SIECI NEURONOWE – laboratorium

Ćwiczenie 1 – proste sieci neuronowe

Celem ćwiczenia jest poznanie podstawowych funkcji wykonywanych przez pojedynczy neuron, obserwacja zachowania neuronu przy różnych funkcjach przejścia oraz określenie wielkości, które mają wpływ na szybkość uczenia neuronu. Ćwiczenie to realizowane jest na dwóch laboratoriach. Na pierwszym z nich realizowany jest model perceptronu prostego, na drugim model Adaline. Implementacja modeli powinna być w Pythonie. Można korzystać z bibliotek do obliczeń matematycznych np.mnożenia macierzy

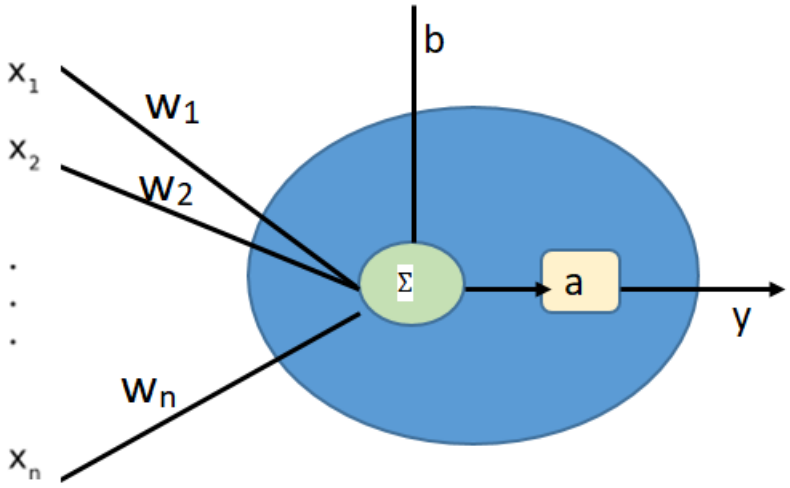
Liczba punktów: 20

Uwaga: Raport z ćwiczenia nie powinien zawierać części teoretycznej a jedynie dokumentować przeprowadzone eksperymenty i płynące z nich wnioski. Każdy eksperyment powinien być powtórzony 10- krotnie. Raport powinien być wykonany w szablonie zostawionym na eportalu w tex i przesłany na eportal do oceny

Laboratorium 1: Prosty perceptron

Wprowadzenie teoretyczne

Idea perceptronu wzięła się od badań nad możliwościami klasyfikowania widzianych wzorców optycznych, tzn. od fotoperceptronu, który był sprzętową realizacją modelu neuronu. Perceptron prosty to model, który będziemy realizować jako model neuronu w postaci programowej. Jego struktura przedstawiona jest na rys. 1.



Rys. 1. Perceptron prosty

Odpowiada on modelowi opisanemu przez Pittsa- McCullocha. Na wejście podawane są sygnały wejściowe $x_1,x_2,\ldots,x_n$. Płyną one przez połączenia, do których przypisane są wagi $w_1,w_2,\ldots,w_n$. Odzwierciedlają one siłę połączeń. Wagi mogą być dodatnie, wówczas odpowiadają synapsom wzmacniającym, kiedy są ujemne odpowiadają połączeniom hamującym. Wagi są liczbami w zakresie [-1,1]

Przetwarzanie sygnału w neuronie jest dwustopniowe. W pierwszej kolejności obliczane jest całkowitego pobudzenie *z,* jako suma (na Rys. 1. oznaczona jako Σ) wejść x_i ważonych przy użyciu wag w_i. Możemy zapisać, że całkowite pobudzenie neuronu wyraża się następująco:

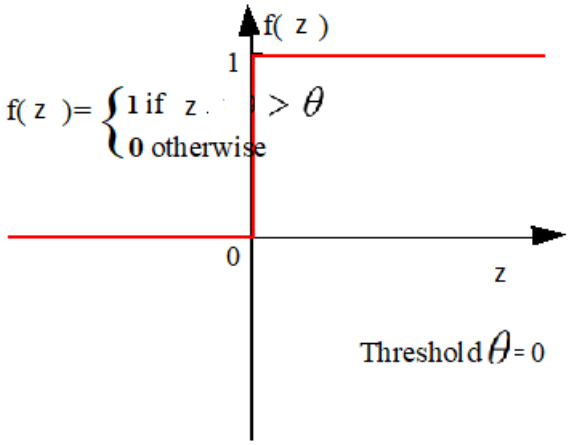
$$z = \sum_{i=1}^{n} w_i \, x_i$$

Dalej następuje przetwarzanie przez funkcję aktywacji (na Rys.1. oznaczona jest jako a). Dla perceptronu prostego jest to funkcja progowa. Wyjście y z neuronu jest równe 1, gdy spełniony jest warunek:

$$\sum_{i=1}^{n} w_i \, x_i > \theta \qquad (1)$$

W tej nierówności θ jest progiem.

