Sprawozdanie – **Podstawy Sztucznej Inteligencji,** rok 3 IS

**Michał Ligęza**, gr. 2

Zadanie nr. 2 **- Budowa​ ​ i​ ​ działanie​ ​ sieci​ ​ jednowarstwowej**

Cel​ ​ ćwiczenia - **Celem​ ​ ćwiczenia​ ​ jest​ ​ poznanie​ ​ budowy​ ​ i​ ​ działania​ ​ jednowarstwowych​ ​ sieci​ ​ neuronowych​ ​ oraz​ ​ uczenie​ ​ rozpoznawania​ ​ wielkości​ ​ liter.**

1. **Opis budowy sieci oraz użytych algorytmów uczenia** Do przygotowania ćwiczenia użyto przygotowanego wcześniej zestawu liter i znaków o wymiarach 8x18, dlatego każdy użyty neuron posiada 144 wejścia. Przygotowana sieć jednowarstwowa posiada 20 wejść – czyli ilość liter które sieć ma się nauczyć, tj. 10 liter małych od a do j, oraz 10 liter dużych od A do J. Obie sieci są jednowarstwowe i składają się z tej samej ilości neuronów. Różnica miedzy nimi występuje w algorytmie nauczania – jedna z nich jest oparta na nauczaniu perceptronu, druga natomiast rozwinięta została z pierwszego rodzaju do formy nauczania z momentum.  
     
   Dla pierwszej sieci zmianę wag przedstawia wzór:  
     
   **Nowa Waga[i]= Poprzednia Waga + Współczynnik Nauki \* wejście[i]\*błąd Przewidywania**  
     
   natomiast dla sieci z momentum:

**Nowa Waga[i]= Poprzednia Waga + Współczynnik Nauki \* wejście[i]\*błąd Przewidywania + Poprzednia Zmiana Wagi [i]\* Współczynnik Momentum**

Sieć z momentum jest dobrze dopasowana do przyspieszenia nauki według tendencji danych, natomiast do zmiennych danych, takich jak litery, predyspozycje takiego uczenia zostaną sprawdzone w ramach laboratorium.

1. **Wyniki**Wyniki w serii 10 losowych sprawdzeń: ilość rozpoznanych liter/ilość błędnie rozpoznanych liter/ilość liter, ilość rozpoznanych liter/ilość znaków, np.: litera B rozpoznana jako B i D daje 1/1/1,

