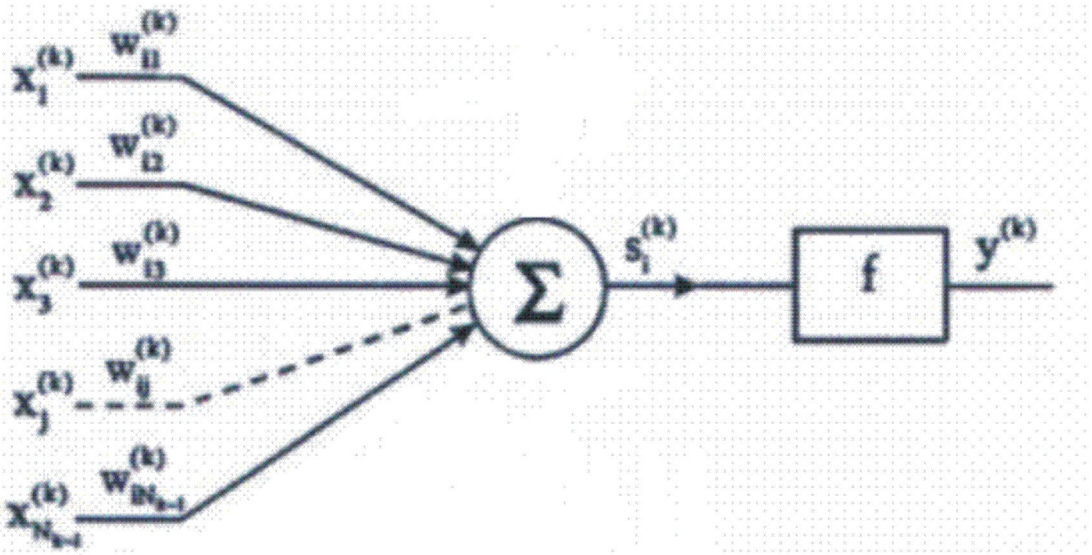
8. Neuron z ciągłą funkcją aktywacji.

W tym ćwiczeniu zapoznamy się z modelem sztucznego neuronu oraz przykładem jego wykorzystania do rozwiązywanie prostego zadania klasyfikacji.

Neuron biologiczny i jego sztuczny model

Sieć neuronowa to wysoce równoległy, rozproszony procesor złożony z prostych elementów obliczeniowych zwanych neuronami. Pierwowzorem dla stworzenia sztucznych sieci neuronowych był model sieci neuronowej organizmów żywych. Podobnie jak neuron z systemu nerwowego istot żywych sztuczny neuron posiada kilka wejść oraz jedno wyjście, będące przetworzoną informacją wejściową. Model neuronu skła-

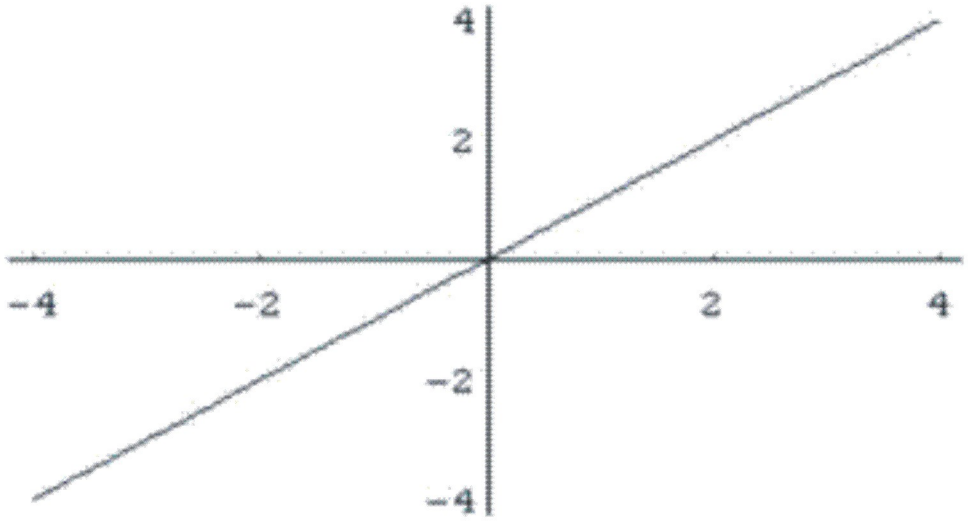


Rysunek 1: Model neuronu.

