POZNAN UNIVERSITY OF TECHNOLOGY ACADEMIC JOURNALS
No 96 Electrical Engineering 2018

DOI 10.21008/j.1897-0737.2018.96.0004

Stanisław PŁACZEK*, Aleksander PŁACZEK**

UCZENIE WIELOWARSTWOWYCH SZEROKICH SIECI NEURONOWYCH Z FUNKCJAMI AKTYWACJI TYPU RELU W ZADANIACH KLASYFIKACJI

W artykule przedstawiono obecnie nowy kierunek rozwoju Sztucznych Sieci Neuronowych w zadaniach aproksymacji i klasyfikacji. W praktyce stosowano sieci o jednej, maksimum dwóch warstwach ukrytych oraz funkcjach aktywacji typu sigmoid lub tanh. Funkcje te charakteryzują się małą zmiennością wartości dla większych wartości zmiennej wejściowej (występują obszary nasycenia) . Konsekwencją tego jest bardzo mała wartość pochodnej funkcji celu, która jest obliczana w algorytmie uczenia typu wstecznej propagacji błędu. W warstwach oddalonych od wyjścia sieci, algorytm operuje wartościami małymi, bliskimi zero, co powoduje, że algorytm jest bardzo wolno zbieżny. W sieciach o wielu warstwach ukrytych (10-15, a nawet więcej), stosuje się odcinkowe funkcje aktywacji pomimo ich formalno – matematycznych niedoskonałości. Stosując metody numeryczne w obliczeniu pochodnej, można ten problem rozwiązać, a tym samych poprawnie obliczyć pochodną funkcji aktywacji. Powyższe pozwala na obliczenie gradientu funkcji celu dla warstw głębokich uzyskując jednocześnie zadawalającą szybkość zbieżności.

SŁOWA KLUCZOWE: sieci neuronowe, algorytmy uczenia, uczenie głębokie, sieci szerokie.

1. STRUKTURA WIELOWARSTWOWEJ GŁĘBOKIEJ SIECI NEURONOWEJ

1.1. Krótka historia rozwoju sieci

Historia rozwoju sztucznych sieci neuronowych sięga lat pięćdziesiątych dwudziestego wieku. Pierwszy Perceptron, oprócz swoich niezaprzeczalnych walorów poznawczych, miał podstawową wadę: Nie potrafił poprawnie sklasyfikować nieliniowych zależności w danych wejściowych (np. dla XOR). To zatrzymało rozwój sieci na kilkanaście lat. Dopiero w latach osiemdziesiątych, do rozwiązania powyższego problemu zaproponowano sieci dwuwarstwowe

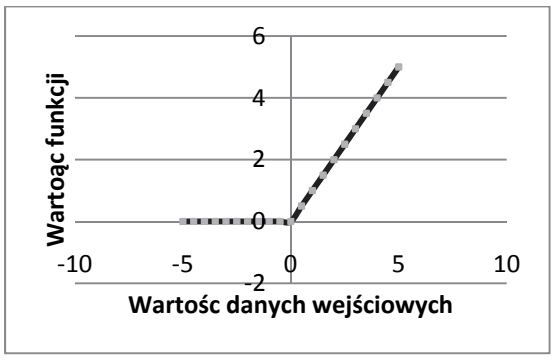
* Wyższa Szkoła Handlowa Wrocław
** Politechnika Śląska, Wydział Automatyki, Elektroniki i Informatyki, WASKO S.A. Gliwice

