

Metody Analizy Danych Sieci neuronowe

dr inż. Marcin Luckner
mluckner@mini.pw.edu.pl

Wersja 1.2
3 grudnia 2021

Mózg jako komputer

- Mózg można traktować jako komputer w którym obliczenia prowadzone są równolegle, a poszczególne procesory tworzą złożoną, nieliniową strukturę [Haykin, 2009].
- Dzięki organizacji elementarnych komórek mózgu (*neuronów*) w rozległe struktury, mózg potrafi dokonywać niektórych obliczeń szybciej i precyzyjniej niż dostępne komputery.
- Proces organizowania neuronów pozwala na naukę, czyli przyswajanie zasad rozwiązywania zagadnień i poznawanie nowych zagadnień, w stopniu niedostępnym dla komputerów.

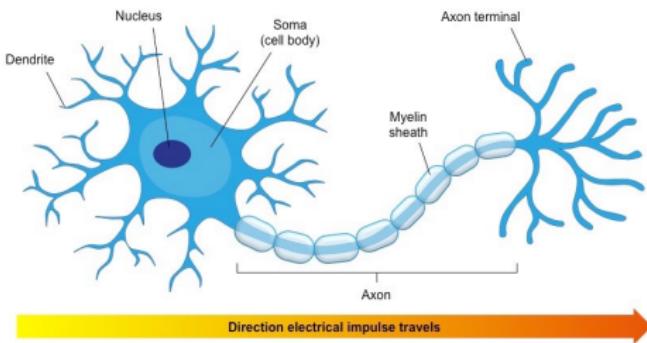
Sztuczne sieci neuronowe

- *Sztuczne sieci neuronowe*, określane skrótnie *sieciami neuronowymi* (ang. (artificial) neural networks, (A)NN) są inspirowane działaniem ludzkiego mózgu.
- Sieć neuronowa symuluje rozwiązywanie problemów w sposób zbliżony do tego w jaki robi to mózg.
- Mimo iż symuluje sposób nauki zachodzący w mózgu, ma nieporównanie mniejsze zdolności do samodzielnego uczenia.
- Sieci neuronowe buduje się raczej w celu rozwiązania konkretnego problemu niż w celu symulacji działania całego mózgu.

Sieć neuronowa - definicja

- Sieć neuronowa jest to procesor o masowej równoległości, zbudowany z prostych jednostek przetwarzających [Haykin, 2009].
 - Sieć posiada naturalną zdolność składowania nabytej wiedzy i jej wykorzystania
 - Sieć przypomina konstrukcją mózg w dwóch aspektach:
 - wiedza jest nabywana ze środowiska w procesie uczenia,
 - siła połączeń pomiędzy neuronami, określana mianem wag synaptycznych, odpowiada za wzmacnienie skojarzeń między różnymi aspektami przechowywanej wiedzy.
 - Procedura uczenia, sprowadza się do modyfikacji wag synaptycznych, aby zapewnić pożądaną reakcję sieci.

Schemat działania neuronu



Rysunek 1: Schemat neuronu biologicznego [Ask A Biologist, 2020]

- Rozgałęziający się dendryt zbiera informacje od innych neuronów.
 - Informacja jest kumulowana w somie i przetwarzana w jeden sygnał.
 - Akson wyrastający z sommy przekazuje sygnał innym neuronom.

Neuron biologiczny - szczegóły

- Neuron (komórka nerwowa) jest podstawową składową systemu nerwowego.
- Z ciała neuronu zwanego somą wyrasta gruby i rozwidlający się na końcu akson, którego odgałęzienia, nazywane kolateralami, kończą się synapsami.
- Akson komunikuje się z innymi neuronami poprzez liczne, cienkie i gęsto rozgałęzione dendryty.
- Neurony nie są fizycznie połączone, emisja sygnału odbywa się poprzez emisję neuroprzekaźników z kolbki synaptycznej.
- Neuroprzekaźniki wnikają w powierzchnię dendrytu łącząc się z różnymi receptorami, które pobudzają neuron.

Obliczenia mózgu

- Identyczny sygnał może być inaczej interpretowany w zależności od typu cząsteczki i receptora w synapsie.
- Sygnał generowany przez neuron jest przekazywany do wszystkich kolbek synaptycznych, ale wytworzenie nowego sygnału wymaga pobudzenia wielu synaps.
- Nie wszystkie są aktywowane jednocześnie, co umożliwia przeprowadzanie obliczeń przez mózg.



Rysunek 2: Obliczenia w mózgu
[Roš and Farinella, 2019]

Model neuronu - zasada działania

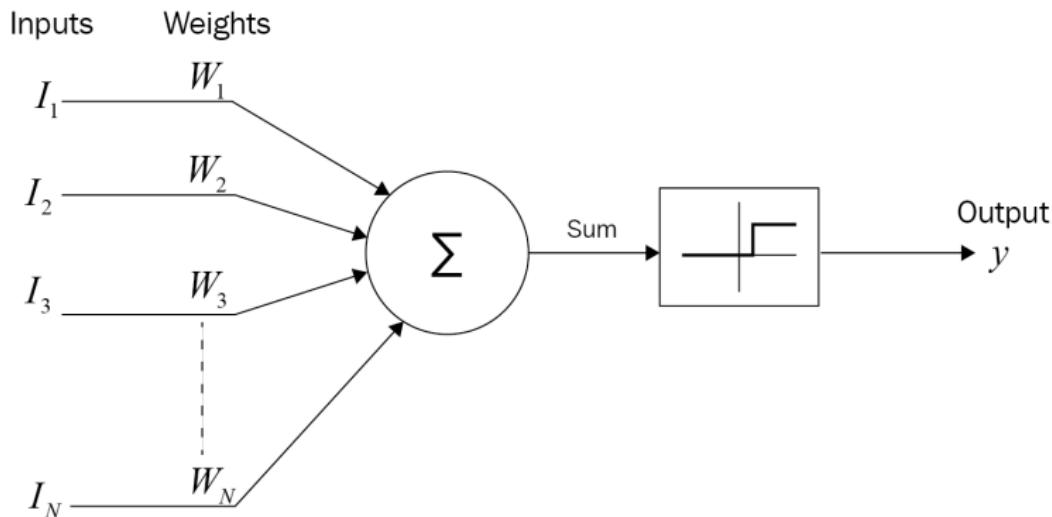
- Jeden z pierwszych modeli neuronu to model McCullocha-Pittsa, zaproponowany w roku 1943.
- Neuron przyjmuje N sygnałów wejściowych.
- Każdy z nich jest przetwarzany z odpowiednią wagą $w_i, i = 1, \dots, N$.
- Na ich podstawie wyznaczana jest wartość sygnału wyjściowego $y = \varphi(u)$ gdzie

$$u = w_0 + \sum_{i=1}^N x_i w_i$$

- W modelu McCullocha-Pittsa funkcja aktywacji neuronu przyjmuje postać skokową:

$$y(u) = \begin{cases} 1 & \text{dla } u \geq \theta \\ 0 & \text{dla } u < \theta \end{cases}$$

Model neuronu - schemat



Rysunek 3: Schemat sztucznego neuronu [Rothman, 2020]

Typowe funkcje aktywacji neuronu

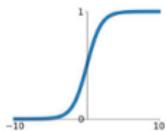
Najczęściej wykorzystywane są następujące funkcje aktywacji:

- funkcja *sigmoidalna unipolarna*, nazywana również *logistyczną*
$$\varphi(u) = \frac{1}{1+e^{-\beta u}}$$
- funkcja *sigmoidalna bipolarna*
$$\varphi(u) = \tanh(\beta u) = \frac{1-e^{-2\beta u}}{1+e^{-2\beta u}}$$
- funkcja liniowa
$$\varphi(u) = \alpha u + \beta$$
- funkcja typu signum
$$\varphi(u) = \begin{cases} 1 & \text{dla } u > 0 \\ -1 & \text{dla } u \leq 0 \end{cases}$$

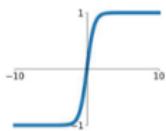
Wizualizacja funkcji aktywacji

Sigmoid

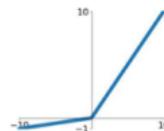
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

