Michał Ligęza

Kraków, 14.10.2017r.

Podstawy Sztucznej Inteligencji – rok III IS

Sprawozdanie z laboratorium nr 1

„Budowa​ ​i​ ​działanie​ ​perceptronu”

1. **Syntetyczny opis budowy oraz algorytmu uczenia.**Przygotowania sieć neuronowa została zaprojektowana do nauki funkcji logicznych dwóch zmiennych (bramek logicznych). Posiada trzy warstwy:   
   -warstwę wejścia, która składa się z dwóch neuronów przyjmujących dane oraz neuronu ukrytego ze stałą wagą;   
   -warstwę ukrytą składającą się z dwóch neuronów uczących się oraz ukrytego neuronu ze stałą wagą,   
   - warstwę wyjścia z neuronem uczącym się i przekazującym wyniki na wyjściu.   
     
   W neuronach uczących się wykorzystano algorytm propagacji wstecznej, który przy początkowych losowych wagach, liczy swój błąd (różnica między własną odpowiedzią a poprawną), zaczynając od wektorów wyjściowych w stronę wejścia (wstecz).
2. **Uproszczone zestawienie wyników**

**Eta = 0.15, Alfa=0.5**

**NAND**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iteracja** | **X** | **Y** | **NAND(X,Y)** | **Wynik** |
| **1** | **0** | **0** | **1** | **0.886839** |
| **2** | **0** | **1** | **1** | **0.949592** |
| **3** | **1** | **0** | **1** | **0.937603** |
| **4** | **1** | **1** | **0** | **0.958033** |
| **9977** | **0** | **0** | **1** | **0.999149** |
| **9978** | **0** | **1** | **1** | **0.9917** |
| **9979** | **1** | **0** | **1** | **0.991621** |
| **9980** | **1** | **1** | **0** | **-0.0134476** |
| **19997** | **0** | **0** | **1** | **0.999428** |
| **19998** | **0** | **1** | **1** | **0.994474** |
| **19999** | **1** | **0** | **1** | **0.994478** |
| **20000** | **1** | **1** | **0** | **0.00934143** |

**XOR**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iteracja** | **X** | **Y** | **XOR(X,Y)** | **Wynik** |
| **1** | **0** | **0** | **0** | **0.886839** |
| **2** | **0** | **1** | **1** | **0.943095** |
| **3** | **1** | **0** | **1** | **0.925361** |
| **4** | **1** | **1** | **0** | **0.94863** |
| **9921** | **0** | **0** | **0** | **0.000147087** |
| **9922** | **0** | **1** | **1** | **0.990118** |
| **9923** | **1** | **0** | **1** | **0.990048** |
| **9924** | **1** | **1** | **0** | **-0.0128431** |
| **19997** | **0** | **0** | **0** | **4.56122e-05** |
| **19998** | **0** | **1** | **1** | **0.993522** |
| **19999** | **1** | **0** | **1** | **0.993539** |
| **20000** | **1** | **1** | **0** | **0.00911457** |

**Eta = 0.3, Alfa=0.7**

**NAND**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iteracja** | **X** | **Y** | **NAND(X,Y)** | **Wynik** |
| **1** | **0** | **0** | **1** | **0.886839** |
| **2** | **0** | **1** | **1** | **0.95028** |
| **3** | **1** | **0** | **1** | **0.939379** |
| **4** | **1** | **1** | **0** | **0.959913** |
| **9961** | **0** | **0** | **1** | **0.999725** |
| **9962** | **0** | **1** | **1** | **0.995511** |
| **9963** | **1** | **0** | **1** | **0.995576** |
| **9964** | **1** | **1** | **0** | **0.0215948** |
| **19997** | **0** | **0** | **1** | **0.999428** |
| **19998** | **0** | **1** | **1** | **0.994474** |
| **19999** | **1** | **0** | **1** | **0.994478** |
| **20000** | **1** | **1** | **0** | **0.00934143** |