z jedną warstwą ukrytą. Powtórnie również odkryto algorytm wstecznej propagacji błędu. Te dwa czynniki spowodowały stosunkowo szybki i dynamiczny rozwój sieci, które znalazły zastosowanie w wielu dziedzinach. Szybko jednak okazało się, że do uczenia sieci neuronowych metodą nadzorowaną (z nauczycielem) lub bez nauczyciela potrzebne są dwa czynniki: duże zbiory danych uczących oraz szybkie procesory z dużą pamięcią operacyjną. Powyższe czynniki są bardzo istotne w zadaniach przetwarzania obrazów czarno - białych jak i kolorowych oraz rozpoznawania mowy. Obrazy wysokiej jakości (o dużej ilości pikseli np. 1000*1000) wymagają ponad 1 mln neuronów w warstwie wejściowej. Kolejne warstwy ukryte również posiadają odpowiednio tysiące i setki neuronów. Kompresja informacji realizowana w głąb sieci neuronowej, w celu uchwycenia, przetworzenia i ekstrakcji elementów cech jest realizowana w sieciach szerokich oraz głębokich. W literaturze spotyka się różne definicje oraz określenia pojęcia Głębokie Uczenie (Deep Learning). W niniejszym artykule przez Głębokie Uczenie rozumie się uczenie szerokiej sieci neuronowej, o wielu warstwach, zdefiniowanej rozmiarem danych wejściowych (wektora wejściowego X). Zastosowano nieliniową funkcję aktywacji typu f(u) = max(0,u), czyli „rectified linear unit”

1.2. Skorygowana funkcja aktywacji ReLu

Na obecnym etapie rozwoju głębokich sieci neuronowych często stosuje się liniowo odcinkowe funkcje aktywacji zdefiniowane jako maksimum dwóch wielkości (1) oraz funkcji przedstawionej na rysunku 1.

$$y = f(u) = \max(0.u) \tag{1}$$

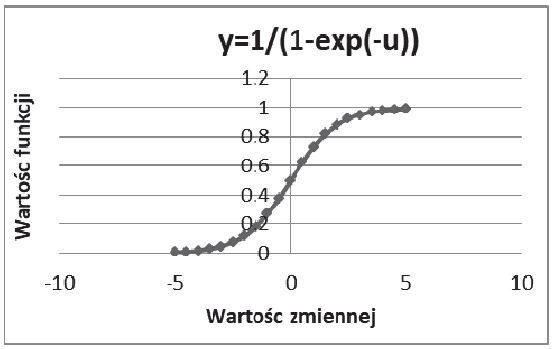


Rys. 1. Kształt funkcji aktywacji max(0,u)

Funkcja ta zastępuje tradycyjną funkcję aktywacji typu sigmoid (2) lub tanh (rys. 2).

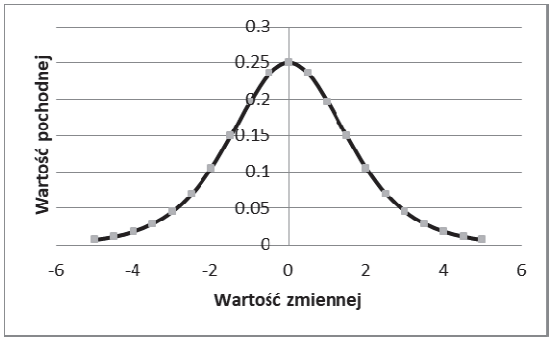
$$y = \frac{1}{1 + e^{-u}} \tag{2}$$

Funkcja sigmoidalna ma dwa odcinki o wartościach zbliżonych asymptotyczne do zera lub jedynki. Jej pochodna (rys. 3) przyjmuje znaczące wartości tylko w bardzo wąskim przedziale, a tym samym bardzo negatywnie wpływa na szybkość zbieżności.



Rys. 2. Kształt i wartość funkcji sigmoidalnej z obszarami nasycenia

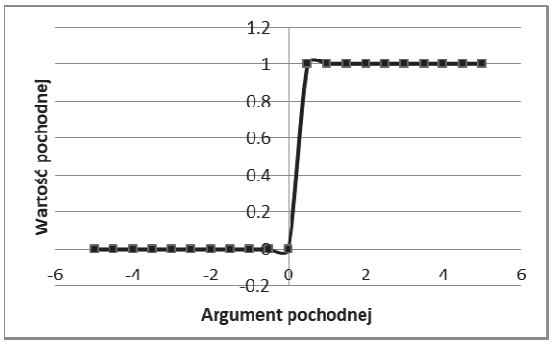
Z drugiej strony pochodna funkcji przyjmuje maksymalną wartość stanowiącą 25% wartości funkcji sigmoidalnej, co dodatkowo wpływa negatywnie na szybkość zbieżności.