**tanh**

$$\tanh(x)$$

**ReLU**

$$\max(0, x)$$

**Leaky ReLU**
 $\max(0.1x, x)$ **Maxout**

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

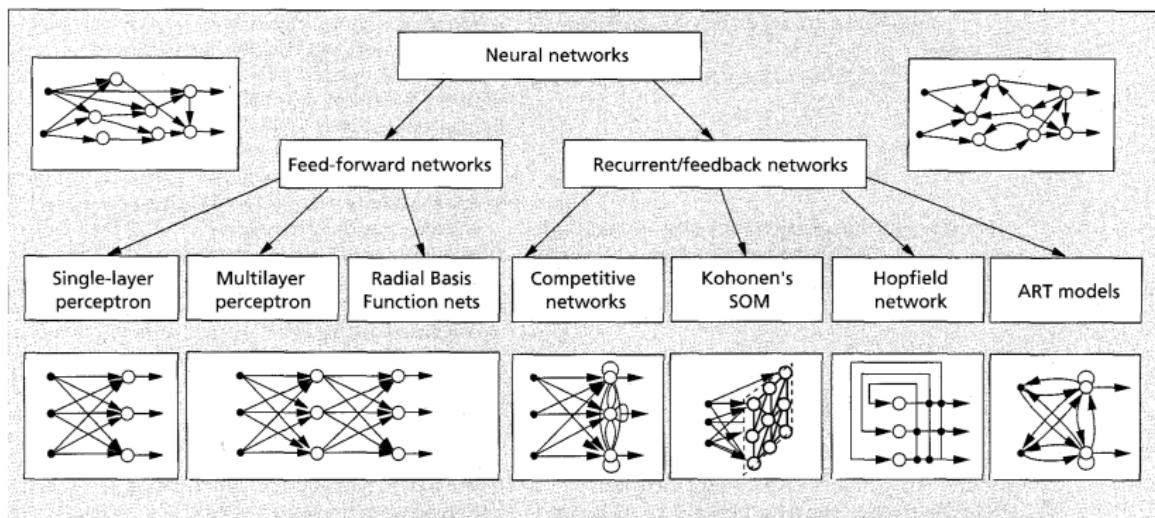


Rysunek 4: Wykres wybranych funkcji aktywacji [Jadon, 2018]

Kategorie sieci neuronowych

- Istnieje wiele rodzajów sieci, różniących się architekturą, doborem liczby neuronów, ich wag i połączeń.
- Do najważniejszych kategorii sieci neuronowych należą [Haykin, 2009, Osowski, 2006]:
 - sieci jednokierunkowe wielowarstwowe, określane terminem: wielowarstwowy perceptron (ang. Multilayer Perceptron, MLP),
 - sieci samoorganizujące się, w tym sieci działające na zasadzie współzawodnictwa - tzw. mapy samoorganizujące się (ang. Self-Organising Maps, SOM),
 - sieci neuronowe radialne (ang. Radial-Basis Function networks, RBF),
 - sieci rekurencyjne (ang. recurrent networks),
 - sieci neuronowe rozmyte (ang. fuzzy neural networks).

Przykłady architektury sieci neuronowych



Klasyfikacja sieci neuronowych [Jain et al., 1996].

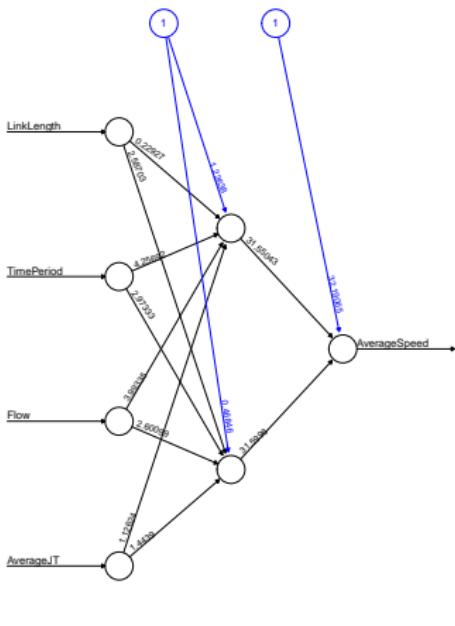
Struktura sieci neuronowej MLP

- Sieci neuronowe typu wielowarstwowy perceptron (ang. multilayer perceptron (MLP)) składają się z warstw neuronów.
- Pierwsza warstwa, warstwa wejściowa (ang. input layer), służy do odczytu danych
- Kolejne warstwy są warstwami ukrytymi (ang. hidden layer) nie przyjmują danych spoza sieci, ani nie generują sygnału wyjściowego
- Warstwa wyjściowa (ang. output layer) pozwala na odczyt wyników predykcji
- Każda z warstw jest połączona tylko i wyłącznie z kolejną warstwą w strukturze sieci

Neurony w sieci MLP

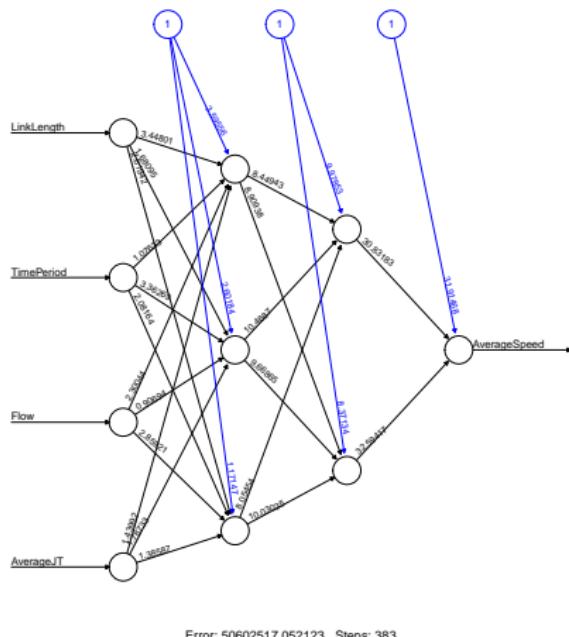
- Neurony zgrupowane są w poszczególnych warstwach
- Sygnały wychodzące z neuronów warstwy o indeksie n są przekazywane wyłącznie do neuronów warstwy $n + 1$.
- Każde z połączeń pomiędzy neuronami z kolejnych warstw ma przypisaną własną wagę.
- Ponieważ typowe algorytmy uczenia sieci neuronowych typu MLP bazują na metodach gradientowych, neurony stosują różniczkowalne funkcje aktywacji, w szczególności funkcje sigmoidalne.

Konstrukcja sieci typu MLP



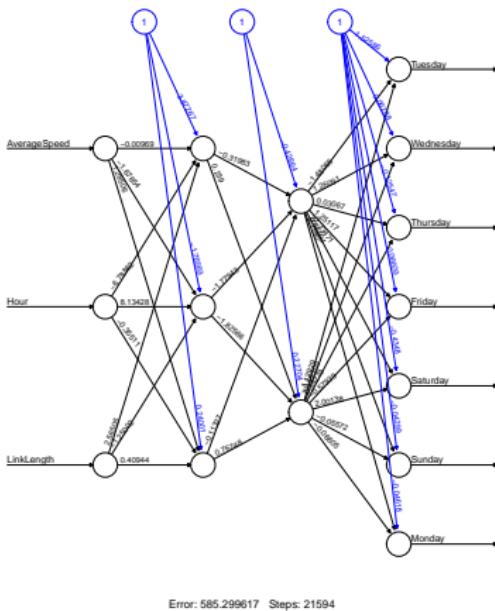
Rysunek 5: Przykład sieci neuronowej dla zagadnienia regresji

Konstrukcja sieci typu MLP II



Rysunek 6: Przykład sieci z dwiema warstwami ukrytymi

Konstrukcja sieci typu MLP III



Rysunek 7: Przykład sieci neuronowej dla zagadnienia klasyfikacji (dwie warstwy ukryte)

Proces budowy sieci typu MLP

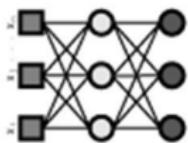
- Budowa sieci MLP wymaga od konstruktora:
 1. doboru liczby warstw,
 2. doboru liczby neuronów w poszczególnych warstwach.
 3. wyboru funkcji aktywacji neuronów,
 4. doboru wag połączeń synaptycznych pomiędzy neuronami kolejnych warstw.

