Implementacja Sieci Neuronowej

w Pythonie

Maciej Kołodziejczyk

Akademia Górniczo-Hutnicza

24 maja 2020

Co to są sieci neuronowe ?

Sieci neuronowe to systemy komputerowe z połączonymi węzłami, które działają podobnie jak neurony w ludzkim mózgu. Korzystając z algorytmów, mogą rozpoznawać ukryte wzorce i korelacje w surowych danych, grupować je i klasyfikować, a wraz z upływem czasu stale się uczyć i ulepszać.

Neurony w mózgu inspiracją dla systemu obliczeniowego.

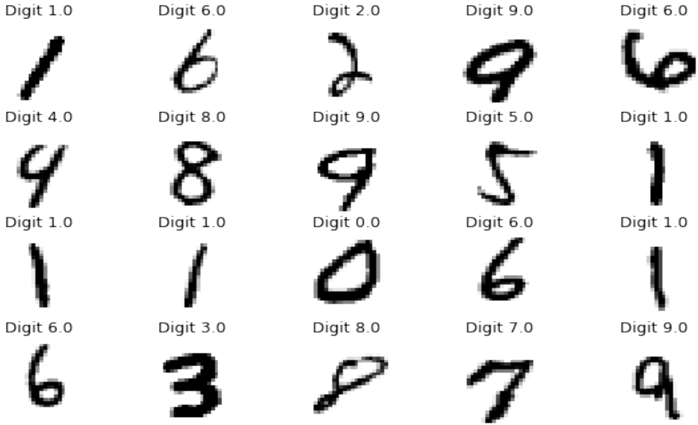
Pierwotnym celem podejścia opartego na sieci neuronowej było stworzenie systemu obliczeniowego, który mógłby rozwiązać problemy takie jak ludzki mózg. Jednak z biegiem czasu badacze przestawili się na wykorzystanie sieci neuronowych do spełnienia określonych zadań, co doprowadziło do odchyleń od podejścia ściśle biologicznego.

Dlaczego sieci neuronowe są ważne ?

- Rozpoznawanie i przetwarzanie mowy.
- Logistyka i transport.
- Przetwarzanie obrazu. Wyszukiwanie wzorca w zdjęciach: twarzy, zwierząt, przedmiotów.
- Prognozowanie obciążenia sieci elektrycznej, zapotrzebowania na energię.
- Identyfikacja substancji chemicznych.
- Automatyczne systemy sterowania.
- Diagnostyka medyczna i chorobowa.
- Prognozy finansowe: ceny aukcji, waluty, obligacje, kontrakty, upadłości.
- Wykrywanie oszustw w bankowosci.
- Kontrola jakości produkcji.

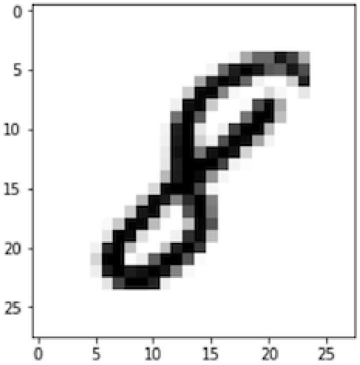
- Convolutional neural networks (CNNs)
- Recurrent neural networks (RNNs)
- **Feedforward neural networks**
- Autoencoder neural networks

Konkretny problem dla sieci neuronowej

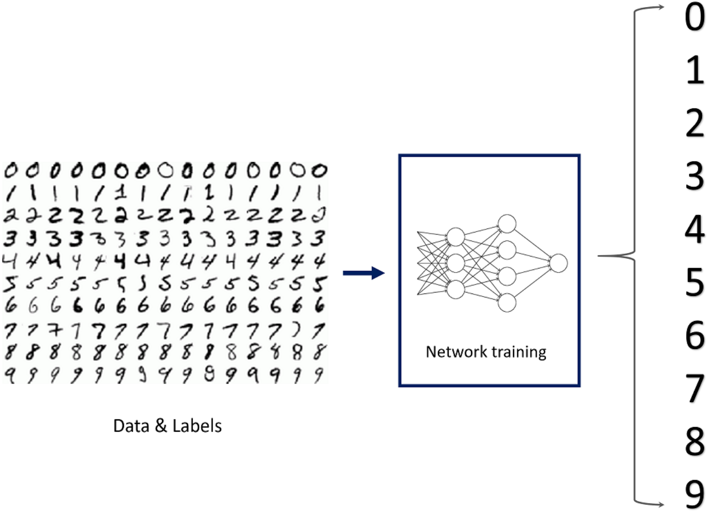


Dlaczego rozpoznawanie odręcznie pisanych cyfr jest trywialne dla ludzi, a dla maszyn nie ?