Rys. 3. Kształt i wartość pochodnej funkcji sigmoidalnej

Z matematycznego punktu widzenia funkcja ReLu jest nieciągła w punkcie u=0. W aplikacjach komputerowych przyjmuje się, że dla u=0, wartość funkcji aktywacji y=0, unikając w ten sposób formalnych problemów. Kształt i wartości pochodnej funkcji aktywacji przedstawiono na rysunku 4. Dla argumentu u<0, pochodna jest równa zero, co ma ogromny wpływ na wartość wag w macierzach

warstw ukrytych. Duży procent wag nie zmienia swojej wartości lub przyjmuje wartości zerowe.



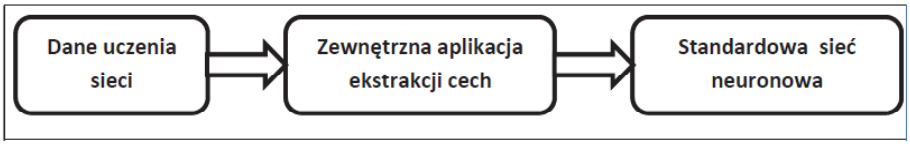
Rys. 4. Kształt i wartość pochodnej funkcji ReLu

2. STRUKTURA GŁĘBOKICH SIECI NEURONOWYCH

2.1. Charakterystyka różnic strukturalnych

Głębokie Uczenie sieci związane jest z pojawieniem się dwóch bardzo ważnych składników:

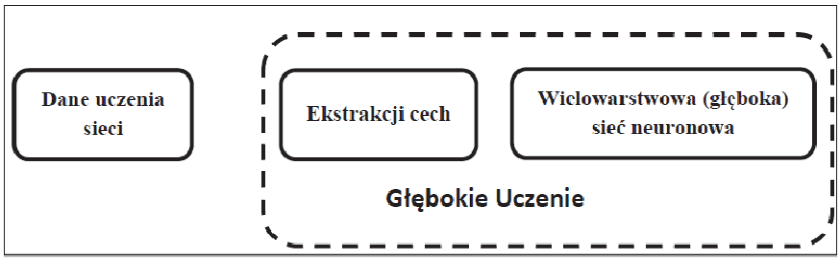
- Dużych baz danych zarówno zaetykietowanych jak i bez etykiet. Pierwsze są używane do uczenia sieci z nauczycielem (nadzorowane), drugie natomiast są używane do klasyfikacji (ang. clustering) dużych zbiorów danych o nieznanej liczbie klas. W strategii uczenia zbiory danych pierwotnych dzielone są na trzy rozłączne podzbiory: uczenia, testowania oraz weryfikacji.
- Pojawienie się szybkich wieloprocesorowych (wielordzeniowych) komputerów, co pozwala na praktyczną realizację złożonych struktur sieci oraz przetwarzanie dużych wielowymiarowych zbiorów danych w oparciu o zasadę równoległego przetwarzania



Rys. 5. Konwencjonalny schemat uczenia i przetwarzania informacji

W pierwszych zastosowaniach sieci neuronowych w zadaniach klasyfikacji, pełny algorytm przetwarzania składał się z dwóch kroków (rys. 5).

Na wstępnym etapie przetwarzania, niezależna aplikacja dokonywała kalibracji danych, filtrowania, skalowania i innych operacji. Tak skalibrowane dane wejściowe przekazywane są do głównej aplikacji uczenia maszynowego. Należy uwzględnić fakt, że wszystkie trzy podzbiory zdefiniowane powyżej muszą być przetworzone zgodnie z przedstawioną procedurą.
Procedura Głębokiego Uczenia integruje w jeden moduł wstępną ekstrakcję cech jak i uczenie końcowe (rys. 6). Powyższe jest możliwe mając do dyspozycji szybki, wielordzeniowy komputer z dużą pamięcią.



