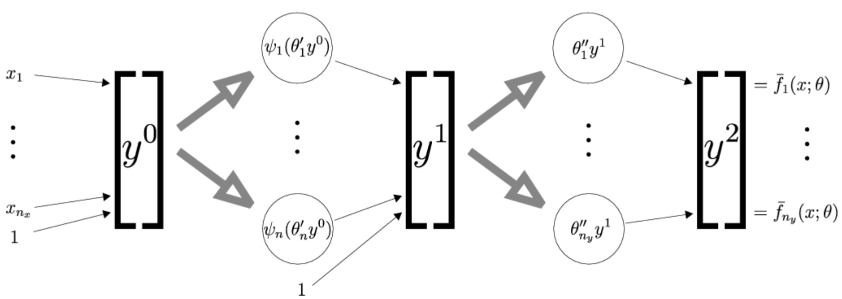
Pochodne po wagach warstwy ukrytej

- s_j - suma obliczana w *j*-tym neuronie warstwy ukrytej

$$\frac{\mathrm{d}q}{\mathrm{d}y_j^1} = \sum_k \frac{\mathrm{d}q}{\mathrm{d}y_k^2}\frac{\partial y_k^2}{\partial y_j^1} = \sum_k \frac{\partial q}{\partial y_k^2}\theta''_{k,j}$$

$$\frac{\mathrm{d}q}{\mathrm{d}s_j} = \frac{\mathrm{d}q}{\mathrm{d}y_j^1}\frac{\partial y_j^1}{\partial s_j} = \frac{\mathrm{d}q}{\mathrm{d}y_j^1}\psi'(s_j)$$

$$\frac{\mathrm{d}q}{\mathrm{d}\theta'_{j,i}} = \frac{\mathrm{d}q}{\mathrm{d}s_j}\frac{\partial s_j}{\partial \theta'_{j,i}} = \frac{\mathrm{d}q}{\mathrm{d}s_j}y_i^0$$



$$\begin{tikzcd} y_i^0 \arrow[r] & s_j \arrow[r] & y_j^1 \arrow[r] & y_k^2 \arrow[r] & q \\ \theta'_{j,i} \arrow[ur] & & \theta''_{k,j} \arrow[ur] & & \end{tikzcd}$$

Zagadnienie aproksymacji funkcji na zbiorze skończonym

- Dany jest skończony zbiór elementów

$$\langle x_i, y_i \rangle, \quad i \in \{1, \ldots, N\}$$

- Należy znaleźć wektor parametrów aproksymatora, który minimalizuje wskaźnik jakości

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \| y_i - \bar{f}(x_i; \theta) \|^2$$

- Działają algorytmy optymalizacji, np. metoda grdientu prostego

Zagadnienie aproksymacji funkcji na zbiorze nieskończonym

- Dany jest generator losowych par $\langle x, y \rangle \sim P_{x,y}$ który generuje kolejne próbki $\langle x_t, y_t \rangle, t = 1, 2, \ldots$
- Po *t*-tej próbce parametr aproksymatora jest aktualizowany (na jej podstawie) do wartości θ_t
- Ciąg parametrów $\theta_t, t = 1, 2, \ldots$ powinien zbiegać do minimum wskaźnika jakości

$$\begin{aligned} J(\theta) &= \mathcal{E}\,\|y - \bar{f}(x;\theta)\|^2 \\ &= \int\!\!\int \|y - \bar{f}(x;\theta)\|^2 P_{x,y}(x,y)\mathrm{d}y\mathrm{d}x \end{aligned}$$

„Reguła delta" i jej wykorzystanie

- Formuła uczenia się ma postać

$$\nabla J(\theta) = \mathcal{E}\left(\frac{\mathrm{d}}{\mathrm{d}\theta^T}\|\bar{f}(X;\theta) - Y\|^2\right)$$

- Działa metoda stochastycznego najszybszego spadku
- Formuła aktualizująca wagi sieci ma postać

$$\theta_{t+1} := \theta_t - \beta_t \frac{\mathrm{d}}{\mathrm{d}\theta_t^T}\|\bar{f}(x_t;\theta_t) - y_t\|^2$$

Uczenie – usprawnienia

- *Minibatch-e*:
  estymator gradientu
  = średnia z kilku kolejnych próbek
- Inercja: $\gamma \in (0,1)$
  zmiana wag
  = - β * gradient + γ * poprzednia_zmiana
- Normalizacja pochodnych:
  dzielenie przez ich odchylenia standardowe
- Algorytm uczenia sieci ADAM:
  połączenie ww.