Dobór liczby warstw

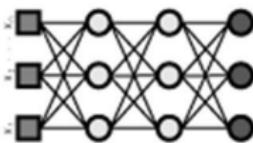
- Dobór liczby warstw, a następnie liczby neuronów na warstwach, jest kluczowy dla uzyskania wysokiej jakości predykcji.
- Sieć zawsze ma warstwę wejściową i wyjściową.
- Liczba warstw ukrytych jest zależona od rozpatrywanego zagadnienia i zazwyczaj waha się od jednej do trzech warstw.

Liczba warstw ukrytych, a możliwości sieci

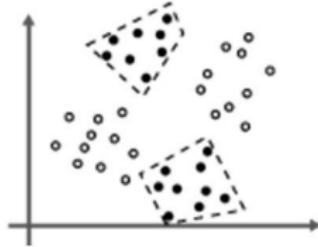
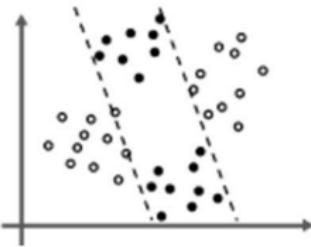
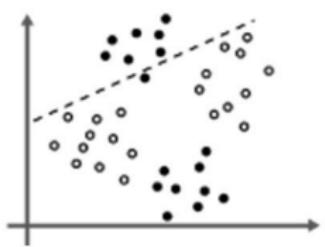
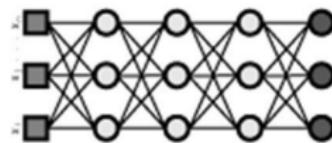
single layer



two layer



three layer



Dobór liczby warstw w zależności od zadania klasyfikacji
[Jain et al., 1996].

Dobór liczby neuronów na warstwach zewnętrznych

- Liczba węzłów wejściowych jest równa liczbie zmiennych niezależnych.
- Przygotowując zmienne wejściowe należy pominąć zmienne o stałych wartościach i zmienne nieistotne dla analizowanego zagadnienia.
- Nadmiarowe zmienne negatywnie wpływają na wynik procesu uczenia sieci, gdyż wymagają stworzenia dodatkowych połączeń i dobrania dla nich wag.
- Liczba neuronów w warstwie wyjściowej wynika z postawionego zagadnienia.

Dobór liczby neuronów na warstwach ukrytych

- Wzrost liczby neuronów w warstwie ukrytej zwiększa zdolność sieci do modelowania nieliniowych zależności w danych [Nisbet et al., 2009].
- Oznacza jednak ryzyko przeuczenia sieci i wydłuża czas uczenia.
- Liczbę neuronów należy dobierać z uwzględnieniem liczby rekordów uczących, liczba wag sieci nie powinna przekraczać liczby rekordów uczących.
 - w sieci o architekturze $5 \times 10 \times 3$ to $50 + 30 = 80$ wag do ustawienia,
 - w sieci $5 \times 30 \times 3$ wag będzie $150 + 90 = 240$ wag do ustawienia
 - próba nauczenia tej ostatniej sieci na 200 rekordach będzie prowadzić do anomalii i pozbawi ją zdolności generalizacji.
- W praktyce liczbę neuronów dobiera się eksperymentalnie stosując ograniczenia zakresu przeszukiwań.

Twierdzenie Kołmogorowa

Twierdzenie Kołmogorowa

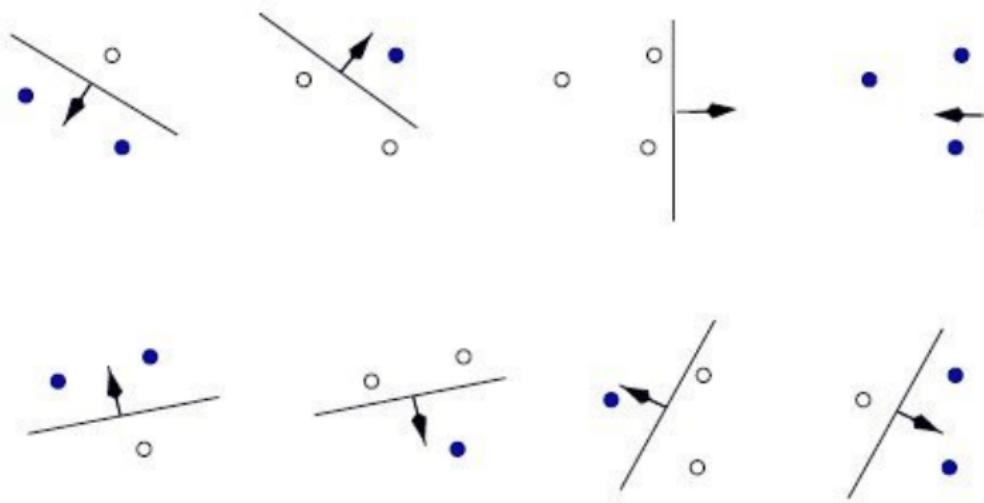
Skutecną aproksymację N -wymiarowego zbioru wejściowego x w M -wymiarowy zbiór wyjściowy d , można uzyskać przy użyciu sieci neuronowych już przy jednej warstwie ukrytej, zawierającej $2N + 1$ neuronów.

- Twierdzenie Kołmogorowa udowadnia, że aproksymacja jest możliwa przy $2N+1$ neuronach w warstwie ukrytej, ale nie mówi, że jest to architektura optymalna.

Miara Wapnika-Czerwonienkisa

- Miara VC_{dim} (Wapnika-Czerwonienkisa) mierzy zdolność uogólniania, to jest zdolność do generowania właściwych rozwiązań dla danych należących do zbioru T , na których sieć nigdy nie była trenowana.
- Miara wylicza liczebność największego zbioru S danych wzorców, dla których system może zrealizować wszystkie możliwe 2^n podziały zbioru S na 2 części za pomocą prostej.
- Jeżeli liczba dobranych obserwacji uczących, przy ustalonej liczbie VC_{dim} będzie zbyt mała to uzyskamy dobre dopasowanie sieci do próbek uczących, ale nie będzie miała ona zdolności uogólniania.
- Zbyt duża liczba obserwacji uczących spowoduje, że sieć nie będzie mogła dopasować wag do analizowanego zagadnienia już na poziomie procesu uczącego.

Możliwości dyskryminacyjne neuronu

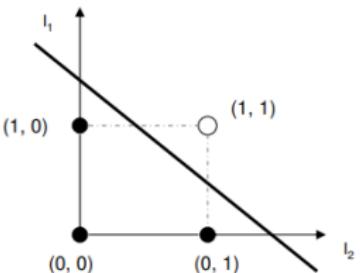


Rysunek 8: Liniowa segregacja trzech punktów

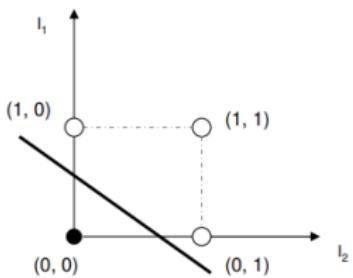
- Dla 3 punktów da się liniowo wyznaczyć wszystkie 2^3 podziałów, zatem dla jednego neuronu $VC_{dim} \geqslant 3$.

