Informatyka, studia niestacjonarne, mgr II st. semestr I

Obliczenia inteligentne **2015/2016**

Prowadzący: dr inż. Kamil Stokfiszewski Niedziela, 13.45

Data oddania: _____ Ocena: _____

Szymon Łyszkowski 206809
Piotr Kluch 206799

Zadnie nr 1. Sieć MADALINE do rozpoznawania liter

1. Cel

Głównym celem zadania było zapoznanie z zasadą działania sieci neuronowej MADALINE (Many ADAptive LINear NEtwork) oraz jej implementacja. Od sieci oczekuje się, że będzie w stanie rozpoznać literę poprzez podanie jej jako wektor zer oraz jedynek.

2. Wprowadzenie

Sieć MADALINE jest de facto połączeniem wielu sieci ADALINE (ADAptive LINear NEtwork). ADALINE jest prostą sztuczną siecią neuronową, która składa się z jednego neuronu. ADALINE posiada warstwę wejsciową, ukrytą oraz wyjściową. Warstwa wejsciowa to zbiór danych (wektor), które mają być rozpoznane (zklasyfikowane) przez sieć. Warstwa ukryta to zbiór wag, który jest mnożony ze zbiorem danych wejściowych tak aby otrzymać wartość na wyjściu sieci. MADALINE korzysta z konceptu ADALINE poprzez zwielokrotnienie warstwy ukrytej i produkcji wielu wyjść dla tego samego zbioru wejściowego. W przypdaku sztucznej sieci MADALINE pominięto proces uczenia z racji na znane wartości wag, które umożliwią rozpoznanie danej litery. Wagi te są odzwierciedleniem macierzy binarnej o wymiarach (4x4) dla danej litery, która ma być rozpoznana. Aby porównać wyjście neuronu, wektor wejściowy jak również wektory wag jest normalizowany aby móc prawidłowo porównać wartość na wyjściu, która jest w przedziale $< 0, 1 >$.

Tym sposobem wyjście neuronu, które wykazało największą wartość (podobieństwo do wektora wejściowego) klasyfikuje literę jako rozpoznaną.

3. Opis implementacji

Implementacja została przygotowana w języku Python. Cała sieć jest realizowana przez klasę MadalineNetwork (networks/madeline_network.py), która wykorzystując zasadę kompozycji używa klasy Neuron (neurons/neuron.py). W konstruktorze klasy MadalineNetwork podajemy następujące parametry: dane wejściowe, liczbę neuronów w sieci oraz dwuwymiarowy wektor wag dla poszczególnych neuronów. Rozpoznawanie liter jest realizowane przez metodę **run_madaline()**.

```
def run_madaline(self):
        self.__normalize_network_vectors()
        output_results = self.__compute_network_outputs()
        return self.__classify_output(output_results)
```

Metoda ta normalizuje wektory, przekazuje dane wejściowe do wszystkich neuronów sieci, oblicza wartości wyjść neuronów oraz zwraca wartość wyjścia o największej wartości. Najwyższa wartość oznacza największe podobieństwo danych wejściowych sieci do wartości wzorcowej. Normalizacja i mapowanie wektora:

```
def __return_normalized_vector(self, input_vector):
        flatten_vector = input_vector.flatten()
        vector_length = numpy.linalg.norm(flatten_vector)
        normalized_vector = []
        for vector_element in flatten_vector:
        normalized_vector.append(vector_element / vector_length)
        return normalized_vector
```

4. Materiały i metody

Litery, które mają być rozpoznawane przez sieć to binarne dwuwymiarowe tablice 4x4 wprowadzone do neuronów sieci. Przykładowa tablica wzorcowa dla litery "Y":

```
def get_Y_four_by_four(self):
        y_sample_array = numpy.array([[1, 0, 0, 1],
                                      [0, 1, 1, 0],
                                      [0, 1, 0, 0],
                                      [1, 0, 0, 0]])
        return y_sample_array, 'Letter Y'
```

Implementacja programu obowiązuje rozpoznawanie liter: **T,V,W,X,Y,Z**. Wzorce tego typu zostały użyte do rozpoznawania liter wejściowych. Każdy wzorzec jest normalizowany i mapowany na wektor jednowymiarowy tak aby można było łatwo obliczyć wartość wyjścia wektora zgodnie ze wzorem:

$$\sum_{i=1}^{n} x_i w_i$$

Każda próbka wejściowa, jest przetwarzana w ten sam sposób co wagi neuronu, tak aby na wyjściu wektora otrzymany skalar był z przedziału $< 0, 1 >$.