Możemy zastosować funkcje progowe w logice unipolarnej (funkcję Heaviside) lub w logice bipolarnej. Na Rys. 2. pokazana jest funkcja progowa unipolarna

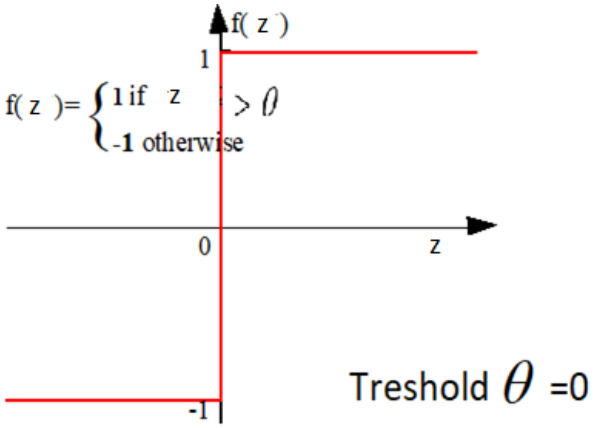


Rys.2. Funkcja progowa unipolarna

Funkcję tę możemy formalnie zapisać jako

$$a = f(z) = \begin{cases} 1 \; jeśli \; z > \theta \\ 0 \; w \; przeciwnym \; przypadku \end{cases} \quad (2)$$

Rys. 3 przedstawia funkcję bipolarną.



Rys.3 Funkcja progowa bipolarna

Formalnie możemy ją wyrazić w następujący sposób:

$$a = f(z) = \begin{cases} 1 \; jeśli \; z > \theta \\ -1 \;\; w \; przeciwnym \; przypadku \end{cases} \quad (3)$$

Tak więc, na wyjściu wartość y w zależności od zastosowanej funkcji aktywacji przyjmuje wartości 0 albo1 lub -1 albo 1.
Warto zauważyć, że kiedy stosujemy funkcję bipolarną, dobrze jest również zamodelować wejścia jako sygnały bipolarne przez reprezentację wartości równej 0 jako -1.
Wartość progu θ może przyjmować wartości w przedziale od $-\infty$ do $+\infty$.

Ręczne ustawianie progu jest dość kłopotliwe, dlatego wygodniej jest korzystać z automatycznie dostrajanego progu. Możemy to zrobić wprowadzając dodatkowe połączenie zwane biasem, oznaczane na Rys. 1 jako b. Na połączeniu tym zawsze płynie sygnał równy 1, a waga w_0 do niego przypisana jest poszukiwana w procedurze uczenia, tak jak inne wagi. W takim przypadku całkowite pobudzenie z obliczane jest jako:

$$z = \sum_{i=0}^{n} w_i \, x_i \tag{4}$$

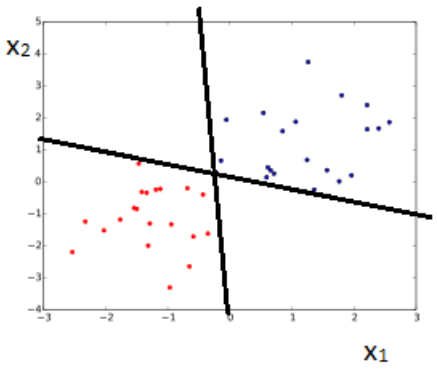
Natomiast nierówność z progiem sprowadza się do sprawdzenia czy suma ważonych sygnałów jest większa od 0.

$$\sum_{i=0}^{n} w_i \, x_i > 0 \tag{5}$$

Podsumowując w perceptronie prostym możemy wyróżnić następujące elementy

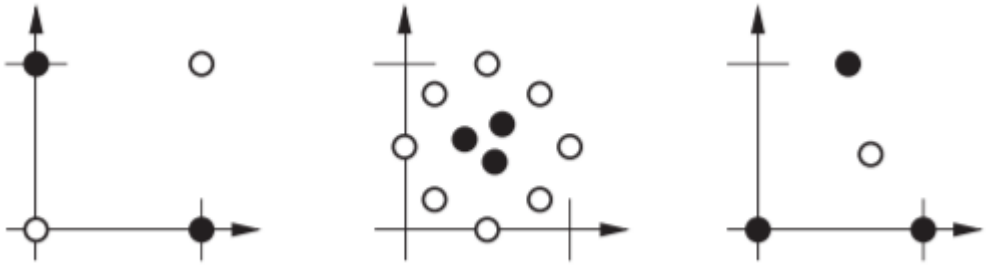
- Wejścia ($x_1,\ldots,x_n$).
- Połączenie b zwane biasem z wagą w_0, które zastępuje ustawiany ręcznie próg θ
- Wagi na połączeniach $w_1,\ldots,w_n$.
- Funkcja Σ sumująca ważone połączenia, dzięki której możliwe jest policzenie całkowitego pobudzenia z neuronu.
- Funkcja aktywacji a.
- Wyjście y, oznaczające odpowiedź