**XOR**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iteracja** | **X** | **Y** | **XOR(X,Y)** | **Wynik** |
| **1** | **0** | **0** | **0** | **0.886839** |
| **2** | **0** | **1** | **1** | **0.936669** |
| **3** | **1** | **0** | **1** | **0.909355** |
| **4** | **1** | **1** | **0** | **0.934255** |
| **9925** | **0** | **0** | **0** | **0.0771261** |
| **9926** | **0** | **1** | **1** | **0.992016** |
| **9927** | **1** | **0** | **1** | **0.992413** |
| **9928** | **1** | **1** | **0** | **0.13974** |
| **19997** | **0** | **0** | **0** | **0.0731456** |
| **19998** | **0** | **1** | **1** | **0.993824** |
| **19999** | **1** | **0** | **1** | **0.994079** |
| **20000** | **1** | **1** | **0** | **0.133895** |

**Eta = 0.05, Alfa=0.3**

**NAND**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iteracja** | **X** | **Y** | **NAND(X,Y)** | **Wynik** |
| **1** | **0** | **0** | **1** | **0.886839** |
| **2** | **0** | **1** | **1** | **0.949128** |
| **3** | **1** | **0** | **1** | **0.93656** |
| **4** | **1** | **1** | **0** | **0.957011** |
| **10017** | **0** | **0** | **1** | **0.997688** |
| **10018** | **0** | **1** | **1** | **0.982674** |
| **10019** | **1** | **0** | **1** | **0.982618** |
| **10020** | **1** | **1** | **0** | **0.000867036** |
| **19997** | **0** | **0** | **1** | **0.998466** |
| **19998** | **0** | **1** | **1** | **0.988454** |
| **19999** | **1** | **0** | **1** | **0.988415** |
| **20000** | **1** | **1** | **0** | **0.000374419** |

**XOR**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iteracja** | **X** | **Y** | **XOR(X,Y)** | **Wynik** |
| **1** | **0** | **0** | **0** | **0.886839** |
| **2** | **0** | **1** | **1** | **0.947027** |
| **3** | **1** | **0** | **1** | **0.933171** |
| **4** | **1** | **1** | **0** | **0.954615** |
| **10121** | **0** | **0** | **0** | **0.00090175** |
| **10122** | **0** | **1** | **1** | **0.976766** |
| **10123** | **1** | **0** | **1** | **0.976702** |
| **10124** | **1** | **1** | **0** | **0.00154414** |
| **19997** | **0** | **0** | **0** | **0.000363142** |
| **19998** | **0** | **1** | **1** | **0.985307** |
| **19999** | **1** | **0** | **1** | **0.985268** |
| **20000** | **1** | **1** | **0** | **0.000601314** |

1. **Analiza i dyskusja błędów uczenia ​w zależności​ ​od​ ​wartości​ ​współczynnika​ ​uczenia​ ​oraz​ ​liczby​ ​danych​ ​uczących  
     
   NAND**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Eta = 0.15, Alfa=0.5 | Eta = 0.3,  Alfa=0.7 | Eta = 0.05,  Alfa=0.3 |
| 1 | 0.886839 | 0.886839 | 0.886839 |
| 2 | 0.949592 | 0.95028 | 0.949128 |
| 3 | 0.937603 | 0.939379 | 0.93656 |
| 4 | 0.958033 | 0.959913 | 0.957011 |
| N | 0.999149 | 0.999725 | 0.997688 |
| N+1 | 0.9917 | 0.995511 | 0.982674 |
| N+2 | 0.991621 | 0.995576 | 0.982618 |
| N+3 | -0.0134476 | 0.0215948 | 0.000867036 |
| 19997 | 0.999428 | 0.999428 | 0.998466 |
| 19998 | 0.994474 | 0.994474 | 0.988454 |
| 19999 | 0.994478 | 0.994478 | 0.988415 |
| 20000 | 0.00934143 | 0.00934143 | 0.000374419 |

**N~10 000**

**Dla funkcji NAND, najdokładniejsze górne wyniki (cel=1) uzyskano równolegle dla eta=0.15,alfa=0,5 oraz eta=0.3, alfa=0.7, z zaznaczeniem, iż szybciej do pożądanego wyniku przybliżyło się drugie z rozwiązań. Natomiast najdokładniejsze wyniki dolne (cel=0) uzyskano przy eta=0.05, alfa=0.3**

**Niezależnie od ustawienia, po około 10 tysiącach iteracji, sieć osiągała błąd nie przekraczający 0.025, natomiast po 20 tysiącach iteracji, błąd nie przekroczył 0.015**

