Paweł Put, Informatyka Stosowana gr. 3

Sprawozdanie nr 3.

Temat ćwiczenia: Budowa i działanie sieci wielowarstwowej.

Cel ćwiczenia: Celem ćwiczenia jest poznanie budowy i działania wielowarstwowych sieci neuronowych poprzez uczenie z użyciem algorytmu wstecznej propagacji błędu rozpoznawania konkretnych liter alfabetu.

Przebieg: Zaimplementowałem w pythonie przy użuciu biblioteki do tworzenia sieci neuronowych pybrain3 dane do nauczania sieci w postaci 20 dużych liter polskiego alfabetu w wektorach 35cio elementowych, które po podzieleniu wyglądały jak tablice 5x7. Przykład litery A:

* Wektor:

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

* Tablica utworzona z wektora:

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

* Interpretacja numeryczna wektora:

(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)

* Interpretacja numeryczna tabeli:

(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 )

Następnie utworzyłem wektor uczący 20sto elementowy, którego kolejne wyjścia miały odpowiadać kolejnym literom. Przykładowo wektor uczący dla litery A wyglądał następująco:

(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, )

Następnie utworzyłem sieć typu feedforward. Jest to sieć jednokierunkowa, w której informacje przekazywane są w jednym kierunku, od wejścia do wyjścia. Moja sieć składa się z warstwy wejściowej składającej się z 35 wejść (gdyż mamy 35 wejść w każdej literze), 20 wyjść (gdyż mamy rozpoznać 20 różnych liter) i do tego z warstwy ukrytej, w której modyfikowałem liczbę neuronów podczas ćwiczenia. Do tego połączyłem wszystkie warstwy w sposób "full connection" co oznacza, że każdy neuron z warstwy wejściowej połączony jest z każdym neuronem z warstwy ukrytej oraz że każdy neuron z warstwy ukrytej połączony jest z każdym neuronem z warstwy wyjściowej. Neurony warstwy ukrytej są sigmoidalne, co oznacza, że jego funkcja aktywacji przybiera postać funkcji sigmoidalnej, unipolarnej lub bipolarnej.

Po utworzeniu sieci zostało ją tylko wytrenować algorytmem wstecznej propagacji błędu. Jest to podstawowy algorytm uczenia nadzorowanego(z nauczycielem) wielowarstwowych sieci neuronowych. Ogólny schemat procesu trenowania sieci wygląda następująco:

1. Ustalamy topologię sieci, tzn. liczbę warstw, liczbę neuronów w warstwach.
2. Inicjujemy wagi losowo (na małe wartości).
3. Dla danego wektora uczącego obliczamy odpowiedź sieci (warstwa po warstwie).
4. Każdy neuron wyjściowy oblicza swój błąd, oparty na różnicy pomiędzy obliczoną odpowiedzią y oraz poprawną odpowiedzią t.
5. Błędy propagowane są do wcześniejszych warstw.
6. Każdy neuron (również w warstwach ukrytych) modyfikuje wagi na podstawie wartości błędu i wielkości przetwarzanych w tym kroku sygnałów.
7. Powtarzamy od punktu 3ciego dla kolejnych wektorów uczących. Gdy wszystkie wektory zostaną użyte losowo zmieniamy ich kolejność i zaczynamy wykorzystywać powtórnie.
8. Zatrzymujemy się, gdy średni błąd na danych treningowych przestanie maleć. Możemy też co jakiś czas testować sieć na specjalnej puli nieużywanych do treningu próbek testowych i kończyć trenowanie, gdy błąd przestanie maleć.

Aby znaleźć taki zestaw wag, dla którego błąd sieci jest jak najmniejszy, możemy zapisać ten błąd

Funkcję od wartości wag. Zwykle błąd liczony jest jako kwadrat odchylenia: d = 1/2 (y-t)2.

Współczynnik uczenia - przyjmuje wartość z przedziału (0, 1>. Jest to wartość, przez którą zostanie pomnożona korekta wag. W programie ustawiane jest to w pliku training.py przy funkcji BackpropTrainer argument learningrate.

Współczynnik bezwładności - powstał podczas prób zwiększenia efektywności procesu uczenia. Wykonuje wygładzanie poprawek zbioru wag. Dzięki niemu wpływ wcześniejszych poprawek wektora wag maleje wykładniczo w miarę oddalenia od bieżącego numeru iteracji. Ostateczna poprawka jest dzięki temu sumą opartą nie tylko o wielkość bieżącego gradientu funkcji celu, ale także o wcześniejsze modyfikacje. W programie ustawiane jest to w pliku training.py przy funkcji BackpropTrainer argument momentum.