Problem XOR

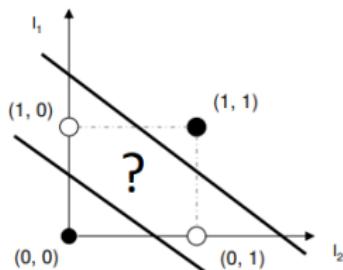
AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1



XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0



Rysunek 9: Próba implementacji bramek logicznych [Araújo, 2018]

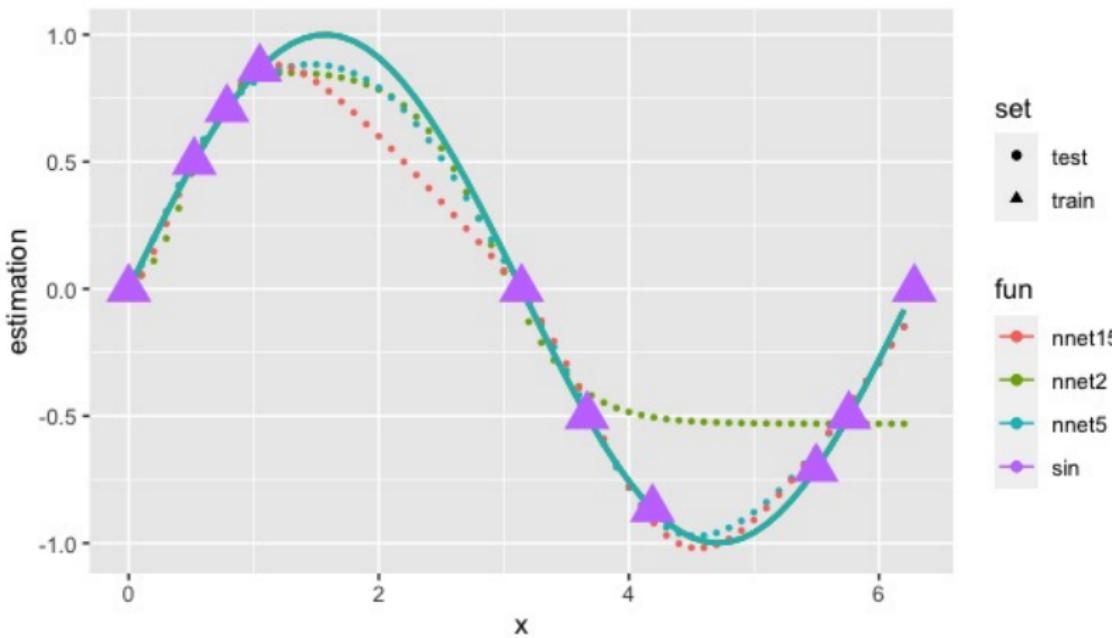
Miara VC_{dim} , a liczba neuronów

- Trudno dokładnie określić miarę VC_{dim} i często jest ona jedynie oszacowywana, ale można założyć, że jej wartość wzrasta przy wzrastającej liczbie wag.
- Można oszacować górny i dolny zakresu miary VC_{dim}

$$2 \left[\frac{K}{2} \right] N \leq VC_{dim} \leq 2N_w(1 + \log N_n)$$

- N – wymiar wektora wejściowego;
- K – liczba neuronów w warstwie ukrytej;
- N_w – całkowita liczba wag sieci;
- N_n – całkowita liczba neuronów w sieci.
- Zbiór uczący powinien zawierać $10 VC_{dim}-20 V_{dim}$ elementów.
- Jeżeli nie dysponujemy odpowiednim zbiorem należy ograniczać liczbę neuronów ukrytych.

Wpływ liczby neuronów na działanie sieci



Rysunek 10: Zjawisko niedouczenia i przeuczenia sieci neuronowej

Dobór funkcji aktywacji

- Proces aktywacji neuronu jest reprezentowany przez funkcję matematyczną.
- Chcąc ograniczyć przestrzeń poszukiwań funkcji można założyć za [Nisbet et al., 2009] stosowanie:
 - W zadaniach klasyfikacyjnych funkcji logistycznej.
 - W zadaniach regresji funkcji liniowej (nie zawsze możliwe do zastosowania).
- W warstwie wyjściowej, dla zadania klasyfikacji, często stosuje się funkcję *softmax*, aby wyjście sieci określało prawdopodobieństwo przynależności do danej klasy.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_k}} \wedge i = 1 \dots K.$$

Dobór wag sieci

- Dobór wag sieci neuronowej odbywa się w następującym procesie uczenia.
- Zbiór uczący musi składać się z par wektorów wartości zmiennych niezależnych i zmiennej zależnej:
 - indeks klasy (w przypadku zagadnienia klasyfikacji)
 - wartość zmiennej ciągłej (w przypadku zagadnień regresyjnych).
- W typowym podejściu do uczenia sieci, jakim jest metoda gradientowa następuje:
 1. Inicjacja wag połączeń losowymi wartościami.
 2. Cykl iteracji:
 - 2.1 Wyznaczana jest wartość funkcji reprezentowanej przez sieć dla rekordów danych uczących.
 - 2.2 Iteracyjnie korygowane są wagi połączeń w sieci na podstawie rozbieżności między wartością sygnałów wyjściowych sieci a oczekiwanyymi wartościami zmiennych zależnych.

Uczenie perceptronu

1. Zainicjuj wagi małymi wartościami losowymi.
2. Wylicz wyjście y neuronu dla wejścia \mathbf{x} .
3. Popraw wszystkie wagi zgodnie z regułą

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta [d - y] \mathbf{x}_i$$

gdzie η to współczynnik uczenia, a d pożądany wynik.

Propagacja wsteczna

- Metoda propagacji wstecznej (ang. back propagation, BP) została zaproponowana w 1986 roku przez Rumelharta i McClellanda.
- W tej gradientowej metodzie optymalizacji, proces uczenia dąży do minimalizacji błędu pomiędzy oczekiwana reakcją sieci (wartości sygnałów wyjściowych y) a faktycznymi wartościami zmiennych zależnych. W procesie tym dobierane są wagи.
- Metoda BP dała podstawę kolejnym metodom, bazującym na założeniu modyfikacji wag połączeń na podstawie wyznaczania gradientu funkcji błędu.

Zadanie optymalizacyjne

- Dla funkcji błędu $E(\mathbf{w})$ znajdź taki kierunek zmian \mathbf{p} , żeby minimalizować funkcję błędu.
- Korzystając z rozwinięcia w szereg Taylora

$$E(\mathbf{w} + \mathbf{p}) = E(\mathbf{w}) + \nabla(E(\mathbf{w}))^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{H}(\mathbf{w}) \mathbf{p} + \dots$$

- funkcja błędu: $E(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^N (o_k - y_k)^2$
- gradient: $\nabla(E(\mathbf{w})) = \left[\frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_m} \right]$
- hesjan: $\mathbf{H}(\mathbf{w}) = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1 \partial w_1} & \dots & \frac{\partial^2 E}{\partial w_1 \partial w_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_n \partial w_1} & \dots & \frac{\partial^2 E}{\partial w_n \partial w_n} \end{bmatrix}$

Modyfikacja wag

- Modyfikację wag przeprowadzamy według wzoru:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \mathbf{p}(t)$$

- Ponieważ zależy nam, aby błąd malał to z rozwinięcia szeregu Taylora do drugiego składnika

$E(\mathbf{w} + \mathbf{p}) = E(\mathbf{w}) + \nabla(E(\mathbf{w}))^T \mathbf{p}$ możemy zauważyć, że błąd maleje jeśli $E(\mathbf{w}(t+1)) < E(\mathbf{w}(t))$ stąd

$$\nabla(E(\mathbf{w}(t)))^T \mathbf{p}(t) < 0 \text{ co zachodzi gdy } \mathbf{p}(t) = -\nabla(E(\mathbf{w}(t)))$$

- Nowo wyliczone wagi przyjmują postać

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla(E(\mathbf{w}(t)))$$

- Jest to uczenie *regułą największego spadku*.

Uczenie sieci

1. Zainicjuj wagi małymi wartościami losowymi.
2. Losowo wybierz wejście \mathbf{x}_i i propaguj je w sieci.
3. Wylicz poprawkę δ_i^l dla warstwy wyjściowej.

$$\delta_i^L = f'(h_i^L) [d_i - y_i^L]$$

gdzie h_i^l jest i -tym wejściem z l -tej warstwy.

4. Wylicz poprawki dla poprzednich warstw $l = L - 1 \dots 1$

$$\delta_i^l = f'(h_i^l) \sum_j \mathbf{w}_{i,j}^{l+1} \delta_j^{l+1}$$

5. Popraw wagi wykorzystując $\Delta \mathbf{w}_{i,j}^l = 2\eta \delta_i^l x_j^l$.
6. Powtarzaj kroki 2-5 aż do osiągnięcia warunku stopu.

