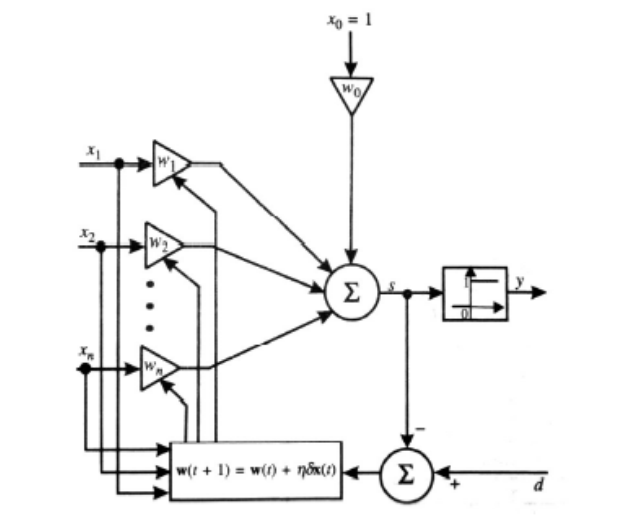
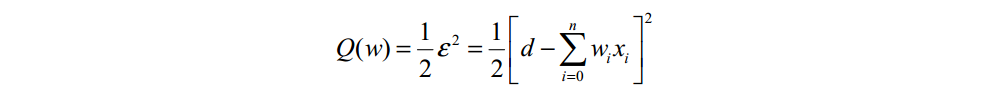
**PSI SCENARIUSZ 3**

**Anna Ruchała, IS III, GR 3**

Schemat neuronu Adaline przedstawiono na poniższym rysunku. Budowa tego neuronu jest  
bardzo podobna do modelu perceptronu, a jedyna różnica dotyczy algorytmu uczenia. Sygnał  
wyznacza się w ten sam sposób, co w przypadku uczenia perceptronu. Jednak w przypadku  
neuronu typu Adaline porównuje się sygnał wzorcowy d z sygnałem s, na wyjściu części  
liniowej neuronu (nie uwzględnia się funkcji aktywacji).



Różnicę między sygnałem wzorcowym, a sygnałem s nazywamy błędem ε = d - s . Uczenie  
neuronu, czyli dobór wag, sprowadza się do minimalizacji funkcji określonej w sposób  
następujący:



Miarę błędu określa się mianem *błędu średniego kwadratowego*.

Algorytm uczenia neuronu typu Adaline w postaci schematu blokowego:

1. Start.
2. Losowy dobór wag.
3. Wprowadź η.
4. t=1.
5. Podaj wektor x(t) na wejście neuronu oraz wczytaj wartość wzorcową d(t).
6. Wyznacz sygnał s(t) na wyjściu części liniowej neuronu.
7. Zmodyfikuj wagi zgodnie ze wzorem w(t+1) = w(t) + ηδx(t), dzie δ = d(t) – s(t).
8. t=t+1.
9. Jeśli nie cała epoka wróć do punktu 5.
10. Oblicz błąd dla całej epoki.
11. Jeśli błąd jest większy od założonego progu wróć do punktu 4.
12. Stop.

Oznaczenia:

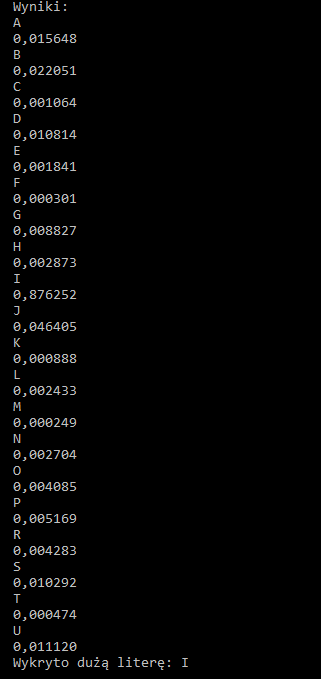
* *i*-numer wagi neuronu,
* *t*-numer iteracji w epoce,
* *d*-sygnał wzorcowy,
* *y*-sygnał wyjściowy neuronu,
* *s*-sygnał wyjściowy sumatora neuronu,
* *x*-wartość wejściowa neuronu,
* η - współczynnik uczenia (0,1).

Celem​ ​ćwiczenia ​ ​jest ​ ​poznanie ​ ​budowy​ ​i ​ ​działania ​ ​wielowarstwowych ​ ​sieci ​ ​neuronowych poprzez​ ​uczenie ​ ​z​ ​użyciem​ ​algorytmu ​ ​wstecznej ​ ​propagacji ​ ​błędu ​ ​rozpoznawania konkretnych ​ ​liter​ ​alfabetu. Wygenerowanie ​ ​danych ​ ​uczących ​ ​i ​ ​testujących, ​ ​zawierających ​ ​20 ​ ​dużych ​ ​liter dowolnie ​ ​wybranego ​ ​alfabetu.









Wnioski:

Sieć składał się z pojedynczych elementów Adaline (ang. Adaptive linear element), które powielone oraz połączone dały układ Madaline (ang. Many Adaline). Widać, iż nauczona sieć bardzo poprawnie rozpoznaje prezentowane wzory, nawet zniekształcone, ale także dostrzega w nich i inne litery. Na tej podstawie można dokładnie badać jakie cechy liter zostały uznane przez siec za istotne dla opisu litery. Zwiększenie precyzji rozpoznawania liter czy też szerzej - znaków i symboli opartych na matrycy prezentowanej sieci można uzyskać poprzez np. powiększenie jej rozmiarów i jednocześnie odpowiednio definiując okno dyskryminatora filtrującego wyniki. Korekta wektora jest tym silniejsza, im większy był odnotowany błąd. Trzeba silnie interweniować przy dużych błędach i dokonywać płynnego i subtelnego dostrajania w przypadku błędów małych.

Przy okazji zapewniony jest warunek, że w przypadku braku błędu, żadne korekty nie będą dokonane. I-ta składowa wektora będzie tym silniej zmieniana w wyniku procesu uczenia, im większa była odpowiadająca jej składowa wektora wejściowego. W przypadku ujemnych wartości odpowiednie składowe wektora będą zmniejszane a nie powiększane. Oznacza to, że w odróżnieniu od wejść „pobudzających”, które dla uzyskania większego sygnały trzeba wzmocnić, wejścia „hamujące” trzeba osłabić.

Dla dużych sieci i ciągów uczących składających się z wielu tysięcy wektorów uczących ilość obliczeń wykonywanych podczas całego cyklu uczenia jest gigantyczna a więc i czasochłonna. Nie zdarza się także aby sieć została dobrze zbudowana od razu. Zawsze jest ona efektem wielu prób i błędów. Ponadto nigdy nie mamy gwarancji, że nawet prawidłowa sieć nie utknie w minimum lokalnym podczas gdy interesuje nas znalezienie minimum globalnego. Dlatego algorytmy realizujące SSN wyposaża się mechanizmy dające nauczycielowi możliwość regulacji szybkości i jakości uczenia. Są to w współczynniki: uczenia i momentum. Wpływają one na stromość funkcji aktywacji i regulują szybkość wpływu zmiany wag na proces uczenia.