Inicjalizacja parametrów sieci

- Hiperparametry sieci $\leftarrow$ k-krotna walidacja krzyżowa
- Liczba neuronów ukrytych: wystarczająca
- Wagi neuronów wyjściowych: zerowe
- Wagi neuronów ukrytych

$$\sim U\left(-1/\sqrt{dim(we)}, 1/\sqrt{dim(we)}\right)$$

- Skalowanie wejść i wyjść, aby odchylenie standardowe każdego ~1

Przeuczenie i co z nim robić (1/4)

- Przeuczenie:
  - Znacznie większy błąd na zbiorze testowym aniżeli treningowym

Przeuczenie i co z nim robić (2/4)

- Wczesne zatrzymanie uczenia
  - Dane → zbiór treningowy i walidacyjny
  - Uczymy na treningowym dopóki strata na walidacyjnym (a nie treningowym) spada

Przeuczenie i co z nim robić (3/4)

- Regularyzacja
  - Stosujemy
    $\bar{q}(\theta) = q(\bar{f}(x;\theta)) + \lambda \|\theta\|^2$
    lub
    $\bar{q}(\theta) = q(\bar{f}(x;\theta)) + \lambda \|\theta\|$

Przeuczenie i co z nim robić (4/4)

- Cel: elastyczność i odporność na braki
- Odrzucanie (*drop-out*)
  - Prawdopodobieństwo *p* ustalone dla warstwy
  - W trakcie uczenie dla danej próbki odrzucamy z p-stwem *p* dane wejście
  - W teście używane są wszystkie wejścia warstwy, ale wagi są przemnożone przez 1-*p*

Perceptron dwuwarstwowy jako klasyfikator

- n_y – liczba klas
- Dla danego wejścia z *i*-tej klasy
  żądane wyjście = [0..010..0]
  1 na *i*-tym miejscu
- Kiedy nauczona sieć produkuje wyjście, patrzymy który jego element jest największy, jego indeks wskazuje klasę

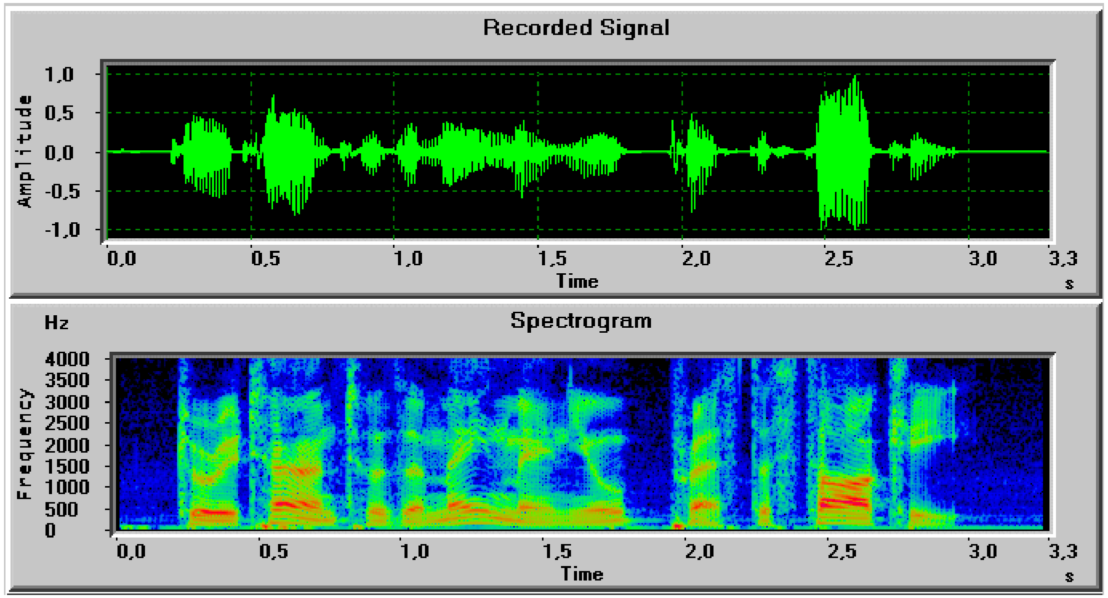
Perceptron wielowarstwowy

- Proste rozszerzenie perceptronu dwuwarstwowego:
  – wiele warstw
  – wszystkie, poza ostatnią zawierają neurony sigmoidalne
  – ostatnia warstwa zawiera neurony liniowe
- Możliwości aproksymacyjne takie jak perceptronu dwuwarstwowego, o ile warstwy są dość „szerokie”
- Łatwiej reprezentuje zależności obejmujące regularności wysokopoziomowe