Perceptron prosty, podobnie jak fotoperceptron, służy do klasyfikacji binarnej wzorców (rozpoznaje czy wzorzec wejściowy należy do danej klasy czy nie). Rozważmy dla ułatwienia problem w dwóch wymiarach (Rys.4.), gdzie przedstawiono punkty należące do dwóch klas (oznaczonych kolorem niebieskim i czerwonym). Punkty te tworzą nasze wektory wejściowe o dwóch współrzędnych [x_1,x_2]



Rys.4. Przykład dwóch klas i podziału przestrzeni dokonywanego przez perceptron

Przetwarzanie sygnału wejściowego w perceptronie sprowadza się do obliczenia w pierwszej kolejności całkowitego pobudzenia a więc $w_1x_1+w_2x_2$. Wartość wyjścia z perceptronu obliczana jest na podstawie obliczenia wartości funkcji aktywacji, która ustana jest przez sprawdzenie nierówności $w_1x_1+w_2x_2>\theta$. Jeśli jest ona prawdziwa, to wskazuje na daną klasę, w przeciwnym przypadku $w_1x_1+w_2x_2<\theta$ wzorzec należy do klasy przeciwnej. A więc te dwie klasy rozdzielane są przez $w_1x_1+w_2x_2=\theta$. Jest to równanie prostej. Na Rys.4 pokazano 2 przykładowe proste oddzielające te dwie przykładowe klasy. Poznaliśmy istotne ograniczenie tego modelu – zdolność do rozwiązywania problemów, w których regiony zawierające klasy wzorców dają się oddzielić linią prostą. Takie problemy nazywamy separowalnymi liniowo.
Warto dodać, że percepton wywołał wiele kontrowersji, w tym czasie kiedy został opisany. Nierealistyczne oczekiwania spowodowały, że na wiele lat zostały zatrzymane badania nad tą dziedziną sztucznej inteligencji. W 1969 roku pojawiła się książka Marvina Minskiego i Seymoura Paperta, będąca totalną krytyką perceptronu. Zaprezentowali oni analizę możliwości i ograniczeń perceptronu. Wynikało z niej, że istnieją poważne ograniczenia na klasy problemów, jakie mogą być odpowiednie dla perceptronu. Perceptron może rozróżniać wzorce, jeżeli są one liniowo separowalne. Ponieważ wiele problemów klasyfikacyjnych nie ma klas liniowo separowalnych, więc warunek ten w znacznym stopniu ogranicza zastosowanie perceptronu.
Minsky i Papert wykazali, że jednym z najprostszych przykładów, który nie może być rozwiązany przy wykorzystaniu perceptronu jest problem XOR . Przykłady problemów nieseparowalnych liniowo pokazane są na Rys.5.



Rys.5. Przykłady problemów nieseparowalnych liniowo.

W przestrzeni trójwymiarowej płaszczyzna jest obiektem dwuwymiarowym. Pojedyncza płaszczyzna dzieli trójwymiarową przestrzeń na dwa oddzielne regiony. Dwie płaszczyzny mogą dawać w wyniku trzy lub cztery oddzielne regiony. Można sobie wyobrazić przez analogię przestrzeń *n*-wymiarową i obiekty będące hiperpłaszczyznami o wymiarze *n-1*. Również w tym przypadku hiperpłaszczyzny dzielą przestrzeń na właściwe odrębne regiony.
Wiele rzeczywistych problemów wymaga separacji regionów w wielowymiarowej przestrzeni, z którymi związane są odrębne klasy. Należy więc zrobić takie rozróżnienie i znaleźć takie hiperpłaszczyzny, które dzielą przestrzeń na właściwe regiony. Jak zobaczymy pewne sieci mogą się uczyć właściwego podziału. Dla przykładu można pokazać wykorzystując pojęcie płaszczyzn, że problem AND jest liniowo separowalny i da się rozwiązać za pomocą perceptronu

Zdolność klasyfikowania wzorców osiąga perceptron w trakcie uczenia, czyli poszukiwania wartości wag, przy których model dokonuje właściwej klasyfikacji. Uczenie perceptronu odbywa się w sposób nadzorowany (ang. supervised). Oznacza

to, że wzorce (uczące/testowe) muszą być przygotowane w postaci par: wzorzec wejściowy, odpowiadająca mu etykieta klasy, czyli $<\mathbf{x}_i,y_i>$. Rosenblatt – twórca perceptronu prostego, był w stanie wykazać, że jeżeli klasyfikacja może być wyuczona w opisanej dalej procedurze, to gwarantuje ona, że zostanie osiągnięta w skończonej liczbie kroków.