Nieodkryty wciąż geniusz ludzkiego mózgu nie ma z tym żadnych kłopotów
Jak rozbić duży problem na mniejsze problemy?
Czy nie dało by się szukać charakterystycznych kształtów dla poszczególnych cyfr, następnie przez odpowiednie warunki ustalić proces decyzyjny ?

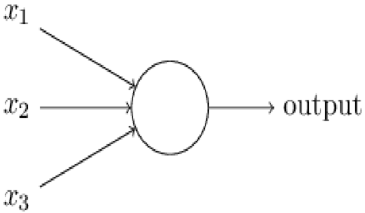


Jak sieci neuronowe rozwiązują problemy?



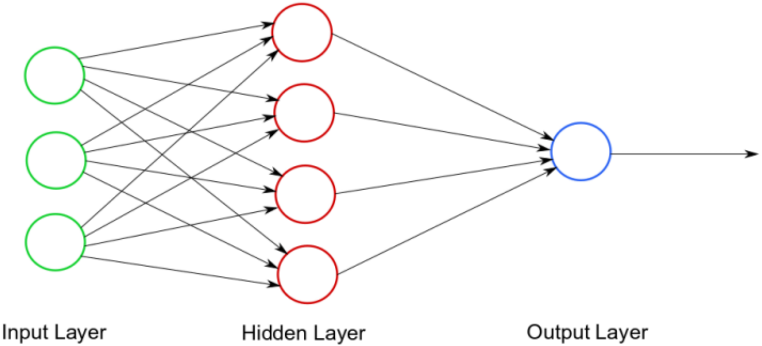
$$output = x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + b \tag{1}$$

- Lubie się bawić $[x_1 = 1]$
- powinnienem sie uczyć $[x_2 = 1]$
- Pogoda ma wpływ na zabawę $[x_3 = 0]$



Wagi (jak poszczególnie wejścia mają znaczenie):
$[w_1 = 10; w_2 = -15; w_3 = 20]$
Nie mamy wpływu na wejście neutronu, ale za pomocą wag i stałej b możemy zmieniać decyzję czyli to co wyrzuci neutron.

Schemat sieci neuronowej



Cechy neutronu:

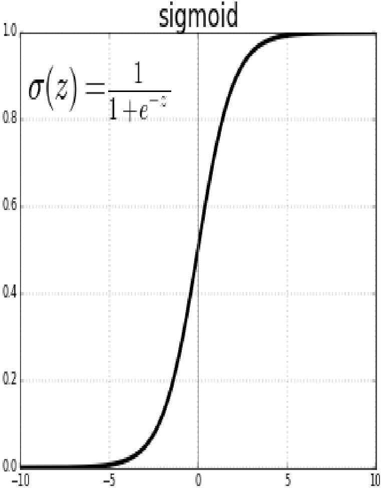
- wejścia neutronu z zakresu od 0 do 1
- dla każdego wejścia wagi (podlegają zmianom)
- wartość stała b

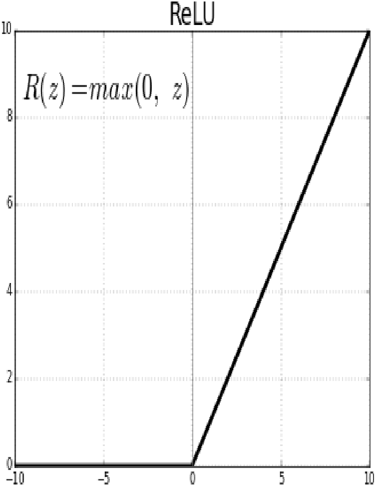
Jak obliczamy wyjście neutronu ?

$$y = \sigma \cdot (w \cdot x + b) \tag{2}$$

Wyjscie neutronu y nazywamy aktywacją neutronu σ jest funkcją aktywacyjną o nazwie sygmoid