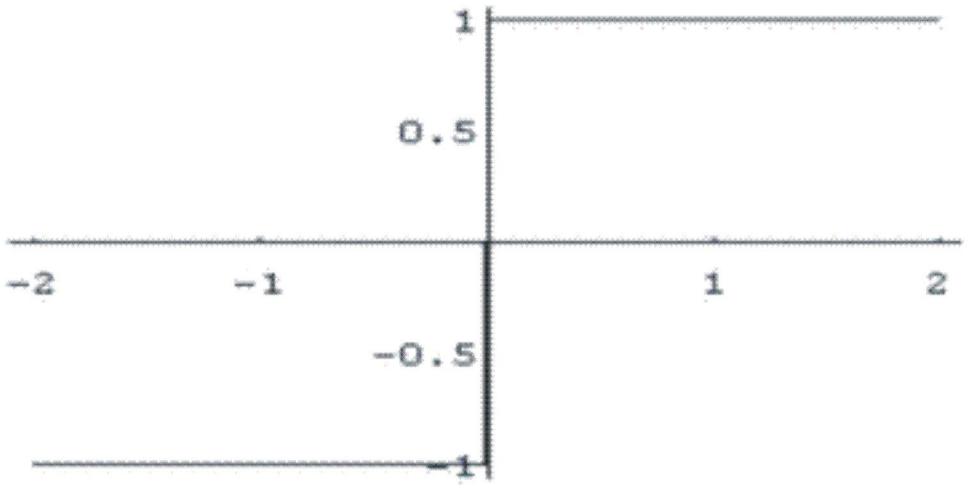
da się z elementu sumacyjnego, do którego dochodzą sygnały wejściowe $x_1, x_2, ..., x_n$ tworzące wektor wejściowy $x = [x_1, x_2, ..., x_n]$ pomnożony przez przyporządkowane im wagi $w_1, w_2, ..., w_n$. Sygnał wejściowy sumatora to suma wartości wejściowych x_i pomnożonych przez odpowiadające im wagi w_i. Informacja wyjściowa neuronu jest przetworzonym sygnałem wejściowym sumatora przez tzw. **funkcję aktywacji f**.

Najczęściej stosowanymi funkcjami aktywacji są:

1. Funkcja liniowa.
2. Funkcja sgn.



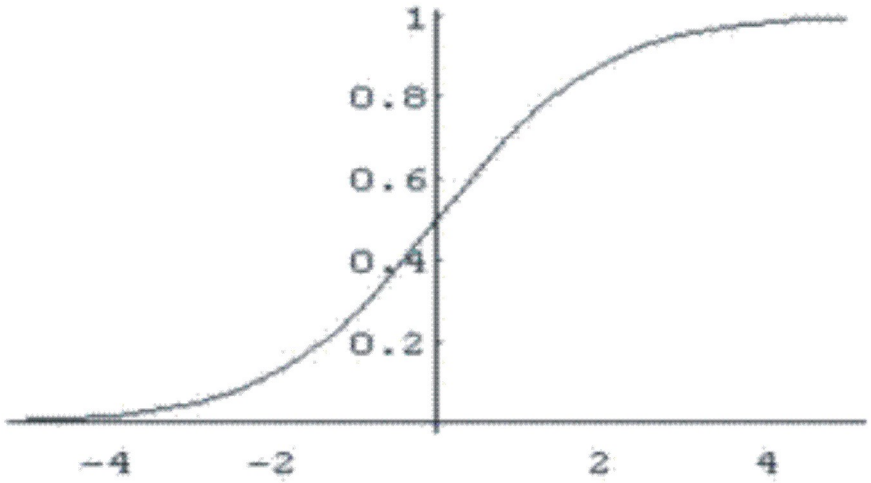
Rysunek 2: Funkcja liniowa.



Rysunek 3: Funkcja sgn.

3. Funkcja sigmoidalna.

Mówiąc najogólniej jak tylko można, z czysto użytkowego punktu widzenia, neuron można traktować jak czarną skrzynkę. Skrzynkę posiadającą pewną ilość wejść i tylko jedno wyjście. Cel jaki zamierzamy osiągnąć to "spowodowanie" aby w odpowiedzi na pewne zadane sygnały wejściowe skrzynka ta zwracała konkretne warości na swoim wyjściu. Dużo bardziej jednak istotne jest to, aby o podaniu na wejście sygnałów nie identycznych, ale podobnych do wcześniej zadanych również zwracała prawodłowe wartości (czyli odpowiedzi). Tak będzie na przykłd z literami. Możemy neuronowi zaprezentować na przykład 2 litery A i C i przypisać odpowiednio wartości sygnałów wyjściowych równe 0 i 1. Oczekiwać będziemy, że po podaniu A otrzymamy 0, 1 gdy podamy C. Jeszcze bardziej jednak ucieszy nas, gdy po podanu inaczej zapisanej litery A otrzymamy odpowied bliską wzorcowej, na przykład 0.001. Podobnie dla C. Aby ta "czarna skrzynka" mogła spełnić to nasze żądanie musimy nauczyć neuron prawidłowego odwzorowania. Właśnie nauką neuronu w tym ćwiczeniu się zajmiemy.