Rys. 6. Struktura zintegrowanego algorytmu uczenia

Niezależnie od przyjętego ostatecznego schematu przetwarzania, zawsze jest potrzebny szybki i wydajny algorytm uczenia szerokiej i głębokiej sieci neuronowej. W artykule przedstawiono wielomodułowy algorytm uczenia, poprzez przetwarzanie informacji w trzech powiązanych przebiegach:

- przetwarzania w przód, w którym wektor danych wejściowych podany na wejście sieci jest sukcesywnie przetwarzany aż do warstwy wyjściowej. Wektor wyjściowy, w zależności od stosowanej funkcji celu, jest porównywany z zadaną wielkością i stanowi ocenę jakości uczenia w danym kroku iteracji lub też całej epoki.
- Uwzględniając odchylenia obliczonego wyjścia z wartością zadaną realizowany jest algorytm wstecznej propagacji błędów, zbudowany jako suma standardowej procedury dla każdej warstwy sieci.
- Obliczony dla każdej warstwy błąd funkcji celu umożliwia obliczenie wpływu każdej wagi warstwy na końcowy błąd uczenia. Stosowanie iteracji po podaniu każdego wektora z epoki na wejście sieci (stochastyczny gradient minimalizacji), generuje stosunkowo duże zmiany funkcji celu i niestabilność procesu uczenia co objawia się trudnościami w określeniu zewnętrznego parametru uczenia jakim jest współczynnik uczenia "α". Z drugiej strony, przetwarzanie całej epoki jest procesem czasochłonnym i wolnobieżnym. W tym też celu dzieli się cały zbiór uczenia (epokę) na mniejsze porcje zwane potocznie „baczami”. Dobór wielkości mini-epoki nie jest zadaniem ła-

twym i pierwsze próby podejmuje się dla wielkości 25-30 wektorów z epoki. Jest to pewien procent z całości zbioru uczenia.

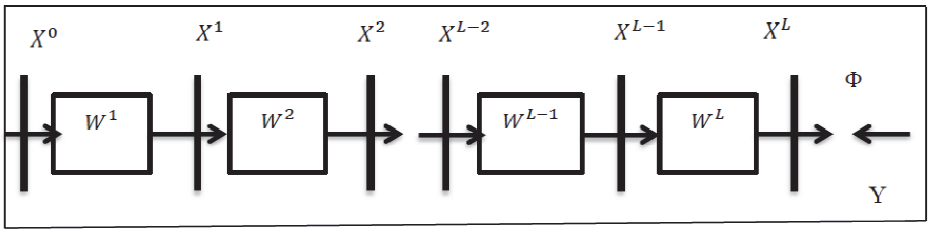
2.2. Koncepcja sieci głębokiego uczenia

Na wejście sieci podaje się dane obrazów biało - czarnych lub kolorowych. Bardzo często powyższe sieci stosuje się w medycynie do przetworzenia obrazów rentgenowskich, tomografii komputerowej itp. Generalnie obowiązuje jedna zasada, że wektor wejściowy jest duży. W przypadku obrazu kolorowego, rozmiar danych wejściowych można przedstawić jako tensor ilości pikseli RGB, czyli trójwymiarową macierz:

$$H \cdot W \cdot D \tag{3}$$

gdzie: H- ilość pikseli w pionie obrazu, W- ilość pikseli w szerokości obrazu, D- ilość kolorowych pod ekranów RGB. D=3 lub D=4, w zależności od sposobu kodowania.

Kompresja danych w procesie ekstrakcji cech, wymaga wielu warstw. Ogólny schemat sieci głębokiego uczenia przedstawiono na rys. 7.



Rys. 7. Struktura sieci głębokiego uczenia

Sieć neuronowa głębokiego uczenia składa się z jednolitych topologicznie i logicznie modułów, zwanych warstwami. W celu uniknięcia nieporozumienia, warstwy sieci numerujemy od pierwszej warstwy ukrytej ,kończąc na ostatniej warstwie wyjściowej. Tak więc schemat sieci na rys. 7. zawiera „L" warstw, w tym „L-1" ukrytych i jedną wyjściową. Wymiary macierzy wag wszystkich warstw sieci uzależnione są od rozmiarów wektorów wyjścia poszczególnych warstw. Zagadnienie doboru ilości warstw sieci jak i dystrybucji ilości neuronów pomiędzy warstwami, należy do najtrudniejszych zadań projektanta sieci. Wzdłuż sieci następuje kompresja danych wraz z ekstrakcją elementów cech. W warstwie wyjściowej następuje końcowe sumowanie składowych cząstkowych. Proces ten jest bardzo skomplikowany i jak dotychczas nie istnieje teoria, która w oparciu o wartości współczynników wag macierzy poszczególnych warstw może określić wyselekcjonowane cechy.

