Sprawozdanie ze scenariusza nr 1

Cel ćwiczenia:

Celem ćwiczenia jest poznanie budowy i działania perceptronu poprzez implementację oraz uczenie perceptronu realizującego wybraną funkcję logiczną dwóch zmiennych.

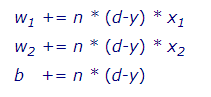
1. Syntetyczny opis budowy oraz wykorzystanego algorytmu uczenia

Perceptron w swojej budowie odwzorowuje budowę komórki nerwowej. Składa się z wejść (odwzorowania dendrytów), bloku sumacyjnego i aktywacji (odwzorowanie jądra komórki) i wyjścia (odwzorowanie aksonu).

Rysunek 1

Każde z wejść posiada swoją wagę. Blok sumacyjny odpowiada za zsumowanie iloczynów wartości wejść i wag tych wejść. Wartość wyliczona przez sumator jest przekazywana do bloku aktywacji, który wykonuje funkcję aktywacji (w przypadku perceptronu jest to funkcja progowa jak na Rysunku 1).

Perceptron może zawierać dodatkowe wejście zwane biasem o stałej wartości 1.

Mechanizm uczenia z nauczycielem polega na wprowadzaniu do neuronu wcześniej przygotowanych danych dla których znany jest właściwy wynik pracy neuronu i porównywaniu otrzymanych wyników z wynikiem prawidłowym. W razie nieprawidłowej pracy zmianie ulegają wagi poszczególnych wejść, zgodnie ze wzorem 1.

Równanie 1

W jest wagą wejścia, n - współczynnikiem uczenia, d – wartością otrzymaną na wyjściu, y – wartością oczekiwaną, x – wartością na wejściu.

1. Zestawienie otrzymanych wyników.

Dla różnej ilości danych uczących:

Ponieważ maksymlna dostępna ilość danych uczących dla funkcji AND wynosi 4 do porówania zastaną użyte następujące zestawy danych:

Zestaw 1: Zestaw 2:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X1 | X2 | Y |  | X1 | X2 | Y |
| 0 | 0 | 0 |  |  |  |  |
| 1 | 0 | 0 |  | 1 | 0 | 0 |
| 0 | 1 | 0 |  | 0 | 1 | 0 |
| 1 | 1 | 1 |  |  |  |  |

Wynik uczenia perceptronu dla zestawu 1:

Wynik uczenia perceptronu dla danych z zestawu 2:

Dla różnej wartości współczynnika uczenia:

Poniżej wynik uczenia perceptronu dla zestawu 1 ze współczynnikiem uczenia N = 0,1:

Poniżej wynik uczenia perceptronu dla zestawu 1 ze współczynnikiem uczenia N = 0,01:

Poniżej wynik uczenia perceptronu dla zestawu 1 ze współczynnikiem uczenia N = 0,001:

1. Analiza i dyskusja błędów uczenia i testowania opracowanego perceptronu w zależności od wartości współczynnika uczenia oraz liczby danych uczących.

Jak widać na powyższych przykładach neuron trenowany na kompletnym zestawie danych uzyskuje pełną sprawność bardzo szybko bo ok. 10 epoki, natomiast kiedy pozbawi się go kilku danych uczących (w tym przypadku bardzo istotnej danej zawierającej jedyny dla tej funkcji wynik 1) to może się nigdy nie nauczyć odpowiadać ze 100% skutecznością.

W przypadku eksperymentu ze zmianą współczynnika nauczania widać jak jego wartość wpływa na szybkość uczenia. Neuron o najwyższym współczynniku osiągnął pełną sprawność już w drugiej epoce, podczas gdy ten o 10-krotnie niższym współczynniku został nauczony około 10 epoki. Kolejne 10-krotne zmniejszenie współczynnika powoduje że neuron potrzebowałby ponad 100 epok na wytrenowanie.