5. Wyniki

Poszczególne neurony sieci rozpoznają następujące litery:
Neuron of index 0 recognizes Letter T
Neuron of index 1 recognizes Letter V
Neuron of index 2 recognizes Letter W
Neuron of index 3 recognizes Letter X
Neuron of index 4 recognizes Letter Y
Neuron of index 5 recognizes Letter Z
Dla sieci przeprowadzone zostały testy dla następujących danych wejściowych:
1)Podana próbka jest w pełni zgodna ze wzorcem litery "Z".

```
[1, 1, 1, 1]
[0, 0, 1, 0]
[0, 1, 0, 0]
[1, 1, 1, 1]

Neuron outputs: [0.79999999999999993, 0.55901699437494745,
0.5, 0.67082039324993692,
0.64549722436790291,
0.99999999999999989]
Neuron with highest output was: 1.0 and its index is 5
```

Litera jest rozpoznawana poprawnie.
2)Litera "Z", która posiada trzy "zaciemnione" - brakujące elementy wzorca.

```
[1, 0, 0, 1]
[0, 0, 1, 0]
[0, 1, 0, 0]
[1, 1, 0, 1]

Neuron outputs: [0.59761430466719678, 0.53452248382484868,
0.59761430466719678, 0.80178372573727297,
0.77151674981045959, 0.83666002653407545]
Neuron with highest output was: 0.836660026534 and its index is 5
```

Przypadek ten jest także rozpoznawany poprawnie.
3)Litera "Y", która posiada jeden "zaciemniony" element oraz jeden dodatkowy (w tle):

```
[1, 0, 0, 1],
[0, 0, 1, 0],
```

```
[0, 1, 0, 0],
[1, 0, 0, 1]

Neuron outputs: [0.51639777949432231, 0.43301270189221941,
0.64549722436790291, 0.86602540378443871,
0.83333333333333359, 0.77459666924148352]
Neuron with highest output was: 0.866025403784 and its index is 3
```

Jest rozpoznawana **niepoprawnie**, gdyż klasyfikowana jest jako "X". Jednakże dla przypadku bez dodatkowego elementu w tle:

```
[1, 0, 0, 1]
[0, 0, 1, 0]
[0, 1, 0, 0]
[1, 0, 0, 0]

Neuron outputs: [0.56568542494923801, 0.47434164902525683,
0.56568542494923801, 0.79056941504209477,
0.9128709291752769, 0.70710678118654746]
Neuron with highest output was: 0.912870929175 and its index is 4
```

Jest już klasyfikowana poprawnie.

6. Dyskusja

Wyniki rozpoznawania przez sieć mogą być różne w zależności od zaciemnienia oraz podobieństwa do danych wzorcowych. Im większa rozdzielczość matrycy wzorcowej oraz danych wejściowych tym wynik może być dokładniejszy. Dla matrycy 4x4 pewne litery nie mogą być dobrze odwzorowane np. "W". Jednakże analiza większych matryc może być trudniejsza z racji na możliwe przesunięcie litery w dowolnym kierunku. Implementacja sieci nie bierze obsługuje takiego przypadku, który nie byłby zaklasyfikowany z uwagi na brak danych (sieć porównywałaby dane wejściowe z pustym obszarem we wzorcu). W takim przypadku należałoby zaimplementować centrowanie litery wejściowej lub porównywanie części matrycy wejściowej ze wzorcem co zdecydowanie niesie za sobą większy nakład obliczeniowy.

7. Wnioski

Rozdzielczość matrycy jest kluczowa dla rozpoznawania przypadków brzegowych (podobieństwo do dwóch lub więcej wzorców).