Należy zwrócić uwagę, że wektor wyjściowy X warstwy „l" jest jednocześnie wektorem wejściowym warstwy „l+1". Powyższa notacja pozwala na zdefiniowanie procedury obliczeniowej stosowanej w każdej warstwie, zarówno dla obliczeń w przód jak i obliczeń zwrotnych, do tyłu.

Dla każdej warstwy ukrytej obliczamy

$$u_i^l = \sum_{j=0}^{j^{l-1}} w_{ij}^l \cdot x_j^{l-1} \tag{4}$$

gdzie: u_i^l - pomocniczy wektor warstwy „l", zdefiniowany jako wejście funkcji aktywacji. Wymiar wektora [1:i^l] dla l=1,2,…L-1,L, L- głębokość sieci, ilość warstw.

Wektor wejściowy mnożony jest przez współczynniki wagowe macierzy W^1 tworząc wektor pośredni wyjścia podsieci nr 1 U^1.

Wektor wyjścia danej warstwy 'l" zdefiniowany jest przyjęta funkcją aktywacji

$$x_i^l = \sigma\left(u_i^l\right) \tag{5}$$

Począwszy od warstwy wejściowej i wektora wejściowego X_0 , w następnych warstwach obliczane są wartości wektorów $U^l, X^l \; dla\, l = 1,2,\ldots L$. W warstwie wyjściowej definiowana jest funkcja celu, często nazywana funkcją błędu.

$$\Phi = \frac{1}{2} \cdot \left\| X^L - Y \right\|^2 \tag{6}$$

gdzie: Y – wektor danych uczących, X^L - wektor wyjścia sieci (warstwy wyjściowej).

Kolejnym krokiem jest obliczenie pochodnych funkcji celu względem wszystkich wag sieci (dla wszystkich warstw).

2.3. Algorytm wstecznej propagacji błędu

W oparciu o rysunek 7, przetwarzanie w przód zapisujemy w formie łańcucha.

$$X_0 \to W^1 \to U^1 \to X^1 \to W^2 \to X^2 \to \ldots \to X^{L-1} \to W^L \to X^L \to \tag{7}$$

Pochodną funkcji celu względem wektora X^{L-1} obliczamy jako pochodną funkcji złożonej:

$$\frac{\partial \Phi}{\partial X^{L-1}} = \frac{\partial \Phi}{\partial X^L} \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial X^{L-1}} \tag{8}$$

gdzie:

$$\frac{\partial \Phi}{\partial X^L} = X^L - Y \tag{9}$$

$$\frac{\partial X^L}{\partial U^L} = \sigma'\left(U^L\right) \tag{10}$$

Wzór (10) określa pochodną funkcji aktywacji względem wektora U^L.

Wzór (8) określa sposób przekazywania części błędu funkcji celu do następnej warstw „L-1". W podobny sposób obliczamy pochodną funkcji celu względem wag warstwy „L"

$$\frac{\partial \Phi}{\partial W^L} = \frac{\partial \Phi}{\partial X^L} \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial W^L} \tag{11}$$

Wzory (8) i (11) posiadają identyczne części. Oznaczmy je odpowiednio jako wektor „ε" czyli błąd przekazywany na początek. Natomiast błąd przekazywany do środka warstwy oznaczmy przez wektor „δ"

$$\varepsilon^L = \frac{\partial \Phi}{\partial X^L} \tag{12}$$

$$\delta^L = \frac{\partial \Phi}{\partial X^L} \cdot \frac{\partial X^L}{\partial U^L} = \frac{\partial \Phi}{\partial X^L} \cdot \sigma'\left(U^L\right) \tag{13}$$

W celu poprawnego przekazania błędu funkcji celu z warstwy "l+1" do warstwy "l", utwórzmy odwrotny ciąg przekształceń podobny do wzoru (7).