Ograniczenia propagacji wstecznej

- Mimo silnych teoretycznych podstaw i możliwości zastosowania w praktyce algorytm propagacji wstecznej ma kilka ograniczeń:
 - proces uczenia może utknąć w minimum lokalnym,
 - zbieżność algorytmu jest raczej wolna,
 - sieć może wykazywać tendencję do tworzenia nadmiarowych połączeń,
 - nowo dodawane przykłady mogą nadpisać zdolność do rozpoznawania wcześniejszych wzorców.
- Podjęto szereg prób naprawienia tych usterek poprzez modyfikację algorytmu [Li et al., 2012].

Metoda momentum

- Metoda momentum uzależnia zmianę wag w kroku następnym nie tylko od ich wartości w kroku obecnym, ale też w poprzednim.
- Zmiana wag wynosi

$$\Delta\mathbf{w}(t+1) = \alpha\Delta\mathbf{w}(t) - \eta(1-\alpha)\nabla(E(\mathbf{w}))$$

- $\Delta w(t+1)$, $\Delta w(t)$ wyliczana i poprzednia zmiana wag,
- α współczynnik momentum z przedziału $(0, 1)$ (zazwyczaj 0.9).
- $\nabla(E(\mathbf{w}))$ gradient sumy kwadratów błędów do wag.

Działanie metody momentum

- Jeżeli w kolejnych iteracjach kierunek modyfikacji wag był ten sam to zmiana zostaje wzmacniona.
- W przypadku odwrotnym, współczynnik momentum przyhamowuje proces zmian.
- Na płaskich odcinkach funkcji celu metoda momentum powoduje przyśpieszenie uczenia.
- W pobliżu minimum lokalnego wartość gradientu jest bliska零u i współczynnik momentum staje się dominujący.
 - Pozwala to na opuszczenie minimum lokalnego.

Wykorzystanie hesjanu

- W rozwinięciu szeregu Taylora trzeci składnik opiera się o drugie pochodne funkcji

$$E(\mathbf{w} + \mathbf{p}) = E(\mathbf{w}) + \nabla(E(\mathbf{w}))^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{H}(\mathbf{w}) \mathbf{p}$$

- Metoda najszybszego spadku ignoruje trzeci składnik, ale są metody, które go wykorzystują.
 - Algorytm zmiennej metryki.
 - Algorytm Levenberga-Marquada.
- Obie metody zapewniają szybką zbieżność, ale ze względu na duże wymogi mogą być stosowane tylko do małych sieci.

Porównanie metod

- Porównano czas zbieżności omawianych metod dla problemu XOR [Rutkowski, 2009].
- Czas mierzono w epokach, czyli iteracjach uczenia.
- Warunkiem stopu był spadek średniego błędu kwadratowego poniżej 0,012.
- Wyniki uśredniono z 10 testów.

Metoda	Epoki
Największego spadku	415
Momentum	250
Zmiennej metryki	8
Levenberga-Marquada	3

Zmienny współczynnik uczenia

- Kolejna modyfikacja zmienia współczynnik uczenia w zależności od zmiany błędu.
- Współczynnik jest wyliczany ze wzoru

$$\eta(t+1) = \begin{cases} k_{inc}\eta(t) & E(t+1) < E(t) \\ k_{dec}\eta(t) & E(t+1) > E(t) \\ \eta(t) & \text{w p.p.} \end{cases}$$

- $k_{inc} > 1$ współczynnik zwiększania uczenia (zazwyczaj 1.05),
- k_{dec} współczynnik redukcji uczenia z przedziału (0, 1) (zazwyczaj 0.7).
- $E(.)$ suma kwadratów błędów.

Ocena architektury sieci

- Ze względu na złożoność architektury sieci jej ocenę przeprowadza się empirycznie.
- Jako miernik można przyjąć dokładność modelu zależną od przyjętej miary błędu np. średni błąd kwadratowy w przypadku regresji lub skuteczność w przypadku klasyfikacji.
- Na wyniki uzyskane dla danej architektury wpływa szereg czynników
 - metoda uczenia,
 - parametry metody uczenia,
 - przyjęte kryterium stopu,
 - warunki początkowe uruchomienia metody.
- W celu oceny konkretnej architektury możemy wielokrotnie przeprowadzić uczenie dla różnych parametrów metody. Następnie przyjąć średnią lub medianę błędu jako miernik jej jakości.

Analiza wpływu neuronu

- Jedną z metod modyfikacji architektury sieci jest szacowanie wrażliwości funkcji celu względem wagi lub neuronu.
 - Wprowadzamy dodatkowy współczynnik dla każdej wagi α_j z przedziału $[0, 1]$.

$$y_i = f \left(\sum_j \mathbf{w}_{ij} \alpha_j y_j \right)$$

- obliczamy współczynnik wrażliwości funkcji celu względem $\alpha_j = 1$

$$\rho_j = \frac{\delta E}{\delta \alpha_j}$$

- odcinamy wagę ustalając $\alpha_j = 0$ jeżeli współczynnik jest mały.

Funkcja kary przy budowie sieci

- Możemy wprowadzić do funkcji celu składniki faworyzujące małe amplitudy wag.
- Zmusza to algorytm uczenia do redukcji wag, co pozwala zbudować mniejszą sieć, po usunięciu martwych neuronów.
- Przykładowa funkcja celu z elementem kary

$$E(W) = E^{(0)}(W) + \gamma \sum_{i,j} \mathbf{w}_{i,j}^2$$

- γ współczynnik kosztu kary

Uwagi do redukcji rozmiaru sieci

- Usuwanie neuronów z małymi wagami nie koniecznie musi być produktywne. Mała waga nie oznacza od razu, że neuron nie jest istotny.
- Po usunięciu neuronów należy douczyć sieć, co spowoduje poprawienie jej działania.

Wymagania co do danych uczących

- Zbiór uczący dla sieci neuronowej nie może zawierać brakujących wartości zmiennych.
- Może zawierać zmienne zależne, będzie to powodowało reprezentację zmiennych zależnych przez neurony w warstwach ukrytych.
- Powinien być poddany przeskalowaniu w celu uzyskania średniej wartości równej zero dla wszystkich zmiennych niezależnych.

Przygotowanie danych

- Wykonujemy standaryzację danych, zapewniając, że każda zmienna ma średnią zero i odchylenie standardowe 1.
- Po tej operacji mogą pojawić się wartości brakujące, czyli NA - dla zmiennych o stałej wartości w zbiorze. Należy je usunąć.
- Konieczne jest zastosowanie identycznej transformacji do danych testowych, jak do uczących. W innym wypadku taki sam rekord z obydwu zbiorów będzie inaczej reprezentowany na wejściu sieci.

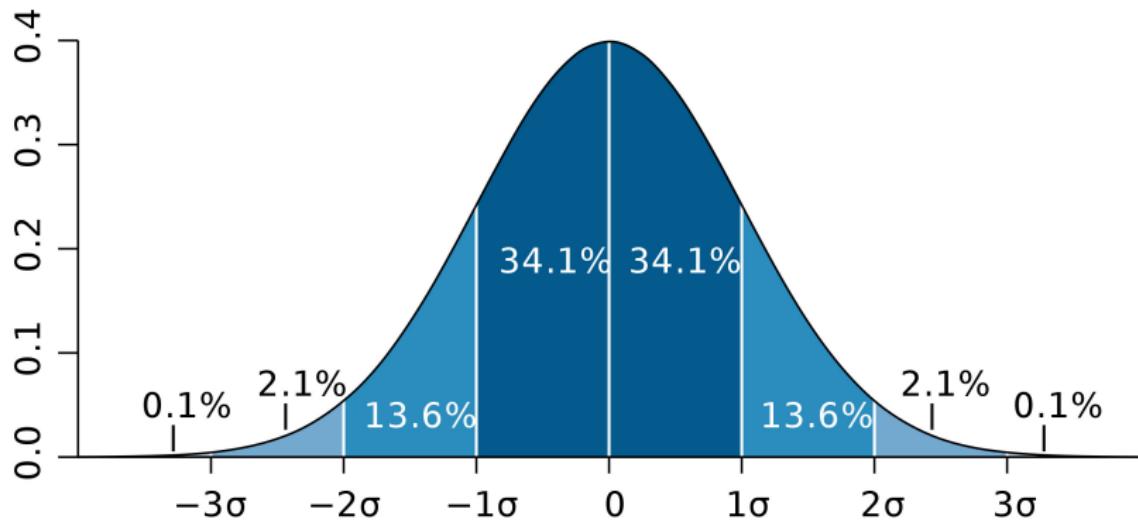
Standaryzacja

- Standaryzacja (ang. *Z-score standardization*) określa o różnicę między wartością, a wartością średnią w stosunku do odchylenia standardowego.

$$\bar{X} = \frac{X - \mu(X)}{\sigma(X)}$$

- Średnie wartości po standaryzacji przyjmą wartość zero, więc znak standaryzowanej zmiennej mówi czy jest ona większa, czy mniejsza od średniej.
- Wartości będą mieścić się w większości w przedziale $[-4, 4]$.