Model uczony jest na zbiorze uczącym, a jego zdolności do klasyfikacji sprawdzane są na zbiorze testowym, dlatego cały zbiór danych $D=\{<\mathbf{x}_1,d_1>,<\mathbf{x}_2,d_2>,\ldots,<\mathbf{x}_v,d_v>\}$ należy podzielić na zbiór uczący i zbiór testowy (Dla bardziej złożonych sieci na zbiór uczący, walidujący i testowy).

Przyjmijmy, że będziemy rozważać problemy liniowo separowalne. Uczenie będzie polegało na przyrostowej zmianie *i*-tej wagi po podaniu każdego wzorca uczącego zgodnie z wzorem (6)

$$w_{i,k}^{t+1} = w_{i,k}^{t} + \alpha \Delta w_{i,k} \tag{6}$$

α jest współczynnikiem uczenia i określa jak szybko będzie zmieniała się waga. Przyrost wagi Δw_i jest proporcjonalny do iloczynu błędu jaki pojawił się na wyjściu i sygnału płynącego po danym połączeniu. Błąd definiowany jest jako różnica między pożądanym sygnałem wyjściowym (etykietą d_k) a otrzymanym sygnałem y_k po przetworzeniu *k*-tego wzorca wejściowego $\mathbf{x}_k$..

$$\Delta w_{i,k} = (d_k - y_k) x_{i,k} \tag{7}$$

Jakie wartości może przyjmować błąd dla perceptronu prostego? Rozważmy funkcję aktywacji unipolarną

$$\delta = (d_k - y_k) = \begin{cases} 1 \; gdy \; d_k = 1 \; a \; y_k = 0 \\ -1 \; gdy \; d_k = 0 \; a \; y_k = 1 \\ 0 \qquad gdy \; d_k = y_k \end{cases} \tag{8}$$

Błąd przyjmuje trzy wartości dyskretne {-1,0,1}. A jakie wartości przyjmuje błąd dla funkcji bipolarnej?

$$\delta = (d_k - y_k) = \begin{cases} 2 \; gdy \; d_k = 1 \; a \; y_k = -1 \\ -2 \; gdy \; d_k = -1 \; a \; y_k = 1 \\ 0 \qquad gdy \; d_k = y_k \end{cases} \tag{9}$$

Widzimy, że w tym przypadku błąd przyjmuje wartości ze zbioru {-2,0,2}.

Algorytm uczenia dla perceptronu prostego:

1. Przyjmij początkowe wagi jako małe wartości losowe, bliskie 0. Ustaw wartość progu.
2. Ustaw wartość współczynnika uczenia.
3. Dla każdego przykładu $\mathbf{x}_k$ ze zbioru uczącego D wykonaj następujące kroki
4. Oblicz całkowite pobudzenie

$$z_k = \sum_{i=1}^{n} w_{i,k} \, x_{i,k}$$

5. Oblicz aktualne wyjście z perceptronu, dla funkcji unipolarnej:

6. $f(z_k) = \begin{cases} 1 \; jeśli \; z > \theta \\ 0 \; w \; przeciwnym \; przypadku \end{cases}$

7. lub dla funkcji bipolarnej:
$$f(z_k) = \begin{cases} 1 \; jeśli \; z > \theta \\ -1 \;\; w \; przeciwnym \; przypadku \end{cases}$$
8. Oblicz błąd $\delta = (d_k - y_k)$
9. Uaktualnij wagi
$w_{i,k}^{t+1} = w_{i,k}^{t} + \alpha \Delta w_{i,k}$;
$$\Delta w_{i,k} = (d_k - y_k) x_{i,k}$$

10. Powtarzaj kroki 1-9 tak długo, aż w danej epoce ani razu nie wystąpi błąd

Możemy też uczyć perceptron na przykład stosując zasadę uczenia Hebba. W takim przypadku przyrost wagi jest proporcjonalny do iloczynu sygnału wejściowego i pożądanego sygnału wyjściowego d_k. W szczególności przyjmujemy, ze przyrost zmiany wag jest równy:

$$\Delta w_{ik} = \begin{cases} x_{i,k} d_{,k} \; jesli \; y_k \neq d_k \\ 0 \; w \; przeciwnym \; przypadku \end{cases}$$