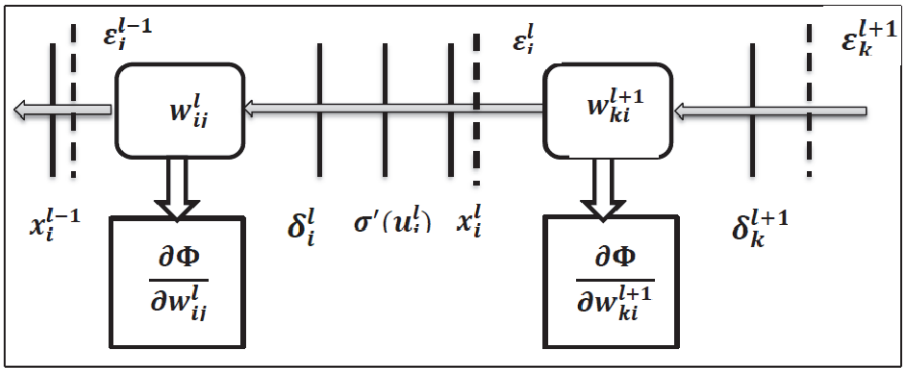
$$W^l \leftarrow \delta^l \leftarrow \sigma'\left(U^l\right) \leftarrow X^l \leftarrow W^{l+1} \leftarrow \sigma^{l+1} \leftarrow \tag{14}$$

Uwzględniając zasady mnożenia macierzy przez wektor oraz operację wektorową nazywaną iloczynem Hadamarda $\bullet$, obliczamy wartość błędu funkcji celu przekazywanego przez warstwę „l+1" do „l" (w tył).

$$\delta^l = \left(W^{l+1}\right)^T \cdot \delta^{l+1} \cdot \sigma'\left(U^l\right) \tag{15}$$

Struktura algorytmu wstecznej propagacji błędu przedstawiono na (rys. 8). W oparciu o oznaczenia jak na rysunku, można napisać bardziej szczegółowy wzór na wsteczną propagację błędu wzdłuż warstw.

$$\delta_i^l = \sigma'\left(u_i^l\right) * \sum_{k=1}^{k^{l+1}} w_{ki}^{l+1} \cdot \delta_k^{l+1}. \tag{16}$$



Rys. 8. Schemat algorytmu wstecznej propagacji błędu

Uwzględniając wzór (12) oraz (rys. 8.) definiujemy drugi sposób obliczania błędu przekazywanego pomiędzy warstwami. Błąd „δ^l". ingeruje w wewnętrzną strukturę warstwy, a mianowicie do swojej struktury włącza pochodną funkcji celu danej warstwy. Na (rys. 8.) granice warstw sieci oznaczone są pionowymi przerywanymi liniami. Warstwa sieci zaczyna się i kończy wektorem wejściowo – wyjściowym X. Chcąc porównać wpływ parametrów poszczególnych warstw (wag w warstwie, funkcji aktywacji) na szybkość przekazywania błędu lub inne interesujące parametry, w przypadku błędu „ δ^l". napotykamy określone trudności. Wprowadzenie błędu granicy warstwy „ε^l. ", powyższe trudności usuwa. Porównajmy parametry warstw.
Dla warstwy „l":

$$\varepsilon_i^l = \sum_{k=1} \varepsilon_k^{l+1} \cdot \sigma_k^{'}\left(u_k^{l+1}\right) \cdot w_{ki}^{l+1} . \tag{17}$$

Ostatecznie pochodna względem wag:

$$\frac{\partial \Phi}{\partial w_{ij}^l} = \varepsilon_i^l \cdot \sigma_i^{'}\left(u_i^l\right) \cdot x_j^{l-1} . \tag{18}$$

Podobnie dla błędu „δ^l". , warstwy „l" można bardzo prosto obliczyć pochodną funkcji celu względem macierzy wag danej warstwy:

$$\frac{\partial \Phi}{\partial w_{ij}^l} = x_i^{l-1} \cdot \delta_j^l \tag{19}$$