Inne architektury sieci

- Sieci głębokie: liczba warstw >= 3
- Sieci bardzo głębokie: liczba warstw >= 10
- Sieci konwolucyjne: state-of-the-art w
  - Rozpoznawaniu obrazu
  - Rozpoznawaniu mowy
- Autoenkodery
- Generacyjne sieci przeciwstawne
  Generative Adversarial Networks - *GAN*

Na marginesie: dźwięk jako obraz



Szybka Transformata Fouriera (FFT):

dźwięk jako funkcja czasu

→ natężenie składników częstotliwościowych

Sieci konwolucyjne

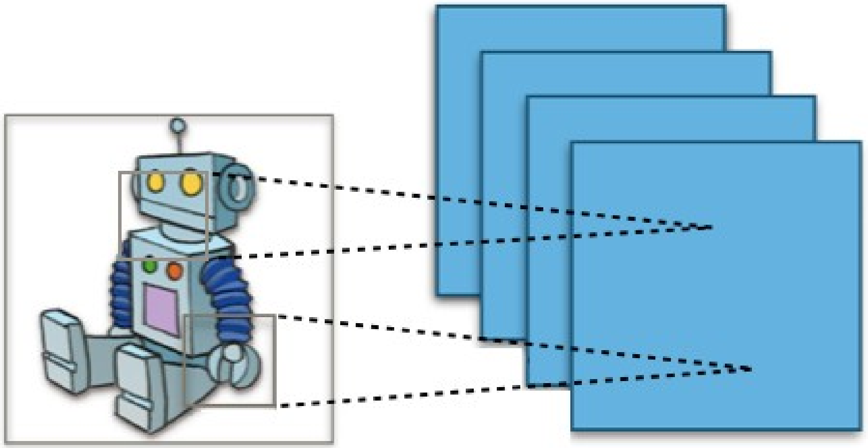
- Głębokie sieci złożone z 4 rodzajów warstw
  – Splotowe
  – ReLU
  – Łączące
  – Każdy-z-każdym

Warstwa splotowa (*convolution*)

- Wyjście neuronu

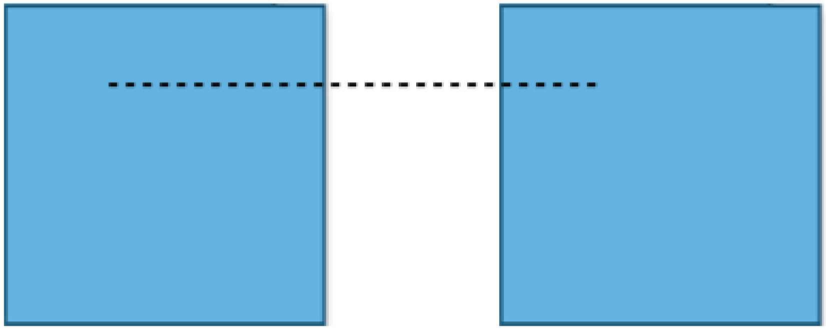
  = wejście^T * wagi
- Wejście

  ← pole recepcyjne

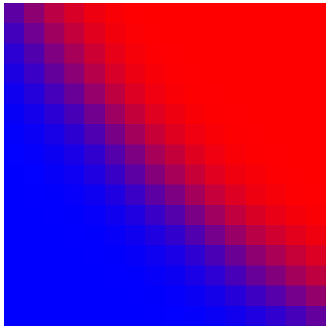


- Warstwa = kilka plastrów
- W plastrze
  - Neurony mają wspólne wagi
  - Pola recepcyjne neuronów pokrywają poprzednią warstwę

Warstwa ReLU (*Rectified Linear Unit*)