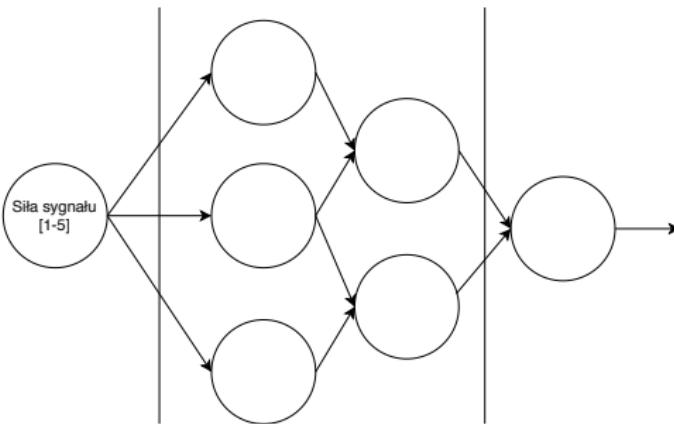
Rozkład danych po standaryzacji



Rysunek 11: Rozkład danych po standaryzacji [Wikipedia, 2019]

Prezentacja danych kategorycznych

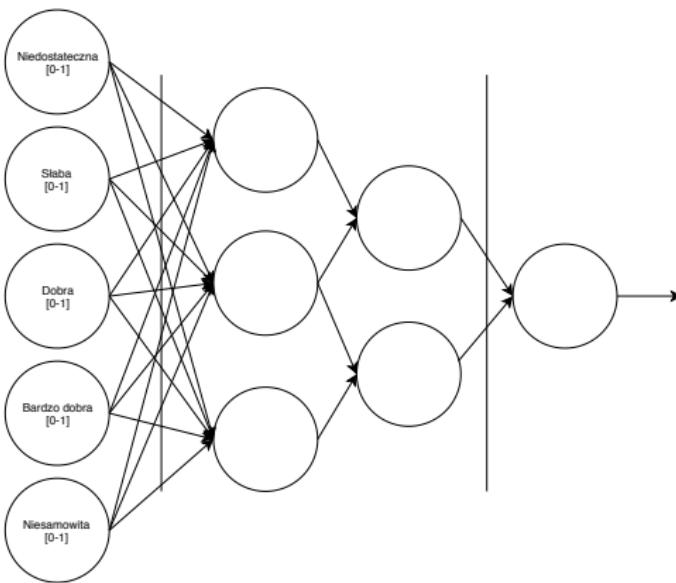
- Zmienną kategoryczną reprezentuje pojedyncza liczba całkowita odnoszącej się do indeksu kategorii.
- Sieć potraktuje ją jako zmienną uporządkowaną.
- Przykładowo, możemy zakodować kategorie opisujące siłę sygnału Wi-Fi
 - Niedostateczna, Słaba, Dobra, Bardzo dobra, Niesamowita



Rysunek 12: Wejście jedno-neuronowe dla kategorii danych

Przetworzenie danych kategorycznych

- Lepsze wyniki może dać stworzenie pięciu neuronów z których tylko jeden przyjmuje wartość 1, a pozostałe 0.
- Jest to kodowanie *one-hot*.



Rysunek 13: Wejście wielo-neuronowe dla kategorii danych

Uczenie sieci

- Celem uczenia sieci jest osiągnięcie możliwie wysokiej zdolności do generalizacji.
- W tym celu:
 - dobór parametrów architektury przeprowadzamy z wykorzystaniem zbioru walidacyjnego lub walidacji krzyżowej,
 - proces uczenia kończymy obserwując wyniki uzyskane na zbiorze walidacyjnym, a nie na zbiorze uczącym.
- Zbiór walidacyjny jest wydzielonym zbiorem nie biorącym udziału w procesie uczenia sieci.
- Wykonanie uczenia z wykorzystaniem z góry określonej liczby iteracji lub w oparciu o wyniki uzyskane na danych biorących udział w procesie uczenia skutkowałoby przeuczeniem sieci (ang. overfitting).

Wydzielenie zbioru walidacyjnego

- Jeżeli nie mamy dostępnego osobnego zbioru walidacyjnego możemy przeprowadzić walidację korzystając z jednej z poniższych metod:
 - walidacja krzyżowa stopnia k z możliwością wielokrotnych powtórzeń
 - walidacja krzyżowa z wydzieleniem jednego rekordu (ang. leave-one-out)
 - próba bootstrapowa

Powtarzanie procesu uczenia

- Ze względu na naturę gradientowych metod optymalizacji, proces uczenia warto wielokrotnie powtarzać.
- W kolejnych powtórzeniach procesu uczenia zmieniamy:
 - parametry metody uczenia,
 - punkt startowy jakim jest zestaw wartości losowo dobranych początkowych wag połączeń.
- Warto pamiętać, że kolejność prezentowania wzorców uczących także wpływa na proces uczenia i warto prezentować je w sposób losowy.

Dobór parametrów uczenia

- Współczynnik uczenia należy dobrać z przedziału $[10^{-3}, \dots, 10]$.
 - Niższa wartość spowalnia proces uczenia, ale zwiększa stabilność procesu.
 - Współczynnik uczenia można zmieniać w trakcie trwania procesu tak, aby kolejne kroki douczania nie zmieniały gwałtownie struktury sieci.
- Wagi początkowe nie powinny być zbyt wysokie, ani bliskie zeru.
 - Duże wagi początkowe wydłużają proces uczenia i mogą doprowadzić do utknięcia w minimum lokalnym.
 - Wagi bliskie zeru spadają często do zera i już się nie zmieniają tworząc martwe neurony.

Przykład klasyfikacji danych



- Zbiór *Optical Recognition of Handwritten Digits Data Set* składa się ze znormalizowanych obrazów ręcznie pisanych cyfr.
- Obrazy 32 na 32 pikseli zostały podzielone na bloki 4x4 w których zliczono czarne piksele.
- Dało to wektory 64 elementów przyjmujących wartości 0...16

Przygotowanie testów

- Dane wejściowe zostały poddane standaryzacji.
- Sieć składa się z jednej warstwy ukrytej, liczącej od 5 do 10 neuronów.
- Testowane są dwie wartości współczynnika uczenia 0.1 i 0.5.
- Razem daje to 12 różnych testów sieci.

Przykładowe wyniki

decay	size	Accuracy	Kappa	Accuracy SD	Kappa SD
0.1	5	0.903	0.892	0.0118	0.0131
0.1	6	0.923	0.914	0.011	0.0122
0.1	7	0.935	0.928	0.0124	0.0138
0.1	8	0.947	0.941	0.00676	0.00752
0.1	9	0.954	0.949	0.00695	0.00773
0.1	10	0.958	0.954	0.00714	0.00793
0.5	5	0.928	0.92	0.0104	0.0115
0.5	6	0.945	0.939	0.00535	0.00594
0.5	7	0.954	0.949	0.00757	0.00841
0.5	8	0.961	0.957	0.00453	0.00503
0.5	9	0.963	0.959	0.00521	0.00579
0.5	10	0.966	0.962	0.00518	0.00576

- Najwyższy poziom dokładności uzyskano dla $\text{decay}=0.5$ i $\text{size}=10$.
- Rosnąca tendencja skuteczności względem liczby neuronów sugeruje, że należy wykonać dodatkowe testy dla większej liczby neuronów.