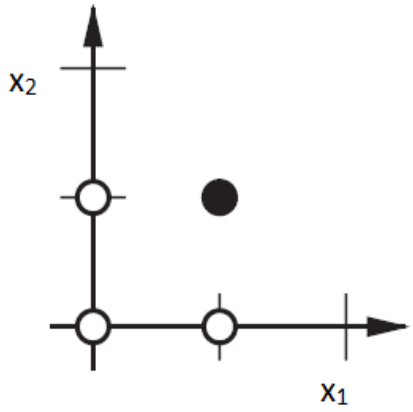
Możemy również popatrzeć na działanie perceptronu prostego przez pryzmat operacji wektorowych/macierzowych. W takim przypadku podawany na wejście wzorzec uczący jest wektorem $\mathbf{x}$ a wagi przypisane do połączeń to wektor $\mathbf{w}_k$

Jeżeli neuron ma n wejść to wektor wejściowy możemy opisać jako $\mathbf{x} = [x_1, x_2, ..., x_n]^t$ gdzie *t* - oznacza transpozycję. W dalszej notacji **x** będzie oznaczało wektor będący kolumną. Odpowiednio $\mathbf{x}^t$ oznacza wektor będący wierszem.

Całkowite pobudzenie może być opisane jako iloczyn skalarny wektora wag i wektora wejściowego. Dla wektorów o jednakowych wymiarach iloczyn skalarny (zapisywany za pomocą specjalnego operatora *) jest zdefiniowany jako suma iloczynów odpowiadających sobie składowych dwóch wektorów. A więc równość:
może być zapisana jako: z = **x** * **w.** Z formalnego punktu widzenia wygodniej jest zastąpić iloczyn skalarny przez zwykły iloczyn transponowanego wektora **x** i wektora **w** , tj. $z = \mathbf{x}^t \mathbf{w}$

Zadanie do realizacji

Należy zaprojektować i zaimplementować perceptron prosty symulujący działanie funkcji AND dla dwóch zmiennych. W tym przypadku mamy dwie klasy (klasa odpowiedzi =1 i klasa =0 zakładając logikę unipolarną) Jest to zwizualizowane na Rys.6. Ciemne kółko oznacza odpowiedź 1, jasne 0.



Rys.6. Funkcja AND

Dla funkcji AND możemy w dwóch wymiarach możemy skonstruować 4 wzorce uczące:

$\{<\mathbf{x}_1=\begin{bmatrix}0\\0\end{bmatrix}, d_1 = 0>, <\mathbf{x}_2=\begin{bmatrix}0\\1\end{bmatrix}, d_2 = 0>, <\mathbf{x}_3=\begin{bmatrix}1\\0\end{bmatrix}, d_3 = 0>, <\mathbf{x}_4=\begin{bmatrix}1\\1\end{bmatrix}, d_4 = 1>\}$

Zbiór ten jest wysoce niezrównoważony – mamy 3 wzorce klasy 0 i 1 z klasą odpowiedzi 1. W takim przypadku trudno jest wydzielić zbiór testowy. Możemy jednak powiększyć zbiór danych przez losowe dogenerowanie punktów w pobliżu każdego z tych 4 wzorców.

Należy wyuczyć perceptron prosty korzystając z algorytmu podanego w części teoretycznej.

Uwaga: W przypadku funkcji bipolarnej proszę pamiętać o zmianie wartości podawanych na wejście, tzn. 0 będzie reprezentowane przez –1, zmienią się też wartości błędów

W celu realizacji ćwiczenia należy wykonać następujące eksperymenty:

1. Ustal małe wartości początkowych wag i przyjmij wartość początkowego progu θ. Przeprowadź eksperymenty z różnymi wartościami progu obserwując liczbę epok koniecznych do wyuczenia modelu.

2.Ustal dynamiczny próg (użyj bias). Niech zakres początkowych wartości wag mieści się w przedziale -1.0 do 1.0. Prowadzić kolejne eksperymenty z uczeniem neuronu, zmieniając zakres początkowych losowych wartości wag w sposób krokowy na przykład –0.8 do +0.8; -0.5 do 0.5; -0.2 do 0.2. itd. Czy ma to wpływ na szybkość uczenia? Czy można znaleźć wartości optymalne, przy których neuron uczy się najszybciej?

3. Jaki wpływ na szybkość uczenia neuronu ma wartość współczynnika uczenia α? Wykonać badania zmieniając krokowo współczynnik uczenia.

4. Zastosować różne funkcje aktywacji neuronu:
-progową unipolarną
-progową bipolarną
Jaki wpływ na szybkość uczenia mają zastosowane funkcje aktywacji?

Wszystkie przeprowadzone badania powinny być udokumentowane w dostarczonym sprawozdaniu.