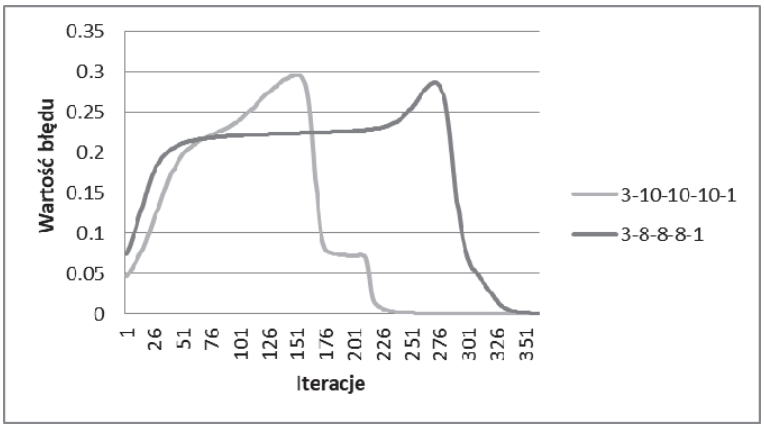
Ostatecznie, stosując metodę gradientową obliczenia wartości wag w poszczególnych iteracjach, wzór obowiązujący dla każdej warstwy pybiera postać (górny indeks „l" zostaje opuszczony).

$$w_{ij}^{t+1} = w_{ij}^{t} - \alpha \frac{\partial \Phi}{\partial w_{ij}^{l}} \tag{20}$$

Zależności od potrzeb można stosować różne podejścia do algorytmu wstecznej propagacji błędu. Różnica w definicji przekazywanego błędu jest niewielka. Wzór (12) może być bardzo przydatny w przypadku aliz wydajności przetwarzania w warstwach ukrytych. W artykule ocenie podlega warstwa wyjściowa i jej funkcja celu, a tym samym w dalszej analizie koncentrujemy się na definicji (13).

3 PRZYKŁAD I WNIOSKI

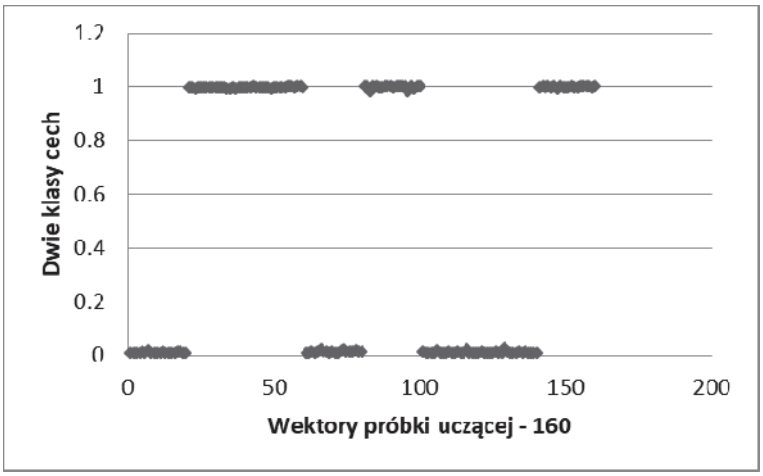
W celu przeanalizowania dynamiki uczenia sieci głębokiej (posiadającej więcej niż jedną warstwę ukrytą), wybrano sieć o 3 warstwach ukrytych. Przeanalizowano wiele struktur sieci, jednak dla porównania charakterystyk uczenia sieci wybrano dwie : 3-8-8-8-1 oraz 3-10-10-10-1. Sieci posiadają identyczną strukturę : warstwa wejściowa zawiera 3 neurony, a warstwa wyjściowa jeden. Trzy warstwy ukryte w danej konfiguracji sieci, różnią się ilością neuronów w warstwach ukrytych – 8 lub 10 odpowiednio. Sieć głęboka konfiguracyjnie różni się od sieci wysokiej (z jedna warstwą ukrytą). Jedna warstwa ukryta zawiera o wiele więcej neuronów niż w sieci głębokiej.