Wyniki na zbiorze testowym

Reference

Prediction	0	1	2	3	4	5	6	7	8	9
0	174	0	0	1	0	0	0	0	0	0
1	0	173	1	3	2	0	1	0	10	2
2	0	1	171	4	0	1	0	0	0	1
3	0	0	3	165	0	2	0	0	1	0
4	1	0	0	0	176	0	1	1	0	1
5	1	0	0	2	0	176	0	1	3	1
6	1	3	0	0	0	0	178	0	0	0
7	0	0	1	2	0	0	0	173	0	2
8	0	3	1	5	2	0	1	0	154	2
9	0	2	0	1	1	3	0	4	6	171

Overall Statistics Accuracy : 0.9527

Dyskusja wyników

- Dokładność klasyfikacji na zbiorze testowym wynosi 0.9527
- Jest to wartość nieco niższa niż wynik na zbiorze uczącym, wynoszący 0.966.
- Gdyby wystąpiła znaczna różnica pomiędzy tymi wartościami, to może wskazywać to na wystąpienie przeuczenia lub odmienną charakterystykę danych testowych.

Zalety i wady sieci neuronowych typu MLP

- Sieci MLP mogą realizować bardzo złożone odwzorowania w wielowymiarowych przestrzeniach, w tym funkcje ciągłe i nieliniowe.
- Sieci MLP mogą być stosowane zarówno w zagadnieniach klasyfikacyjnych, jak i regresyjnych.
- Sieci MLP nie wymagają żadnych założeń na temat rozkładów danych, choć wskazana jest standaryzacja danych wejściowych. Nie stosują również żadnej weryfikacji hipotez ani pojęcia statystycznej istotności.
- Jednakże sieci odznaczają się bardzo złożonym odwzorowaniem. W praktyce wymusza to ich traktowanie jako tzw. czarnej skrzynki. Co utrudnia ich praktyczne wykorzystanie.
- W praktyce może być nawet niemożliwe określenie przeciw-domeny funkcji modelowanej przez sieć regresyjną.

Zasadność stosowania sieci MLP

- O zasadności stosowania sieci MLP przesądza ich skuteczność, czyli uzyskiwany poziom błędu.
- Jednakże, ze względu na nieczytelny proces podejmowania decyzji, zastosowanie innych algorytmów decyzyjnych może być bardziej praktyczne.
- Z tego powodu zalecane jest podchodzenie do problemu stosując różne metody uczenia maszynowego, w tym sieci neuronowe, i wybór sieci jako docelowe rozwiązanie tylko wtedy, gdy dają znacznie lepsze wyniki niż czytelniejsze metody.

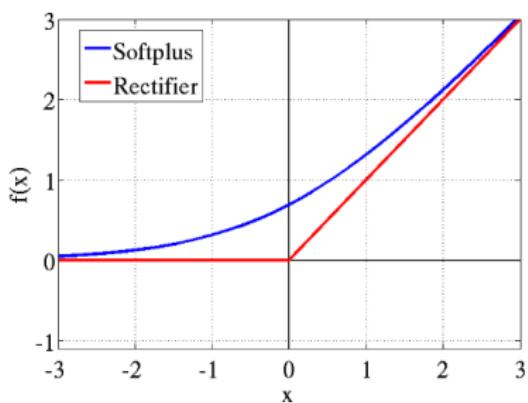
Sieci głębokie

- Uczenie głębokie przywróciło zainteresowanie sieciami neuronowymi, które wcześniej opadło po pojawienniu się nowych metod uczenia maszynowego takich jak SVM.
- Dzisiaj sieci wielowarstwowe (powyżej 3 warstw ukrytych) są stosowane do analizy obrazu i mowy.
- Jednakże ich powstanie wymagało przewyściżenia trudności technicznych i zmiany założeń stojących za uczeniem sieci.

Problem zanikającego gradientu

- W przypadku tworzenia perceptronu wielowarstwowego spotykamy się z problemem zanikającego gradientu (ang. *vanishing gradient problem*).
- W ramach algorytmu propagacji wstecznej zmieniamy wagi sieci stosując poprawki wynikające z wyliczonej pochodnej funkcji aktywacji.
- Jednakże ponieważ pochodne przyjmują wartości z przedziału $[0, 1]$, a wyliczana poprawka opiera się na ilorazie pochodnych z kolejnych warstw, poprawki na pierwszej warstwie są niewielkie.
- Powoduje to spadek efektywności uczenia i brak poprawy jakości sieci mimo dodawania kolejnych warstw.

ReLU



Rysunek 14: Funkcja ReLU
[Glorot et al., 2011]

- Rozwiązaniem jest stosowanie funkcji prostownika.
- Oparcie jednostki neuronu na tej funkcji ma silne biologiczne i matematyczne uzasadnienie [Hahnloser et al., 2000].
- Jednocześnie wykazano pozytywny wpływ jej stosowania na proces uczenia sieci głębokich [Glorot et al., 2011]
- Nową jednostkę nazywamy ReLU *Rectified Linear Unit*

ReLU - dyskusja

- Funkcja aktywacji ReLU to

$$f(x) = \max(0, x)$$

- Funkcja nie jest różniczkowalna w zerze, dlatego przyjmuje się w nim wartość 0 dla funkcji pochodnej

$$f'(x) = \begin{cases} 0, & \text{dla } x \leq 0 \\ 1, & \text{w p.p.} \end{cases}.$$

- W przypadku funkcji ReLU nie istnieje problem wygasającego gradientu, bo zawsze przekazuje to samo wzmacnienie przy aktywacji.
- Może za to pojawić się problem martwych neuronów.

Uczenie głębokie

- Innym podejściem do problemu było zaproponowanie uczenia każdej warstwy sieci osobno [Hinton et al., 2006].
 1. Szybki, nienadzorowany algorytm zachłanny ustala z osobna wagę poszczególnych warstw.
 2. Bazując na tych wynikach, algorytm nadzorowany globalnie dostraja wagę.
- Na zbiorze testowym MNIST (zawierającym pisane ręcznie cyfry) autorzy wykazali, że zaproponowane podejście daje lepsze wyniki niż MLP i SVM (błąd 1.25% na zbiorze testowym liczącym 10 tysięcy cyfr).
- Złożoność sieć odpowiadała wycinkowi kory mózgowej myszy o objętości 0.002 mm^3 .
- Zaproponowaną konstrukcję zaczęto nazywać siecią głęboką (*deep network*), a proces uczenia uczeniem głębokim (*deep learning*).

Idea uczenia głębokiego

- W uczeniu głębokim tworzymy zestaw sieci neuronowych, które mają reprezentować olbrzymi zbiór danych.
- Przetworzone dane są reprezentowane jako hierarchia konceptów z których każdy jest definiowany w odniesieniu do prostszych konceptów, a bardziej abstrakcyjne reprezentacje są wyliczane w odniesieniu do mniej abstrakcyjnych [Goodfellow et al., 2016].
- W oderwaniu od problemu badawczego, sieć głęboka najpierw tworzy w procesie nienadzorowanym reprezentację istotnych cech występujących w zbiorze danych, który musi być bardzo duży.
- Następnie można wykorzystać powstałe cechy do rozwiązania konkretnego problemu np. klasyfikacji w procesie nadzorowanym.

Reprezentacja konceptów



Layer 1: The computer identifies pixels of light and dark.



Layer 2: The computer learns to identify edges and simple shapes.



Layer 3: The computer learns to identify more complex shapes and objects.