Uwaga: Wykonany program musi umożliwiać prowadzącemu przetestowanie czy po podaniu wartości wybranych przez niego (na przykład 1;0 jako sygnały wejściowe lub ich przybliżenia np. wartości 0.99 zamiast 1) neuron produkuje właściwą dla realizowanej przez niego funkcji wartość oraz sprawdzenie czy wyniki badań opisane w sprawozdaniu są otrzymane za pomocą przedstawionego programu.

Sposób oceny: podane procenty odnoszą się do połowy liczby punktów przypisanej do zadania, tj. 10 pkt

Implementacja modelu Perceptronu na zajęciach - 40% całkowitej liczby punktów przypisanej do zadania.
Przebadanie wpływu inicjalizacji wag na szybkość uczenia - 20% całkowitej liczby punktów przypisanej do zadania.
Przebadanie wpływu współczynnika uczenia na proces uczenia - 20% całkowitej liczby punktów przypisanej do zadania.
Przebadanie wpływu funkcji aktywacji - 20% całkowitej liczby punktów przypisanej do zadania.

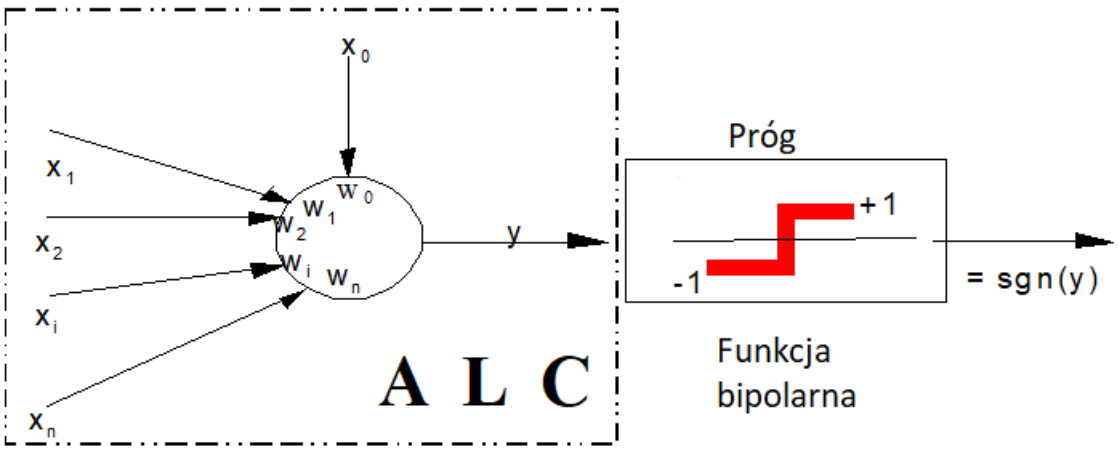
Laboratorium 2: Adaline

Cel: poznanie modelu Adaline i jego porównanie z perceptronem prostym

Wprowadzenie teoretyczne

Drugim modelem, który odegrał istotną rolę w rozwoju sieci neuronowych jest Adaline. Jego architektura jest bardzo podobna do perceptronu, jak to można zauważyć na Rys. 7. W różnych podręcznikach różnie podchodzi się do jego architektury. Czasem twierdzi się, ze Adaline obejmuje jedynie element liniowy (ALC) bez funkcji progującej, w innych tej nazwy używa się do architektury obejmującej element sumujący i funkcję

progującą. Można znaleźć zastosowania obu wersji. Pierwsza wersja okazała się bardzo przydatna w filtracji sygnałów, druga może być używana jako klasyfikator binarny.



Rys. 7. Architektura Adaline (ALC= Adaptive Linear Combiner)

Oprócz podobieństwa architektury, Adaline ma te same ograniczenia co i perceptron prosty, a więc nadaje się do rozwiązywania problemów separowalnych liniowo.

To co zdecydowanie różni oba elementy to reguła uczenia. Adaline uczony jest regułą LMS (Least Mean Square). Jest to również reguła uczenia nadzorowanego, a więc do uczenia tego modelu potrzebny zbiór wzorców w postaci par:

$$D=\{<\mathbf{x}_1,d_1>,<\mathbf{x}_2,d_2>,\ldots,<\mathbf{x}_v,d_v>\}$$

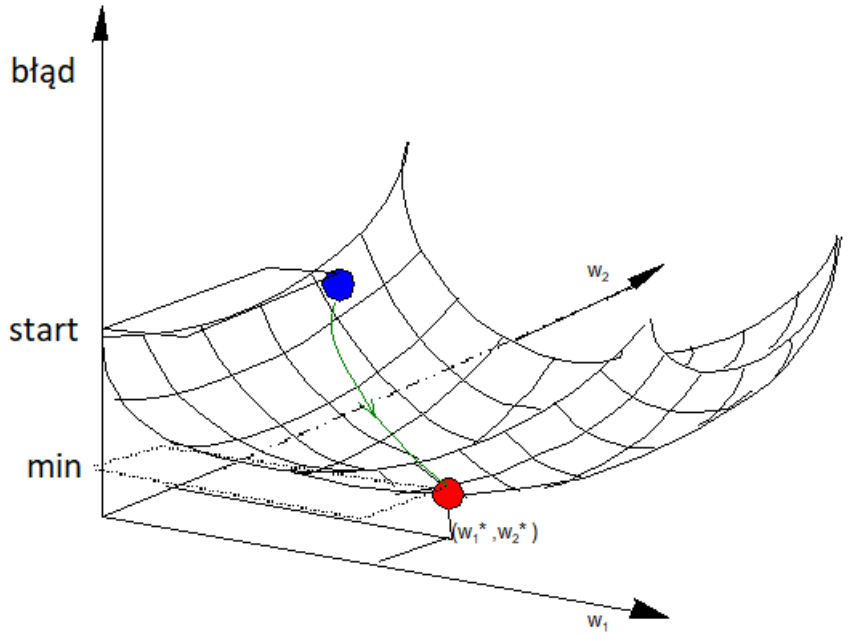
Próbujemy znaleźć najlepszy wektor wag **w***, który odwzorowuje wektor wejściowy $\mathbf{x}_k$ w zadane wyjście d_k..

Oznaczmy $\varepsilon_k = d_k - y_k$ jako błąd występujący po zastosowaniu *k*-tego wektora wejściowego. Naszym celem jest zminimalizować błąd średniokwadratowy dla zbioru wektorów wejściowych:

$$< \varepsilon_k^2 >=< (d_k - \boldsymbol{w}^t \boldsymbol{x}_k)^2 >$$

$$< \boldsymbol{\varepsilon}_k^{\,2} >= \frac{1}{L}\sum_{k=1}^{L} \boldsymbol{\varepsilon}_k^{\,2}$$

Gdzie L- jest liczbą wzorców wejściowych w zbiorze uczącym.
Zauważmy, że błąd liczony jest między **zadanym wyjściem** a **całkowitym pobudzeniem neuron** (po wyjściu z sumatora), bez użycia funkcji progującej.
Poszukiwanie optymalnych wag jest procesem iteracyjnym. Ten proces pokazany jest na Rys.8. Rozpoczyna się od wybrania arbitralnych wag początkowych i od tego punktu wyznaczamy kierunek, najszybszego spadku błędu. Zmieniamy wagi o niewielką wartość, tak aby nowy wektor leżał niżej na wykresie. Cały proces powtarzamy tak długo dopóki nie zostanie osiągnięte minimum błędu.



Rys.8. Wizualizacja poszukiwania optymalnych wag w metodzie LMS.

Poszukiwanie kierunku najszybszego spadku sprowadza się do policzenia gradientu na tej powierzchni.

Wagi będą zmieniały się zgodnie ze wzorem:

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \mu\nabla\varepsilon(\boldsymbol{w}(t)) \qquad (10)$$

Tzn. przyrost wag jest proporcjonalny do gradientu błędu a kierunek powinien być przeciwny do gradientu. W tym wzorze μ jest współczynnikiem uczenia.

Widrow i Hoff zaproponowali obliczenie gradientu $\nabla\varepsilon(w(t)$ na podstawie kwadratu błędu wyznaczonego w danym kroku iteracji dla danego wzorca ze zbioru uczącego . tzn. $\delta^2(t) = (d_k - \boldsymbol{w^t}\mathbf{x})^2$

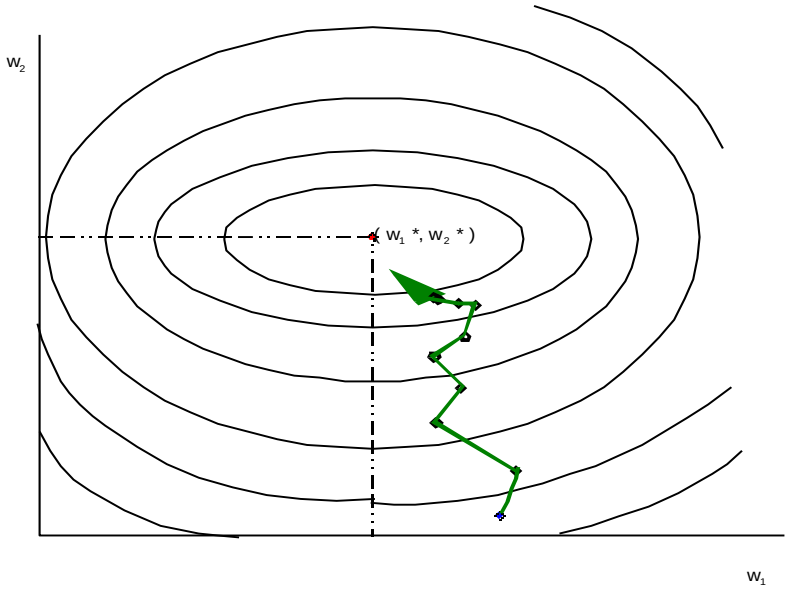
Po zróżniczkowaniu otrzymujemy:

$\nabla\varepsilon(w(t) = -2\delta(t)\boldsymbol{x}$, stąd uaktualniony wektor wag wyraża się wzorem

$$\boldsymbol{w}\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \mu\delta\boldsymbol{x}(t) \qquad (11)$$

Występująca w gradiencie liczba 2 została uwzględniona we współczynniku uczenia.

Rys. 9. pokazuje w jaki sposób odbywa się przejście od wylosowanego początkowego punktu w przestrzeni wag do punktu końcowego, który odpowiada minimum błędu (z założoną dokładnością). Widać, że pokonywana droga nie jest gładka. Gradient aproksymowany w danym punkcie jest prostopadły do konturu wykresu błędu w tym punkcie. Kierunek nie zawsze wskazuje na punkt odpowiadający minimum (mamy do czynienia z paraboloidą). W wyniku otrzymujemy drogę w kierunku do minimum funkcji błędu w postaci odcinków.



Rys. 10. Przecięcia przez wykres błędu (poziomice). Pokazany jest sposób dojścia do min błędu.

Algorytm uczenia LMS

1.Określ dopuszczalny błąd
2.Przetwórz w ALC wektor $\mathbf{x}_k$.
3.Określ wartość kwadratu błędu wykorzystując bieżącą wartość wektora wag.

$$\delta = (d_k - \boldsymbol{w}^t \boldsymbol{x}_k)^2$$

4.Uaktualinić każdą składową wektora wag zgodnie:

$$w_i(t+1)=w_i(t)+2\mu\ \delta\ x_i$$

5.Powtarzać kroki od 1 do 4 aż wartość błędu średniokwadratowego będzie akceptowalna, tzn. mniejsza od dopuszczalnego błędu.

Zadanie do realizacji
Należy zaprojektować i zaimplementować Adaline symulujący działanie funkcji AND dla dwóch zmiennych. Proszę użyć funkcji bipolarnej jako funkcji aktywacji. Należy wyuczyć model korzystając z algorytmu podanego wyżej.

W celu realizacji zadania należy wykonać następujące eksperymenty:

1. Przyjąć dynamiczny próg (użyć bias). Niech zakres początkowych wartości wag mieści się w przedziale -1.0 do 1.0. Prowadzić kolejne eksperymenty z uczeniem neuronu, zmieniając zakres początkowych losowych wartości wag w sposób krokowy na przykład –0.8 do +0.8; -0.5 do 0.5; -0.2 do 0.2. itd. Czy ma to wpływ na szybkość uczenia? Czy można znaleźć wartości optymalne, przy których neuron uczy się najszybciej?

2. Jaki wpływ na szybkość uczenia neuronu ma wartość współczynnika uczeniaμ? Wykonać badania zmieniając krokowo współczynnik uczenia.

3. Jaki jest wpływ przyjętej wartości dopuszczalnego błędu.

4. Porównać uzyskane wyniki dla obu modeli z ćwiczenia 1(Perceptron i Adaline)

Wszystkie przeprowadzone badania powinny być udokumentowane w dostarczonym sprawozdaniu. Ponadto liczba powtórzeń w każdym eksperymencie dla Perceptronu i dla Adaline powinna być jednakowa.

Sposób oceny: następujące elementy brane są przy ocenie (suma za całość -10 punktów)

Implementacja modelu Adaline na zajęciach - 20% całkowitej liczby punktów przypisanej do zadania.
Przebadanie wpływu inicjalizacji wag na szybkość uczenia - 20% całkowitej liczby punktów przypisanej do zadania.
Przebadanie wpływu współczynnika uczenia na proces uczenia - 20% całkowitej liczby punktów przypisanej do zadania.
Przebadanie wpływu funkcji aktywacji - 20% całkowitej liczby punktów przypisanej do zadania.
Porównanie wyników działania Perceptronu i Adaline - 20% całkowitej liczby punktów przypisanej do zadania.

Przesłanie raportu do prowadzącego do dnia 14 października 2020