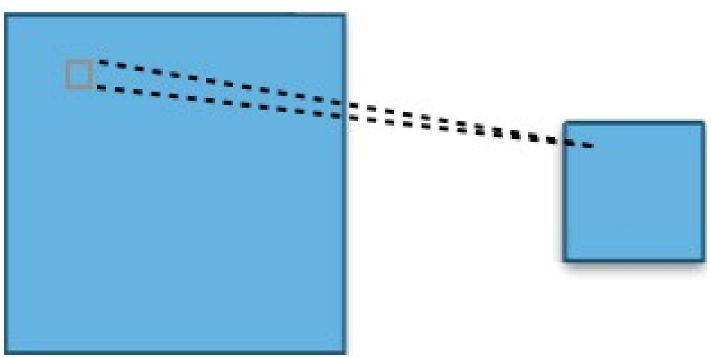


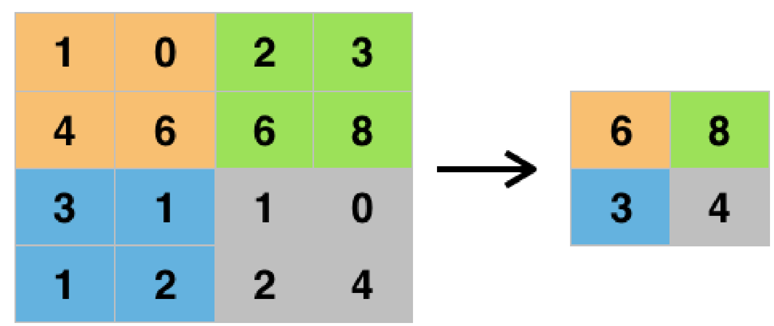
- Wyjście_neuronu = funkcja_aktywacji(wejście)
- Zwykle: f-cja-aktywacji(w) = max(0, w)
- Splot + ReLU
  = identyfikacja pewnej cechy
- Zastąpienie ReLU sigmoidalnymi
  → wolna nauka



Warstwa łącząca (*pooling*, *subsampling*)

- Wyjście neuronu
  = prosta funkcja pola recepcyjnego
- Zwykle ta funkcja to max
- Motywacja: musimy wiedzieć, że coś jest
  w okolicach, niekoniecznie gdzie dokładnie

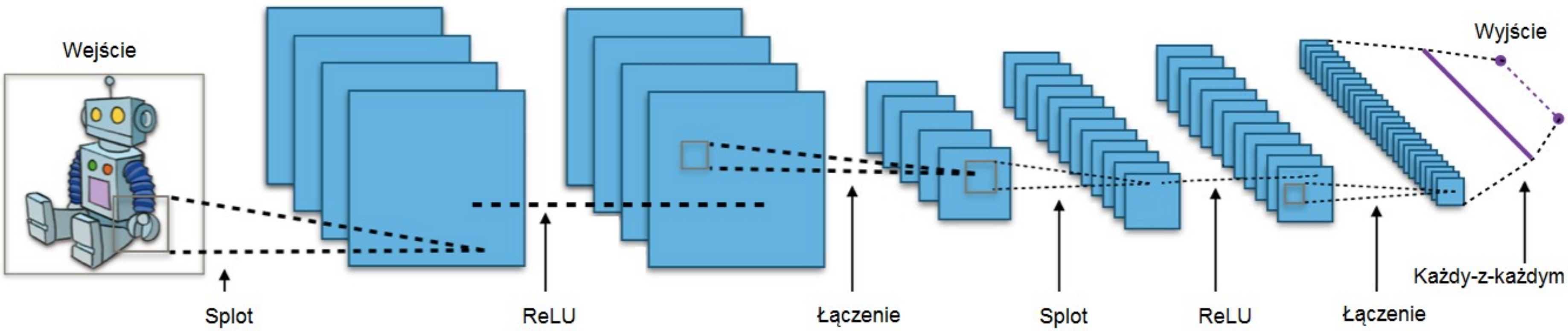




Warstwa każdy-z-każdym (*fully connected*)

- Jak w perceptronie wielowarstwowym
- Funkcje aktywacji neuronów
  - Sigmoidalne
  - ReLU
  - Liniowe

Sieć konwolucyjna – cała struktura



- Wyjście:
  - Klasyfikacja – tyle wyjść ile klas
  - Identyfikacja – jedno wyjście tak/nie
  - inne

Sieci rekurencyjne

- Połączenia cykliczne, z opóźnieniami
- Implementacja systemu dynamicznego
- Zastosowania:
  - prognozowanie
  - odtwarzanie stanu systemu częściowo obserwowanego

Sieci impulsowe

- Ang: *Spiking neural networks*
- Temat intensywnych badań
- Neurony stanowiące mniej-więcej wierne modele biologicznych odpowiedników
- Sieć działa w czasie rzeczywistym