1. Wnioski.

Pierwszy podstawowy wniosek z eksperymentu dotyczy ilości danych uczących. Im jest ich więcej tym skuteczność uczenia jest większa. Można tez spekulować czy jego szybkość również będzie większa wraz ze wzrostem ilości danych uczących. Wy,aga to jednak osobnego eksperymentu.

Kolejny wniosek dotyczy związku współczynnika uczenia z szybkością uczenia neuronu. Z powyższych danych można wnioskować iż wyższy współczynnik uczenia przekłada się na szybsze uczenie. Jest to stwierdzenie prawdziwe dla prostych zagadnień. W sieci złożonej z kilku neuronów rozwiązujących bardziej skomplikowane problemy zbyt wysoki współczynnik uczenia może prowadzić do dłuższego uczenia neuronu lub nawet przedwczesnego przetrenowania. Potwierdzenie tej teorii wymaga jednak osobnego eksperymentu.

1. Listing.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Perceptron

{

class Program

{

class Input

{

public double x0;

public double x1;

public double y;

public Input(double x0, double x1, double y)

{

this.x0 = x0;

this.x1 = x1;

this.y = y;

}

}

class Perceptron

{

public List<double> W;

public double Bias;

public List<Input> data;

public double N;

int licznikEpok;

string[] output;

int accuratness;

public void Run()

{

List<Input> inputLearn = new List<Input>(); //Ustawia dane uczące

inputLearn.Add(new Input(0, 0, 0));

inputLearn.Add(new Input(1, 0, 0));

inputLearn.Add(new Input(0, 1, 0));

inputLearn.Add(new Input(1, 1, 1));

List<Input> inputCheck = new List<Input>(); //Ustawia dane walidujące

inputCheck.Add(new Input(0, 0, 0));

inputCheck.Add(new Input(1, 0, 0));

inputCheck.Add(new Input(0, 1, 0));

inputCheck.Add(new Input(1, 1, 1));

for (int i = 0; i < 100; i++) //Każdy obieg pętli to kolejna epoka

{

licznikEpok++;

LoadInput(inputLearn);

Learn();

LoadInput(inputCheck);

Evaluate();

if (licznikEpok % 1000 == 0)

Console.WriteLine(licznikEpok);

}

System.IO.File.WriteAllLines(@"OUTPUT.txt", output);

Console.WriteLine("Waga 0: " + W[0]);

Console.WriteLine("Waga 1: " + W[1]);

}

private int activateFunction(double x)

{

if (x < 1)

return 0;

return 1;

}

public void LoadInput(List<Input> input)

{

data = input;

}

public void Learn()

{

double sum;

foreach (Input element in data)

{

sum = 0;

//sum = element.x0 \* W[0] + element.x1 \* W[1] + Bias;

sum = element.x0 \* W[0] + element.x1 \* W[1];

int result = activateFunction(sum);

if (element.y == result)

{

//neuron działa poprawnie

}

else

{

//trzeba poprawić wagi

W[0] += N \* (element.y - result) \* element.x0;

W[1] += N \* (element.y - result) \* element.x1;

Bias += N \* (1 - result) \* 1;

}

}

}

public void Evaluate()

{

accuratness = 0;

double sum;

foreach (Input element in data)

{

sum = 0;

//sum = element.x0 \* W[0] + element.x1 \* W[1] + Bias;

sum = element.x0 \* W[0] + element.x1 \* W[1] ;

double result = activateFunction(sum);

if (element.y == result)

{

//neuron działa poprawnie

accuratness++;

}

else

{

//błąd

}

}

output[licznikEpok - 1] = accuratness.ToString();

}

public Perceptron()

{

W = new List<double>(); //Lista wag

Random random = new Random();

W.Add(random.NextDouble());

W.Add(random.NextDouble());

Bias = random.NextDouble();

N = 0.01; //Współczynnik uczenia

licznikEpok = 0;

output = new string[100]; //rozmiar tabicy na dane wyjściowe

}

}

static void Main(string[] args)

{

Perceptron p1 = new Perceptron();

p1.Run();

Console.WriteLine("Press any key");

Console.ReadLine();

}

}

}