litera A rozpoznana jako A = 2/1/2, 0/0

litera E rozpoznana jako E i F=3/2/3,0/0

znak # nierozpoznany jako litera = 3/2/3, 0/1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ilość | Algorytm prostego uczenia | | | | Algorytm z momentum | | | |
| 50 | 2/3/6,6/3 | 1/1/8,0/2 | 5/0/7,0/3 | 1/0/4,1/6 | 1/0/9,0/1 | 3/4/7,0/3 | 3/1/6,2/4 | 3/1/8,0/2 |
| 100 | 5/0/8,0/2 | 5/2/7,1/3 | 6/3/7,1/3 | 7/1/8,0/2 | 3/5/7,0/3 | 2/0/5,0/5 | 7/1/8,0/2 | 2/2/6,2/4 |
| 250 | 8/0/9.0/1 | 6/0/6,0/4 | 6/0/7,0/3 | 2/0/3,0/7 | 6/0/6,0/4 | 4/0/4.0/6 | 5/0/5,0/5 | 4/0/8,0/2 |
| 500 | 7/0/7,0/3 | 7/0/7,0/3 | 8/0/8,0/2 | 6/0/6,0/4 | 7/0/7,0/3 | 8/0/8,0/2 | 7/0/8,0/2 | 5/0/5,0/5 |
| 1000 | 7/0/7,0/3 | 8/0/8,0/2 | 9/0/9,0/1 | 7/0/7,0/3 | 8/0/8,0/2 | 5/0/5,0/5 | 5/0/5,0/5 | 8/0/8.0/2 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ilość | Algorytm prostego uczenia | | | | Algorytm z momentum | | | |
| 50 | 0/1/7,0/3 | 0/0/8,0/2 | 0/0/6,0/4 | 0/0/5,0/5 | 1/0/7,0/3 | 0/0/7,0/3 | 1/0/7,0/3 | 0/0/6,3/4 |
| 100 | 0/0/3,2/7 | 0/0/6,0/4 | 0/1/5,0/5 | 0/0/6,0/4 | 4/2/8,1/2 | 3/0/7,0/3 | 3/1/9,0/1 | 2/0/5,0/5 |
| 250 | 4/0/8,0/2 | 5/0/7,0/3 | 1/0/5,0/5 | 3/0/7,0/3 | 3/0/5,0/5 | 7/0/7,0/3 | 7/1/8,0/2 | 3/1/5,0/5 |
| 500 | 4/0/5,0/5 | 6/0/6,0/4 | 4/0/5,0/5 | 7/1/7,0/3 | 7/0/7,0/3 | 5/0/5,0/5 | 4/1/5,0/5 | 7/0/7,0/3 |
| 1000 | 6/0/6,0/4 | 5/0/5,0/5 | 6/0/6,0/4 | 9/0/9,0/1 | 6/0/6,0/4 | 7/0/7,0/3 | 5/0/5,0/5 | 8/0/8,0/2 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ilość | Algorytm prostego uczenia | | | | Algorytm z momentum | | | |
| 50 | 0/1/2,0/8 | 0/0/5,1/5 | 3/1/7,0/3 | 2/1/5,0/5 | 1/6/8,2/2 | 3/0/8,0/2 | 2/1/5,0/5 | 2/1/7,0/3 |
| 100 | 1/0/3,1/7 | 5/3/6,0/4 | 5/3/8.2/2 | 4/2/6,1/4 | 4/2/6,0/4 | 5/5/6,2/4 | 6/2/9,0/1 | 5/1/6,0/4 |
| 250 | 7/1/9,0/1 | 6/0/7,0/3 | 7/0/7,0/3 | 7/0/7,0/3 | 4/0/5,0/5 | 5/1/8,0/2 | 3/2/6,0/4 | 3/4/3,5/7 |
| 500 | 8/0/8,0/2 | 7/0/7.0/3 | 7/0/7,1/3 | 3/0/3,0/7 | 7/4/8,0/2 | 8/0/9,0/1 | 6/0/7,0/3 | 6/0/8,0/2 |
| 1000 | 6/0/6,0/4 | 7/0/7,0/3 | 4/0/4,0/6 | 8/0/8,0/2 | 6/0/8,0/2 | 6/0/6,0/4 | 8/0/8,0/2 | 4/0/4,0/6 |

znak ! rozpoznany jako i = 3/2/3, ½  
  
Wartości: współczynnik uczenia 0.5, współczynnik momentu 0.1  
Wartości: współczynnik uczenia 0.1, współczynnik momentu 0.05  
Wartości: współczynnik uczenia 0.5, współczynnik momentu 0.3  
Wartości: współczynnik uczenia 0.7, współczynnik momentu 0.1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ilość | Algorytm prostego uczenia | | | | Algorytm z momentum | | | |
| 50 | 5/3/8,0/2 | 2/0/7,0/3 | 1/0/4,0/6 | 1/0/6,1/4 | 2/5/7,1/3 | 3/6/7,2/3 | 2/1/9,1/1 | 3/4/9,0/1 |
| 100 | 4/0/5,0/5 | 3/3/6,1/4 | 7/0/8,1/2 | 4/3/6,3/4 | 2/3/6,1/4 | 4/0/6,0/4 | 4/0/7,1/3 | 4/1/7,1/3 |
| 250 | 6/1/7,0/3 | 7/0/7,0/3 | 8/0/9,0/1 | 6/0/6,0/4 | 7/0/7,1/3 | 8/2/8,0/2 | 3/1/4,2/6 | 6/1/7,1/3 |
| 500 | 8/0/8,0/2 | 6/0/6,1/4 | 7/0/7,0/3 | 6/0/6,1/4 | 8/0/8,0/2 | 3/0/3.0/7 | 9/2/10,0/0 | 5/0/5,2/5 |
| 1000 | 6/0/6,0/4 | 4/0/4,0/6 | 7/0/7,0/3 | 9/0/10,0/0 | 6/3/8,0/2 | 6/0/6,0/4 | 6/0/6,0/4 | 6/2/8,1/2 |

**Opracowanie wyników:**

Wartości: współczynnik uczenia 0.5, współczynnik momentu 0.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ilość | Algorytm prostego uczenia | | | Algorytm z momentum | | |
|  | % dobrych odpowiedzi | % błędnych  odpowiedzi | % błędów w znakach | % dobrych odpowiedzi | % błędnych  odpowiedzi | % błędów w znakach |
| 50 | 36 | 16 | 50 | 33 | 20 | 20 |
| 100 | 76 | 20 | 20 | 54 | 30 | 14 |
| 250 | 88 | 0 | 0 | 82 | 0 | 0 |
| 500 | 100 | 0 | 0 | 96 | 0 | 0 |
| 1000 | 100 | 0 | 0 | 100 | 0 | 0 |