**XOR**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Eta = 0.15, Alfa=0.5 | Eta = 0.3,  Alfa=0.7 | Eta = 0.05,  Alfa=0.3 |
| 1 | 0.886839 | 0.886839 | 0.886839 |
| 2 | 0.943095 | 0.936669 | 0.947027 |
| 3 | 0.925361 | 0.909355 | 0.933171 |
| 4 | 0.94863 | 0.934255 | 0.954615 |
| N | 0.000147087 | 0.0771261 | 0.00090175 |
| N+1 | 0.990118 | 0.992016 | 0.976766 |
| N+2 | 0.990048 | 0.992413 | 0.976702 |
| N+3 | -0.0128431 | 0.13974 | 0.00154414 |
| 19997 | 4.56122e-05 | 0.0731456 | 0.000363142 |
| 19998 | 0.993522 | 0.993824 | 0.985307 |
| 19999 | 0.993539 | 0.994079 | 0.985268 |
| 20000 | 0.00911457 | 0.133895 | 0.000601314 |

**N~10 000**

**Dla funkcji XOR, najdokładniejsze górne wyniki (cel=1) uzyskano dla eta= 0.3, alfa=0.7, gdzie odchylenie nie przekroczyło 0.01, natomiast dolne wyniki (cel=0) dla eta=0.05, alfa=0.3, dla których błąd wynosił poniżej 0.001**

1. **Wnioski  
     
   Dla większości ustawień, 10 tysięcy pojedynczych procesów** nauczania wystarczyło, by wyniki otrzymane przez sieć nie przekraczały 3\*10-2. Podwojenie ilości nauczania powodowało, że dokładność zwiększała się o wartości rzędu 10-3. Przyjmując że wartość jest dokładna, jeżeli maksymalna różnica wyniku dokładnego od otrzymanego jest równa 0.1, całość rozwiązań otrzymanych dla funkcji NAND jest wystarczająco dokładna, natomiast dla funkcji XOR wszystkie ustawienia poza Eta = 0.3, Alfa=0.7 dają wyniki precyzyjne.
2. **Listing kodu programu**  
   struktura **Connection** – przechowuje dane na temat połączenia

{

**weight** – waga połączenia

**deltaWeight** – ostatnia zmiana wagi połączenia

}

typedef **Layer** – wektor Neuronów, typ opisujący warstwę sieci

klasa **Neuron** – implementacja neuronu sztucznej sieci

{

**public**:

**Neuron**() – konstruktor, przyjmuje ilość wyjść oraz swój indeks

**feedForward**() – sumuje iloczyny danych na wejściach z ich wagami

**setOutputVal**() – ustawia wartość wyjścia

**getOutputVal**() – zwraca wartość wyjścia

**calcOutputGradients**() – liczy gradienty neuronów wyjścia

**calcHiddenGradients**() – liczy gradienty neuronów ukrytych warstw

**updateInputWeights**() – poprawia wagi połączeń wejścia

**private**:

**static** **eta** – wartość eta w funkcji zmiany wag

**static** **alpha** – wartość alfa w funkcji zmiany wag

**static** **randomWeight**() – ustawia generowaną wage początkową

**n\_outputVal** – wartość na wyjściu

**n\_gradient** – wartość gradientu

**n\_outputWeights** – wektor połączeń, posiadający wagi połączeń

**n\_myIndex** – wartość indeksu w warstwie

**sumDow**() – sumuje wagowo gradienty w następnej warstwie

**transferFunction**() – liczy funkcję transferu (w tym przypadku tanh(x))

**transferFunctionDeriverative**() – liczy pochodną funkcji transferu

}

Klasa **Net** – implementacja sieci neuronowej

{

**public**:

**Net** – konstruktor, przyjmujący wektor liczb naturalnych opisujących ilość neuronów w sieci, przy czym długość wektora to ilość warstw

**feedForward**() – pobiera dane wejściowe, oblicza przewidywany wynik

**backProp**()- propagacja wsteczna

**getResults**() – do przekazanego wektora zapisuje wynik

**private**:

**n\_layers** – wektor warstw

**n\_error** – zawiera błąd (wynik oczekiwany minus otrzymany)

**recentAverageError** – poprzedni błąd

**recentAverageSmoothingFactor** – poprzedni współczynnik błędu

}

WYKONANIE:

W pętli (20 tys powtórzeń) {

myNet.feedForward(inputVals);

myNet.backProp(targetVals);

myNet.getResults(resultVals);

}