Rysunek 4: Funkcja sigmoidalna.

Nauka

Jedynymi parametrami, za pomocą których możemy sterować zachowaniem neuronu są wagi. Od ich doboru będzie zależało jakie zadania i w jakim stopniu neuron będzie w stanie wykonać. Na początku podkreślmy to o czym już powiedzieliśmy a co być może umknęło nam. Po pierwsze musimy mieć zdefiniowane dwa zbiory.

1. Zbiór pierwszy, to zbiór sygnałów jakie będą podawane na wejście neuronu; w przykładzie – sygnały reprezentujące litery A i C).

2. Zbiór drugi, to zbiór prawidłowych sygnałow wyjściowych neuronu (w przykładzie 0 i 1).

Oznaczmy pierwszy z tych zbiorów przez P, drugi przez T. Elementy tych zbiorów będziemy oznaczac prza małe literki – odpowiednio p i t, z ineksem wskazujacym na numer. Tak wiec mamy: $P = \{p_1, p_2\}$ $T = \{t_1, t_2\} = \{0, 1\}$ gdzie p_1 i p_2 to wektory powstałe w oparciu o poniższe obrazki, według zasady: czarne pole = 1, białe = 0 (odstępu co 5 dodane są dla czytelnosci):

```
XXXXX        XXXXX
X   X        X
X   X        X
XXXXX        X
X   X        X
X   X        X
X   X        XXXXX
```

p_1 =11111 10001 10001 11111 10001 10001 10001
p_2 =11111 10000 10000 10000 10000 10000 11111
Oba zbiory tworzą zbiór uczący $L = \{P, T\}$ Powiedzieliśmy także, że chcemy aby wyście z neuronu różniło się jak najmniej od odpowiedzi oczekiwanej przez nas, czyli chcemy aby różnica $(t_i - y_i)$ dla $i = 1, ..., n$ była jak najmniejsza (przy podawanym na wejście p_1 lub p_2). Inaczej mówiąc dążymy do minimalizacji określonej tym sposobem funkcji. Skoro jednak ma to być funkcja, to powinna mieć jakąś zmienną (lub zmienne). Przed chwilą powiedzieliśmy, że jedyne co może sie w neuronie zmieniać to jego wagi. Ostatecznie więc otrzymujemy nastepującą definicję funkcji $E(w) = (t_i - y_i)$,

gdzie $w = [w_1, ..., w_n]$ Po prawej stronie nie widać aby y_i zależało od w, ale łatwo jest to zmienić... Zgodnie z informacjami ze wstępu

$$net_i = \sum_{k=1}^{n}(x_k^i w_k)$$

$$y_i = f(net_i)$$

a więc

$$E(w) = (t_i - f(\sum_{k=1}^{n} x_k^i w_k))$$

Zwykle aby zapewnić sobie, że funkcja E ma minimum i aby łatwiej się nam potem liczyło, przyjmuje się (i my też tak zrobimy)

$$E(w) = \frac{1}{2}(t_i - y_i)^2$$

Mamy więc zdefiniowaną pewną funkcję i poszukujemy jej minimum. Do tego celu świetnie nadaje sie metoda gradientowa – metoda wyznaczania minimum funkcji. Zgodnie z nią

$$w(t+1) = w(t) - \eta E'(w(t))$$

Teraz musimu zdecydować się na funkcję stosowaną jako funkcja wyjścia. Przyjmijmy, że

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

Policzmy zatem gradient funkcji E

$$\nabla E = \left[\frac{\partial E}{\partial w_1}, ..., \frac{\partial E}{\partial w_n}\right]$$

$$\frac{\partial E}{\partial w_p} = E'_{w_p}(w) = \left[\frac{1}{2}(t_i - f(\sum_{k=1}^{n} x_k^i w_k))^2\right]'_{w_i} =$$

$$\frac{1}{2}2(t_i - f(\sum_{k=1}^{n} x_k^i w_k))(t_i - f(\sum_{k=1}^{n} x_k^i w_k))' =$$

$$= -(t_i - f(\sum_{k=1}^{n} x_k^i w_k))(\sum_{k=1}^{n} x_k^i w_k)'_{w_p} f'(\sum_{k=1}^{n} x_k^i w_k) =$$