Funkcje aktywacyjne - sygmoid, ReLu



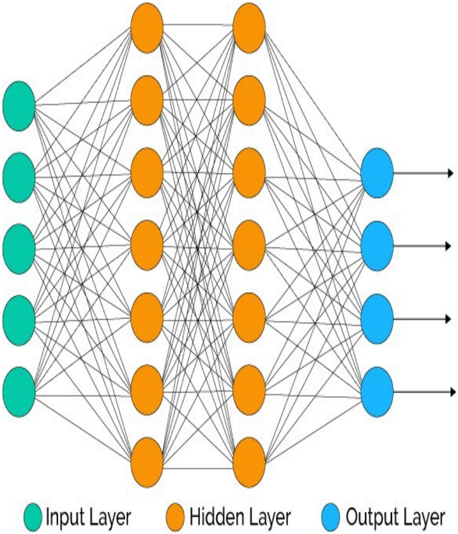


Gładki kształt σ zapewnia, że małe zmiany Δw_j w wagach i Δb stałej b spowodują niewielką zmianę $\Delta output$ na wyjściu neuronu. Powałując się na formuły rachunku różniczkowego mamy wzór który gwarantuje, że $\Delta output$ jest dobrze przybliżone przez:

$$\Delta output = \sum_j \frac{\partial output}{\partial w_j} \Delta w_j + \frac{\partial output}{\partial b} \Delta b \tag{3}$$

Wniosek płynący z równania (1) $\quad \Delta output$ jest liniową funkcją zmian Δw_j wag i zmian Δb stałej b.

Wektor danych wejsciowych, wektor wyjsć obliczony przez sieć, wektor prawidłowych wyników



$$\begin{bmatrix} 0.678 \\ 0.014 \\ 0.812 \\ 0.234 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

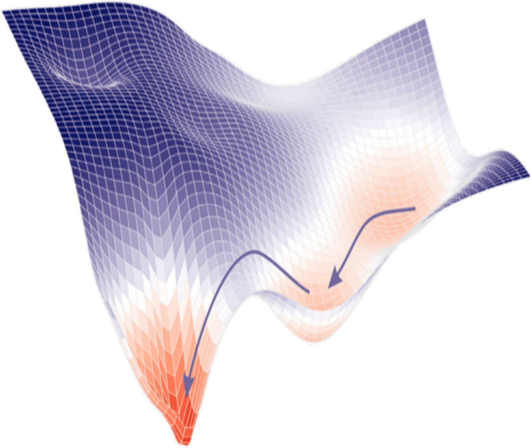
Potrzebujemy algorytmu, który pozwala znaleźć wagi w i stałe b, w taki sposób, aby wektor wyjściowy był zbliżony do $y(x)$ dla wszystkich danych wejściowych x. Definiujemy funkcję kosztu:

$$C(w,b) = \frac{1}{2n}\sum_x \left\| y(x) - a \right\|^2 \tag{4}$$

w i b to macierze wag i stałych, n to liczba danych wejściowych, a to wektor wyjściowy sieci. Sumujemy po wszystkich danych wejściowych.

- $C(w,b) > 0$
- $C(w,b) \approx 0$ kiedy $y(x) \approx a$
- algorytm, ktory znajdzie w i b, dla których $C(w,b) \approx 0$

Wyobraźmy sobie piłkę. Ustalmy, że rysunek obok jest w 3D. Piłkę kładziemy w losowym miejscu. Piłka zaczyna się toczyć w kierunku najstromiej opadającego zbocza. Ostatecznie znalazła się w minimum funkcji.



Gradient funkcji $C(w, b)$ zależnej od wielu zmiennych, oznaczamy przez ∇C to wektor, który wskazuje kierunek, w którym funkcja $C(w, b)$ rośnie najszybciej:

$$\nabla C = \left[\frac{\partial C}{\partial w}, \frac{\partial C}{\partial b} \right] \tag{5}$$

Innymi słowy: Gradient ∇C mówi jak zmieniać wszystkie wagi w i stałe b, tak aby optymalnie szybko zmnieniała się wartość funkcji kosztu $C(w, b)$.

Gradient Descent - update rule

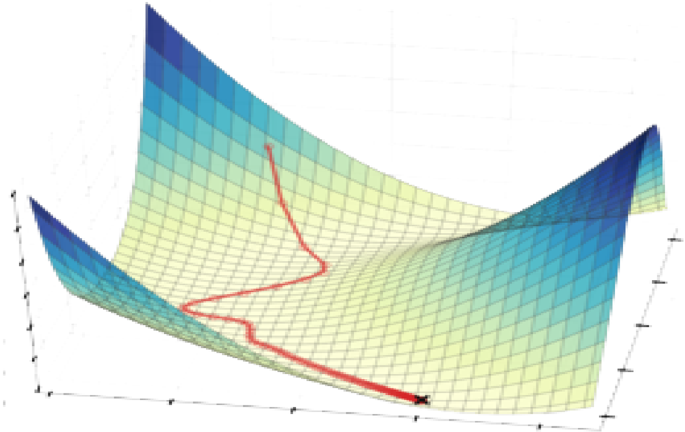
- Obliczamy gradient ∇C dla pojedyńczej jednostki treningowej.
- Następnie updatujemy wagi i stałe według wzoru:

$$w \rightarrow w' = w - \eta \nabla C \tag{6}$$

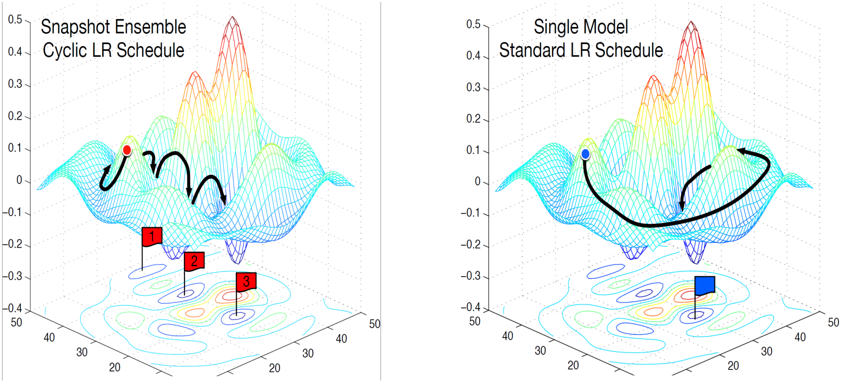
$$b \rightarrow b' = b - \eta \nabla C \tag{7}$$

- Postępujemy tak dalej dla kolejnych pojedyńczych danych wejściowych.

Jeśli będziemy to robić w kółko, będziemy obniżać koszt $C(w, b)$, dopóki nie osiągniemy globalnego minimum.



Stochastic gradient descent



Backpropagation to algorytm, który służy do wyznaczenia gradientu funkcji kosztu. Po wyznaczeniu gradientu dla zestawu danych treningowych lub niektórych danych wejściowych (SGD), dostajemy macierz liczb, które mówią o tym jakich zmian musimy dokonać w wagach i stałych.

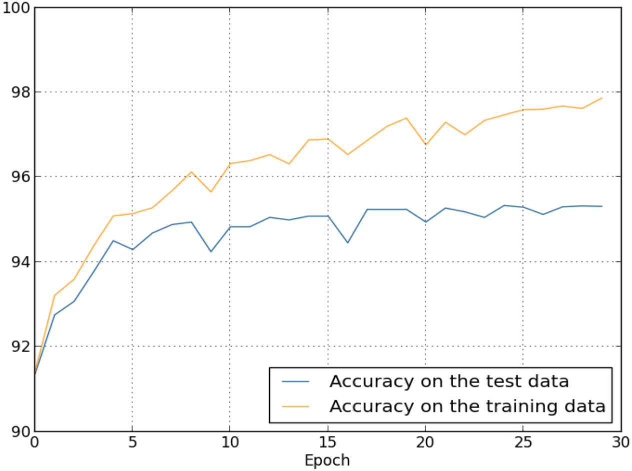
$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j$$

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$

Wykresy

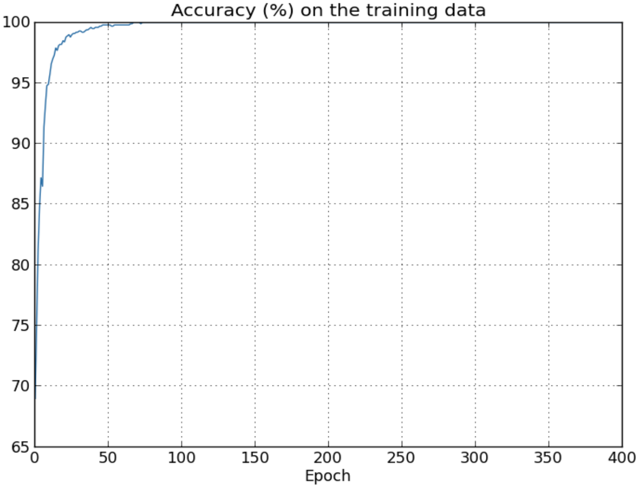


100
98
96
94
92
90
0
5
10
15
20
25
30
Epoch
Accuracy on the test data
Accuracy on the training data

Wykresy


Cost on the training data
0.010
0.009
0.008
0.007
0.006
0.005
0.004
200
250
300
350
400
Epoch

Wykresy



Accuracy (%) on the training data
100
95
90
85
80
75
70
65
0
50
100
150
200
250
300
350
400
Epoch

Źródła

- https://www.sas.com/en_us/insights/analytics/neural-networks.html
- https://pl.wikipedia.org
- http://neuralnetworksanddeeplearning.com/
- https://www.youtube.com/watch?v=aircAruvnKk&t=772s