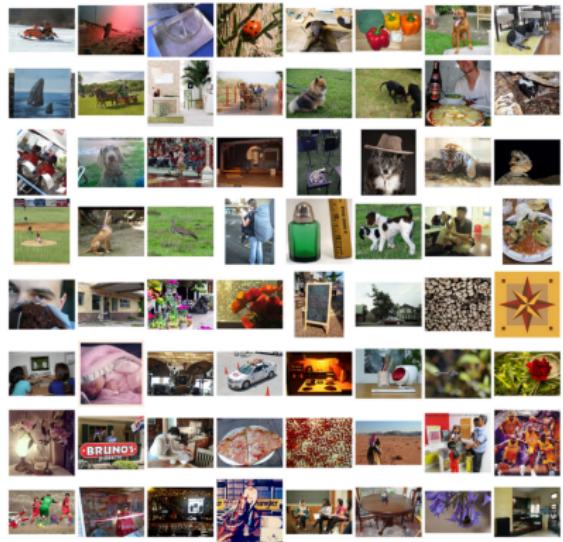


Layer 4: The computer learns which shapes and objects can be used to define a human face.

Rysunek 15: Tworzenie konceptów przy rozpoznawaniu twarzy
[Zaccone et al., 2017]

Warstwy w sieciach głębokich

- Każda warstwa rozwiązuje jeden problem, ma jedno zadanie.
- Dobranie większej liczby warstw pozwala na wyznaczanie odpowiedniej hierarchii.
- Zwiększanie liczby warstw pozwala też na zmniejszenie liczby neuronów w poszczególnych warstwach.
- Osobne uczenie warstw pozwala na zrównoleglenie obliczeń, więc przyrost warstw nie musi być bardzo kosztowny czasowo.

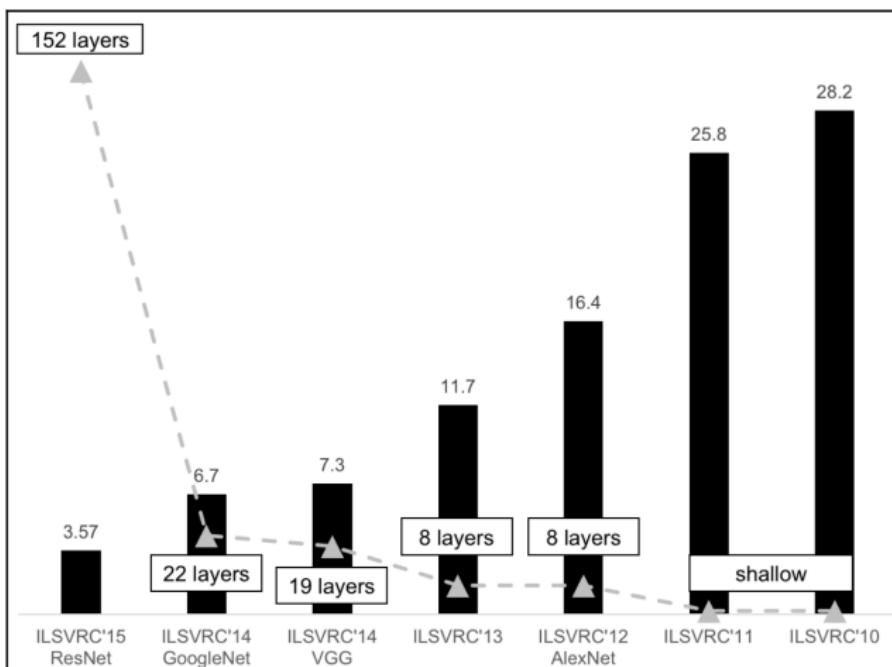


Rysunek 16: Przykładowe obrazy z ImageNet [Russakovsky et al., 2015]

ImageNet

- ImageNet jest zbiorem 14,197,122 skategoryzowanych obrazów.
- W latach 2010-2017 rocznie organizowano zawody polegające na klasyfikacji obrazów poprzez przypisanie ich do jednej z tysiąca klas.
- Zbiór stał się standardowym benchmarkiem do testowania algorytmów.
- Od 2013 konkurs wygrywały sieci głębokie.

Przykłady użycia - klasyfikacja



Rysunek 17: Wyniki klasyfikacji zdjęć [Shanmugamani, 2018]

Przykłady użycia - opisywanie zdjęć

A person riding a motorcycle on a dirt road.



Two dogs play in the grass.



A skateboarder does a trick on a ramp.



A dog is jumping to catch a frisbee.



A group of young people playing a game of frisbee.



Two hockey players are fighting over the puck.



A little girl in a pink hat is blowing bubbles.



A refrigerator filled with lots of food and drinks.



A herd of elephants walking across a dry grass field.



A close up of a cat laying on a couch.



A red motorcycle parked on the side of the road.



A yellow school bus parked in a parking lot.



Describes without errors

Describes with minor errors

Somewhat related to the image

Unrelated to the image

Rysunek 18: Opisywanie zdjęć [Vinyals et al., 2015]

Przykłady użycia - kopiowanie stylu



Rysunek 19: Sylizacja zdjęć [Vinyals et al., 2015]

Podsumowanie

- Uczenie głębokie zrewolucjonizowało obszar uczenia maszynowego i rzeczywiście doprowadziło do wyników kojarzonych potocznie ze sztuczną inteligencją.
- Wykład nie wyczerpuje zagadnienia, innym popularnym typem sieci są sieci rekurencyjne przetwarzające sekwencje o nieokreślonej długości.
- Sieci te stosowane są do automatycznej gry na giełdzie, analizy mowy, czy automatycznego tłumaczenia.

Bibliografia |

[Araújo, 2018] Araújo, L. (2018).

Solving xor with a single perceptron.

[Ask A Biologist, 2020] Ask A Biologist (2020).

Neurons and nerves.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011).

Deep Sparse Rectifier Neural Networks Xavier.

In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 315–323.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016).

Deep Learning.

MIT Press.

<http://www.deeplearningbook.org>.

Bibliografia II

[Hahnloser et al., 2000] Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. (2000).

Digital selection and analogue amplification coexist in a cortex- inspired silicon circuit.

Nature, 405(6789):947–951.

[Haykin, 2009] Haykin, S. S. (2009).

Neural networks and learning machines.

Pearson Education, Upper Saddle River, NJ, third edition.

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006).

A fast learning algorithm for deep belief nets.

Neural computation, 18(7):1527–1554.

[Jadon, 2018] Jadon, S. (2018).

Introduction to different activation functions for deep learning.

Bibliografia III

[Jain et al., 1996] Jain, A. K., Jianchang Mao, and Mohiuddin, K. M. (1996).

Artificial neural networks: a tutorial.

Computer, 29(3):31–44.

[Li et al., 2012] Li, J., Cheng, J.-h., Shi, J.-y., and Huang, F. (2012).

Brief Introduction of Back Propagation (BP) Neural Description of BP Algorithm in Mathematics.

Advances in Computer Science and Information Engineering, 2:553–558.

[Nisbet et al., 2009] Nisbet, R. A., Miner, G., and Iv, J. F. E. (2009).

Handbook of Statistical Analysis and Data Mining Applications.

Academic Press.

[Osowski, 2006] Osowski, S. (2006).

Sieci neuronowe do przetwarzania informacji.

Oficyna Wydawnicza Politechniki Warszawskiej.

Bibliografia IV

[Roś and Farinella, 2019] Roś, H. and Farinella, M. (2019).

Neurokomiks.

Wydawnictwo Marginesy.

[Rothman, 2020] Rothman, D. (2020).

The mcculloch-pitts neuron.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015).

ImageNet Large Scale Visual Recognition Challenge.

International Journal of Computer Vision (IJCV), 115(3):211–252.

[Rutkowski, 2009] Rutkowski, L. (2009).

Metody i techniki sztucznej inteligencji.

Wydawnictwo Naukowe PWN.

Bibliografia V

[Shanmugamani, 2018] Shanmugamani, R. (2018).

Deep Learning for Computer Vision.

Pact.

[Vinyals et al., 2015] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015).

Show and tell: A neural image caption generator.

Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 07-12-June-2015:3156–3164.

[Wikipedia, 2019] Wikipedia (2019).

Rozkład normalny.

[Zaccone et al., 2017] Zaccone, G., Karim, R., and Menshawy, A. (2017).

Deep Learning with TensorFlow.

Pact.