Piotr Smuga gr2

Podstawy sztucznej inteligencji

Sprawozdanie nr 4

Cel: Celem ćwiczenia jest poznanie działania reguły Hebba dla sieci jednowarstwowej na przykładzie grupowania liter alfabetu.

W ramach projektu wykonałem następujące czynności:

a) Wygenerowałem dane uczące i testujące, zawierające 20 dużych liter alfabetu angielskiego w postaci dwuwymiarowej tablicy 5x7 pikseli dla jednej litery.

b) Zaimplementowałem jednowarstwową sieć oraz zastosowałem regułę Hebba z i bez współczynnika zapominania.

c) Przeprowadziłem uczenie sieci dla różnych współczynników uczenia i zapominania.

d) Przeprowadziłem testowanie sieci.

Reguła Hebba - Jest to jedna z najpopularniejszych metod samo uczenia sieci neuronowych. Polega ona na tym, że sieci pokazuje się kolejne przykłady sygnałów wejściowych, nie podając żadnych informacji o tym, co z tymi sygnałami należy zrobić. Sieć obserwuje otoczenie i odbiera różne sygnały, nikt nie określa jednak, jakie znaczenie mają pokazujące się obiekty i jakie są pomiędzy nimi zależności. Sieć na podstawie obserwacji występujących sygnałów stopniowo sama odkrywa, jakie jest ich znaczenie i również sama ustala zachodzące między sygnałami zależności.

Po podaniu do sieci neuronowej każdego kolejnego zestawu sygnałów wejściowych tworzy się w tej sieci pewien rozkład sygnałów wyjściowych - niektóre neurony sieci są pobudzone bardzo silnie, inne słabiej, a jeszcze inne mają sygnały wyjściowe wręcz ujemne. Interpretacja tych zachowań może być taka, że niektóre neurony „rozpoznają” podawane sygnały jako „własne” (czyli takie, które są skłonne akceptować), inne traktują je „obojętnie”, zaś jeszcze u innych neuronów wzbudzają one wręcz „awersję”. Po ustaleniu się sygnałów wyjściowych wszystkich neuronów w całej sieci - wszystkie wagi wszystkich neuronów są zmieniane, przy czym wielkość odpowiedniej zmiany wyznaczana jest na podstawie iloczynu sygnału wejściowego, wchodzącego na dane wejście (to którego wagę zmieniamy) i sygnału wyjściowego produkowanego przez neuron, w którym modyfikujemy wagi. Łatwo zauważyć, że jest to właśnie realizacja postulatu Hebba - w efekcie opisanego wyżej algorytmu połączenia między źródłami silnych sygnałów i neuronami które na nie silnie reagują są wzmacniane.



Figure 1 Reguła uczenia Hebba bez nauczyciela

Sygnałem uczącym jest sygnał wyjściowy neuronu:



Korekta wektora wag:





Z zasady tej wynika, iż dodatnia wartość składnika korelacyjnego yixj powoduje wzrost wagi wij, czyli silniejszą reakcję neuronu przy kolejnej prezentacji tego samego obrazu wejściowego. Obrazy wejściowe, które często się powtarzają dają zatem silniejszą odpowiedź na wyjściu.

Regułę uczenia Hebba nazywa się także także uczeniem korelacyjnym. Celem jest takie dopasowanie wag, aby uzyskać najlepszą korelację między sygnałami wejściowymi, a zapamiętanym w formie wartości wag wzorcem, na który określony neuron ma zareagować.

Uczenie neuronu z zastosowaniem reguły Hebba może być prowadzone również z nauczycielem. W uczeniu z nauczycielem wartość sygnału wyjściowego a zastępuje się wartością zadaną t.



- W programie użyłem funkcję jako funkcję aktywacji

- każda litera reprezentowana jest jako tablica zawierająca 5x5 liczb reprezentującą piksele

- dane uczące składają się z tablicy 20 liczb reprezentujących 20 pierwszych wielkich liter alfabetu angielskiego:

{0,1,1,1,0,1,0,0,0,1,1,0,0,0,1,1,1,1,1,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1},

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

public static final int[][] learning\_letters = {

{0,1,1,1,0,1,0,0,0,1,1,0,0,0,1,1,1,1,1,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1},

{1,1,1,1,0,1,0,0,0,1,1,0,0,0,1,1,1,1,1,0,1,0,0,0,1,1,0,0,0,1,1,1,1,1,0},

{0,1,1,1,0,1,0,0,0,1,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,1,1,1,0},