$$= -(t_i - f(\sum_{k=1}^{n} x_k^i w_k)) x_p^i f'(\sum_{k=1}^{n} x_k^i w_k) =$$

$$= -(t_i - f(net_i)) x_p^i f'(net_i) =$$

$$= -(t_i - y_i) x_p^i f'(net_i).$$

Iloczyn $(t_i - y_i)f'(net_i)$, czyli $\frac{\partial E}{\partial net}$ gdyż $\frac{\partial E}{\partial w_p} = \frac{\partial E}{\partial net}\frac{\partial net}{\partial w_p}$ nazywany jest sygnałem błędu delta i oznaczamy go δ. Następnie

$$f'(x) = f(x)(1 - f(x))$$

Uwaga: zatem pochodna w punkcie x wyraża się przez wartość tejże funkcji w tym samym punkcie x. Oznacza to, że nie musimy przeprowadzać prawie żadnych dodatkowych obliczeń w celu obliczenia pochodnej. Ostatecznie

$$\frac{\partial E}{\partial w_p} = -(t_i - y_i)x_p f(net_i)(1 - f(net_i)) = -(t_i - y_i)x_p y_i(1 - y_i)$$

(gdzie w naszym przypadku $i = 1, 2$). W praktyce jednak i zmienia się w dużo większym zakresie. Po prostu po jednokrotnej prezentacji próbki uczącej p_1 (to znaczy podaniu na wejścia neuronu sygnałów z p_1) i jednokrotnej prezentacji p_2 wprowadzone poprawki wag będą niewielkie. Zatem ponownie będziemy musieli zaprezentować neuronowi dane wejściowe p_1 i p_2 a następnie jeszcze raz i jeszcze raz i jeszcze... Dlatego też możemy przyjąć, że na przykład $i = 1, 2, ...$ i gdy i jest liczbą nieparzystą to na wejście podajemy p_1, gdy zaś i jest liczbą parzystą, to p_2. Teraz wiemy już wszystko aby móc nauczyć nasz neuron rozpoznawania liter A i C. Pozostaje jedynie w bardziej zwartej formie zaprezentować rozważany algorytm.

Algorytm

Mamy dany zbiór uczący L następującej postaci: $L = \{P, T\}$ $P = \{p_1, p_2, ..., p_{ilośćobrazów}\}$ $T = \{t_1, t_2, ..., t_{ilośćobrazów}\}$ $t_k \in [0, 1]$ $k-$ numer obrazu, $k = 1, 2, ..., ilośćobrazów$ $p_i = [p_{i,1}, ..., p_{i,n}]$, gdzie $n-$ ilość wejść

1. Wybór $\eta > 0$ (współczynnik uczenia), $E_{max} > 0$ (maksymalny błąd jaki chcemy osiągnąć), $C_{max} > 0$ (ilość kroków uczenia).

2. Losowy wybór początkowych wartości wag (zmienne $w_1, ..., w_n$) jako niewielkich liczb (na przykład z przedziału $[-1, 1]$). $c := 0$.

3. $l := 0$, $E := 0$

4. Podanie jednego z obrazów ze zbioru P (na przykład $k - tego$) na wejscia neuronu $x_i = p_{k,i}$ $k-$ numer obrazu, $n = 1, ..., n$, $n-$ ilosć wejść $d = t_k$ i obliczenie sygnału wyjściowego neuronu $y = f(net)$ $net = \sum_{k=1}^{n} x_k w_k$

5. Uaktualnienie wartości wag według wzoru $w_p(t+1) = w_p(t) + \eta(d-y)(1-y)x_p y$

6. Obliczanie błędu $E = E + \frac{1}{2}(d - y)^2$

7. Jesli $l < ilośćobrazów$ to $l := l + 1$ i przejscie do kroku 4.

8. Jesli E $< E_{max}$, to kończymy algorytm. Jeśli $c < C_{max}$, to $c := c + 1$ i przechodzimy do kroku 3. W przeciwnym razie kończymy algorytm.

Teraz kilka uwag:

1. W kroku 4 wybieramy jeden z obrazów ze zbioru P. Najlepiej jeśli kolejność wybieranych obrazów będzie losowa.

2. Zmienna c wskazuje na kolejne cykle uczące. Jak widać jeden cykl składa się z prezentacji wszystkich wzorców uczących.

Zadanie

Należy zaimplementować zaprezentowany algorytm dla identycznego zadania jak rozważanego w tekście (rozpoznawanie pewnych liter). Program ma mieć możliwość, po nauczeniu neuronu, podawania przez użytkownika wartości sygnałów wejściowych i obliczenia dla nich odpowiedzi neuronu.