Przetestuję działanie programu dla różnego zestawu danych:

1. Neuronów w warstwie ukrytej:

* 1
* 10
* 20
* 60

1. Współczynników nauczania:

* 0,01
* 0,1
* 0,5
* 0,8

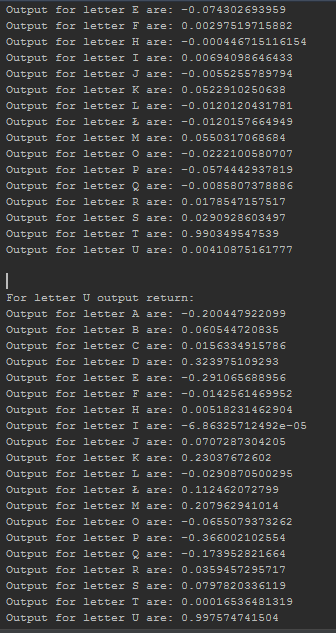
1. Współczynników bezwładności:

* 0,01
* 0,1
* 0,5
* 0,8

Jeżeli dla 20000 epok nie uda się nauczyć neuronu, to przerywamy nauczanie.

|  |  |  |  |
| --- | --- | --- | --- |
| **liczba neuronów w warstwie ukrytej** | **współczynnik nauczania** | **współczynnik bezwładności** | **liczba epok** |
| 1 | 0,1 | 0,1 | nie udało się nauczyć sieci |
| 10 | 0,1 | 0,1 | sieć nauczyła się liter: I. J, L, Ł, T |
| 20 | 0,1 | 0,1 | 4000 |
| 60 | 0,1 | 0,1 | 1400 |
| 100 | 0,1 | 0,1 | 3600 |
| 60 | 0,01 | 0,1 | 2500 |
| 60 | 0,1 | 0,1 | 1400 |
| 60 | 0,15 | 0,1 | nie udało się nauczyć sieci |
| 60 | 0,2 | 0,1 | nie udało się nauczyć sieci |
| 60 | 0,5 | 0,1 | nie udało się nauczyć sieci |
| 60 | 0,8 | 0,1 | nie udało się nauczyć sieci |
| 60 | 0,1 | 0,01 | 1200 |
| 60 | 0,1 | 0,1 | 1400 |
| 60 | 0,1 | 0,5 | 6000 |
| 60 | 0,1 | 0,8 | nie udało się nauczyć sieci |

Przykładowy screen z wynikami



Omówienie wyników:

* Neuronów w środkowej części nie może być ani za mało, ani za dużo, co wydaje się z logicznego punktu widzenia oczywiste. Jeżeli mamy nasze informacje zależą od zbyt wielu neuronów to mogą mieć problem z ustawieniem odpowiednim wektora wyjściowego, natomiast jeżeli będzie ich za mało to mogą nie podołać ogromowi realizowanego problemu. W naszym zadaniu wyszło, że gdy mamy 20 i 100 neuronów jest to gorsze niż 60.
* Im mniejszy współczynnik bezwładności tym szybciej uczy się sieć
* Współczynnik uczenia może być dla tej sieci maksymalnie 0,1. Powyżej tego sieć nie może nauczyć się działania w ciągu 20000 epok.

Szukanie odpowiedniej ilości neuronów w ukrytej powłoce: Aby to uczynić należy znaleźć liczbę neuronów z przedziału 20, a 100 w ukrytej powłoce, dla której sieć nauczy się szybciej niż w 1400 epok. Metoda, której użyjemy to metoda prób i błędów:

* Dla 40 - sieć rozpoznała poprawnie litery: A, C, D, E, F, H, I, K, L, Ł, R, S, T, reszta liter była blisko
* Dla 50 - sieć działa prawidłowo dla 1300 epok więc w okolicach 50 jest optymalna liczba neuronów w ukrytej warstwie.

Wykres ilości neuronów w warstwie ukrytej do ilości epok na naukę sieci:

Wnioski:

* Sieć wielowarstwowa idealnie rozwiązuje trudniejsze zagadnienia ze sztucznej inteligencji. Wystarczy jedna dodatkowa warstwa neuronów a sieć zyskuje o wiele więcej możliwości.
* Należy dokładnie przygotować się do tego, aby zbudować sieć o optymalnej ilości neuronów, gdyż jak jest ich za mało to sieć się niczego nie nauczy, a jak za dużo to trwa to za długo.
* Należy odnaleźć optymalne wartości współczynników nauczania i bezwładności dla naszej sieci. Gdy nie zrobimy tego odpowiednio, to nasza sieć będzie uczyła się wiekami, albo może nawet przez nieodpowiednie współczynniki nie nauczyć się tego, czego od niej wymagamy, pomimo idealnej budowy.

Bibliografia:

* http://pracownik.kul.pl/files/31717/public/Model\_neuronu\_sigmoidalnego.pdf
* http://sasha.sector-alpha.net/~ptsnoop/bitsnoop/pybrain/docs/documentation.pdf
* http://pracownik.kul.pl/files/31717/public/Model\_neuronu\_sigmoidalnego.pdf
* https://pl.wikipedia.org/wiki/Propagacja\_wsteczna
* http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad3/w3.htm
* http://sknbo.ue.poznan.pl/neuro/ssn/pliki/uczenie/uczenie1.html