{1,1,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,1,1,1,0},

{1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,1,1,1,0,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1},

{1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,1,1,1,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0},

{0,1,1,1,0,1,0,0,0,1,1,0,0,0,0,1,0,1,1,1,1,0,0,0,1,1,0,0,0,1,0,1,1,1,0},

{1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,1,1,1,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1},

{0,1,1,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,1,0},

{1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,0,0,0,1,0,1,1,1,0},

{1,0,0,0,1,1,1,0,1,1,1,0,1,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1},

{1,0,0,0,1,1,0,0,0,1,1,1,0,0,1,1,0,1,0,1,1,0,0,1,1,1,0,0,0,1,1,0,0,0,1},

{0,1,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,0,1,1,1,0},

{1,1,1,1,0,1,0,0,0,1,1,0,0,0,1,1,1,1,1,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0},

{1,1,1,1,0,1,0,0,0,1,1,0,0,0,1,1,1,1,1,0,1,0,1,0,0,1,0,0,1,0,1,0,0,0,1},

{0,1,1,1,0,1,0,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0,1,1,0,0,0,1,0,1,1,1,0},

{1,1,1,1,1,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0},

{1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,0,1,1,1,0},

{1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,1,0,1,1,0,1,0,1,0,1,0,1,0},

{1,0,0,0,1,1,0,0,0,1,0,1,0,1,0,0,0,1,0,0,0,1,0,1,0,1,0,0,0,1,1,0,0,0,1}

};

Wyniki:

A-> Result: 4.681647738846465E-6

B-> Result: 5.006119476889349E-6

C-> Result: 3.188923881129846E-6

D-> Result: 4.68918395650634E-6

E-> Result: 4.565677656212667E-6

F-> Result: 3.7930809878858765E-6

G-> Result: 3.9457664153459226E-6

H-> Result: 5.870339441187086E-6

I-> Result: 1.081617541696061E-6

J-> Result: 3.7962622189382024E-6

M-> Result: 5.411661398571329E-6

N-> Result: 6.772197576645443E-6

O-> Result: 3.6310848259815644E-6

P-> Result: 3.937083359181377E-6

R-> Result: 4.526432339499428E-6

S-> Result: 3.5068469831844506E-6

T-> Result: 1.9944112410075915E-6

U-> Result: 4.819776528324662E-6

W-> Result: 5.068599641788962E-6

X-> Result: 6.792973447220169E-6

Figure 1 Wyniki dla poszczególnych liter przy współczynniku uczenia rónym 0.1 i współczynniku zapominania równym 0.1

Figure 2 Wyniki dla poszczególnych liter przy współczynniku uczenia równym 0.2 i współczynniku zapominania 0.1

Figure 3 Wyniki dla poszczególnych liter przy współczynniku uczenia równym 0.1 i współczynniku zapominania równym 0.2

Figure 4 Wyniki dla poszczególnych liter przy współczynniku uczenia równym 0.1 i współczynniku zapominania równym 0

Wnioski:

* Uczenie sieci wypada gorzej, gdy nie stosuje się współczynnika zapominania, ponieważ wagi rosną wtedy bardzo szybko i uniemożliwiona jest ich stabilizacja.
* Im mniejszy współczynnik uczenia tym dokładniejszy wynik. Aby otrzymać wiarygodne dane musimy odpowiednio dobrać współczynniki uczenia a także bezwładności
* wraz ze zwiększaniem wartości współczynnika uczenia zmniejsza się liczba iteracji, co ma wpływ na działanie programu i powoduje uzyskanie błędnych wyników. Aby zredukować liczbę błędnych wyników można zwiększyć liczbę iteracji.
* dobór wag ma bezpośredni wpływ na działanie perceptronów w sieci, w zależności od ich wartości zmienia się poprawność wyników i liczba iteracji. Wartości wag mają największy wpływ na efekt końcowy, często wagi są ustalane losowo.
* dane uczące wpływają na poprawność uczenia perceptronu, przy zbyt małej ilości otrzymujemy błędne wyniki. Aby uzyskać lepsze wyniki powinniśmy dostarczyć wystarczającą liczbę danych wejściowych
* bardzo ważny jest odpowiedni dobór współczynnika uczenia, współczynnika zapominania, wag oraz liczby danych uczących, gdyż mają bezpośredni wpływ na działanie sieci.

Listing kodu:

* Cały listing kodu został umieszony w repozytorium Git pod następującym adresem: https://github.com/psmuga/PSI/tree/master/Scenariusz%204/src