Rys. 9. Charakterystyka dynamiki uczenia sieci w funkcji numeru iteracji

Warstwa ukryta musi mieć możliwość zdekodowania cech w danym wektorze wejściowym. Do tego potrzebna jest odpowiednia ilość neuronów. W warstwie wyjściowej następuje złożenie zdekomponowanych cech. W sieci głębokiej możliwości konfiguracyjne są bardzo duże i trudno wybrać strukturę opty-

malną. W artykule zdecydowano się na wybór konfiguracji sieci zawierającej identyczną ilość neuronów w warstwach ukrytych.
Ostatecznie siec głęboka zawiera w swojej strukturze o wiele więcej wag w macierzach połączeń. Uczenie takiej sieci napotyka na wiele trudności. Po pierwsze, do uczenia sieci potrzebny jest stosunkowo duży zbiór danych.



Rys. 10. Wynik klasyfikacji zbioru wejściowego na dwie klasy

Po drugie, jeżeli ilość danych jest ograniczona, to stosować trzeba skomplikowane techniki regularyzacyjne, co komplikuje algorytm uczenia.

Do uczenia sieci wykorzystano zbiór wejściowy o trzech zmiennych zawierający dwie klasy. Całkowity zbiór danych podzielono na trzy rozdzielne podzbiory: zbiór uczący – 160 wektorów w epoce, zbiór testowy – 50 wektorów oraz zbiór weryfikujący 30 wektorów. Powyższa dekompozycja jest zgodna z teoretycznymi zaleceniami lecz ma bardzo niekorzystny wpływ na wielkość zbioru uczenia – ulega zmniejszeniu.
Wniosek : Uczenie sieci głębokiej jest możliwe pod dwoma warunkami. Posiadając dostatecznie duży zbiór uczący oraz stosujemy odpowiednią funkcję aktywacji. Standardowo stosowana funkcja aktywacji typu sigmoid, obniża amplitudę błędu w każdej warstwie o 25%, co w konsekwencji powoduje zanik błędu uczenia w warstwach głębokich. Funkcja typu ReLu posiada pewne teoretyczno – matematyczne ograniczenia, które w zadaniach uczenia maszynowego są pomijane.

Proponowane w artykule rozwiązania są częścią dużego tematu związanego z przetwarzaniem obrazów medycznych i klasyfikacją cech z wykorzystaniem wielowarstwowych sieci neuronowych. Obrazy 2D lub 3D wymagają przetwa-

rzania dużej ilości pikseli, a tym samym wydajność algorytmu uczenia jest bardzo istotna.

LITERATURA

[1] Quoc V. Le, Part 1: Nonlinear Classifiers and The Backpropagation Algorithm, Google Brain, Google Inc, CA 94043 2015.
[2] Jianxin Wu, Introduction to Convolution Neural Networks, Nanjing University, China 2017.
[3] http://cs231n. github.io/optimization-1.
[4] Ian Goodfellow, Yoshua Bengio, Aaron Courvile. Deep Learning, .2016
[5] Sebastian Raschka, Python Machine Learning.
[6] S. Kevin Zhou, Hayit Greenspan, Dinggang Shen, Deep Learning for Medical Image Analysis, 2016.

TEACHING MULTILAYER WIDE NEURAL NETWORKS WITH RELU ACTIVATION FUNCTION IN THE CLASSIFICATION TASKS

In the article, a new way of artificial neural network development in the classification task is introduced. In the past, neural networks with two or maximum three hidden layers were used. The sigmoid or tanh activation functions were implemented as well. These functions have very interesting properties that are very useful in the learning algorithms. Unfortunately, they have a saturation area for the small and big argument's value. As a consequence, if the derivatives are calculated in every hidden layer, they values are very small, near zero. It has a very negative impact on the property of the learning algorithm. In this area, an algorithm is working very slowly. Two factors now have big impact on the neural network development: big databases and power microprocessors. Therefore, a deep neural network with many hidden layers could be used in practice tasks. To improve the gradient calculation a new activation function, ReLU, is used. In the article, the properties of these neural networks are studied. It is the first step to building more powerful networks that are known as Convolutional Neural Networks.

(Received: 31.01.2018, revised: 10.03.2018)