Wartości: współczynnik uczenia 0.1, współczynnik momentu 0.05

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ilość | Algorytm prostego uczenia | | | Algorytm z momentum | | |
|  | % dobrych odpowiedzi | % błędnych  odpowiedzi | % błędów w znakach | % dobrych odpowiedzi | % błędnych  odpowiedzi | % błędów w znakach |
| 50 | 0 | 4 | 0 | 7 | 0 | 23 |
| 100 | 0 | 6 | 10 | 41 | 10 | 9 |
| 250 | 48 | 0 | 0 | 83 | 12 | 0 |
| 500 | 91 | 4 | 0 | 96 | 4 | 0 |
| 1000 | 100 | 0 | 0 | 100 | 0 | 0 |

Wartości: współczynnik uczenia 0.5, współczynnik momentu 0.3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ilość | Algorytm prostego uczenia | | | Algorytm z momentum | | |
|  | % dobrych odpowiedzi | % błędnych  odpowiedzi | % błędów w znakach | % dobrych odpowiedzi | % błędnych  odpowiedzi | % błędów w znakach |
| 50 | 0 | 16 | 0 | 29 | 29 | 17 |
| 100 | 0 | 35 | 24 | 74 | 37 | 15 |
| 250 | 90 | 3 | 0 | 68 | 32 | 0 |
| 500 | 100 | 0 | 0 | 81 | 12 | 0 |
| 1000 | 100 | 0 | 0 | 92 | 0 | 0 |

Wartości: współczynnik uczenia 0.7, współczynnik momentu 0.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ilość | Algorytm prostego uczenia | | | Algorytm z momentum | | |
|  | % dobrych odpowiedzi | % błędnych  odpowiedzi | % błędów w znakach | % dobrych odpowiedzi | % błędnych  odpowiedzi | % błędów w znakach |
| 50 | 36 | 12 | 50 | 31 | 50 | 50 |
| 100 | 72 | 24 | 33 | 32 | 51 | 21 |
| 250 | 93 | 3 | 0 | 92 | 15 | 29 |
| 500 | 100 | 0 | 15 | 96 | 7 | 14 |
| 1000 | 96 | 0 | 0 | 86 | 18 | 8 |

1. **Analiza**  
   Przy początkowej wartości współczynnika nauki 0.5 oraz współczynnika momentum 0.1 sieć z momentum pokazała lepsze wyniki niż sieć klasyczna: szybciej nauczyła się odpowiednich liter oraz lepiej korygowała błędy. Po zmianie współczynnika momentum na 0.3, sieć klasyczna lepiej korygowała błędy. Sieć z momentum szybciej nauczyła się liter, natomiast po przekroczeniu bariery około 200 pojedynczych nauczań, sieć klasyczna zaczęła pokazywać o wiele lepsze wyniki niż sieć z momentum. Dobierając współczynniki nauki i momentum odpowiednio 0.1 oraz 0.05, mimo nieznacznie wyższych wartości błędów, dominowała sieć z momentum, dając szybciej lepsze odpowiedzi. Mimo tego 100% odpowiedzi obie sieci uzyskały w tym samym czasie. Przy współczynnikach nauki i momentum danych jako 0.7 oraz 0.1, obie sieci zwracały wiele błędów, choć to sieć klasyczna szybciej je korygowała. Sieć z momentum wolniej uczyła się odpowiednich liter, częściej się myląc, natomiast podobnie jak sieć klasyczna, podatna była na „analfabetyzm wtórny” - obie sieci potrafiły „cofnąć się” z nauką literek.
2. **Wnioski**  
   Sieć klasyczna spełniła swoje zadanie, dosyć dobrze się uczyła - choć wolniej niż sieć z momentum, była to nauka o wiele stabilniejsza i zawierała w sobie mniej błędów. Można powiedzieć że ta sieć była bardziej ostrożna w wynikach. Sieć z momentum, mimo tego iż uczyła się szybciej, podawała wiele błędnych odpowiedzi, a jej nauka mogła zostać zaprzepaszczona (np. przez następujące po siebie zestawy posiadające wiele przeciwnych danych na tych samych wejściach – sieć z momentum „głupiała” w takiej sytuacji. Do takiej formy przygotowywania nauki należy stwierdzić że z użytych algorytmów o wiele lepiej sprawdza się klasyczny algorytm nauczania, odrzucając algorytm z momentum za jego niestabilność.
3. **Listing kodu**  
   class Perceptron {

private:

double learningRate; // współczynnik nauki

vector<double> weights; // wagi na wejściach

public:

Perceptron(int numWeights, double rate); // konstruktor ustalający ilość wejść i współczynnik nauki

double guess(double inputs[144]); // metoda podająca przewydiwany wynik dla podanych danych

double train(double inputs[144], double target); //metoda szkoląca dla podanych danych i przewidywanego wyniku

};  
class Net

{

private:

double data[144]; //dane na wejściu

int ilosc=20; //ilość neuronów w sieci jednowarstwowej

char \*litery; //litery przypisane odpowiednim neuronom

double \*wyjscie; //wektor wyjścia

Perceptron \*p[20]; //neurony

public:

Net(double nauka); //konstruktor wymagający współczynnika nauki

~Net(); //destruktor

void setInput(int \*in); //ustawienie wejścia

void train(double \*wyniki); //uczenie po sanych wynikach( z osobno ustawionym wejściem)

void train(double \*data,double \*wyniki); //uczenie po danych i wynikach

void guess(double \*x); //zgaduje po danych

double \*getOutput(); //zwraca wektor wyjścia

void showOutput(); //wypisuje wektor wyjścia

void showTheOne(int i); //wypisuje wynik wyjscia wybranego neuronu

};  
class Momentowy

{

private:

double learningRate; //współczynnik nauki

double momentum; //współczynnik momentum

vector<double> weights; //wektor wag

vector<double> delta; //wektor poprzedniej zmiany wag

public:

Momentowy(int numWeights, double rate, double moment); //konstuktor ustalający ilość wejść, współczynnik nauki i momentum

~Momentowy(); //destruktor

double guess(double inputs[144]); // metoda podająca przewydiwany wynik dla podanych danych

double train(double inputs[144], double target);//metoda szkoląca dla podanych danych i przewidywanego wyniku

};  
class MomentumNet

{

private:

double data[144]; //dane na wejściu

int ilosc = 20; //ilość neuronów w sieci jednowarstwowej

char \*litery; //litery przypisane odpowiednim neuronom

double \*wyjscie; //wektor wyjścia

Momentowy \*p[20];//neurony

public:

MomentumNet(double nauka,double moment);//konstruktor wymagający współczynnika nauki i momentum

~MomentumNet();//destruktor

void setInput(int \*in);//ustawienie wejścia

void train(double \*wyniki);//uczenie po sanych wynikach( z osobno ustawionym wejściem)

void train(double \*data, double \*wyniki);//uczenie po danych i wynikach

void guess(double \*x); //zgaduje po danych

double \*getOutput(); //zwraca wektor wyjścia

void showOutput();//wypisuje wektor wyjścia

void showTheOne(int i);//wypisuje wynik wyjscia wybranego neuronu

};

double \*returnData(char s);//zwraca stworzony zestaw danych dla zadanego znaku  
double \*a = returnData((char)"a");//zapis danych dla litery a, podobnie dla innych liter

double w\_a[21]{} //zestaw wyników, jakie powinno przyjąć a, podobnie dla innych liter  
int \*check(int \*wyniki, char litera, double \*ret) //funkcja zliczająca wyniki- ilość odpowiednich wystąpień znaków, dla zadanej litery, dla zadanego wektora wyjścia.   
wyniki: 0-prawidłowe wystąpienie litery, 1-błędne wystąpienie litery dla zadanej litery, 2-suma zadanych liter, 3-ilośc pokazanych liter, gdy na wejściu była nie uczona litera/znak/ przypadkowe wejście, 4-ilość sprawdzonych nieuczonych liter, znaków, przypadkowych wejść   
void net(int ile, int test, double nauka,bool show) //funkcja tworząca sieć Net i ucząca ją ile-iteracji pojedynczej litery, testująca ją test-razy, dla zadanego współczynnika nauki, wyświetlając lub nie wewnętrzne etapy nauki  
void momentumnet(int ile, int test,double nauka, double moment,bool show) //funkcja tworząca sieć MomentumNet i ucząca ją ile-iteracji pojedynczej litery, testująca ją test-razy, dla zadanego współczynnika nauki i momentum, wyświetlając lub nie wewnętrzne etapy nauki