|  |  |
| --- | --- |
| **Scenariusz 5** | |
| **Temat ćwiczenia:** | Budowa i działanie sieci Kohonena dla WTA |
| **Wykonał:** | Mateusz Mazur, gr 2 |

1. **Cel ćwiczenia**

Celem ćwiczenia jest poznanie budowy i działanie sieci Kohonena przy wykorzystaniu reguły WTA do odwzorowania istotnych cech kwiatków.

1. **Budowa sieci i wykorzystany algorytm uczenia**

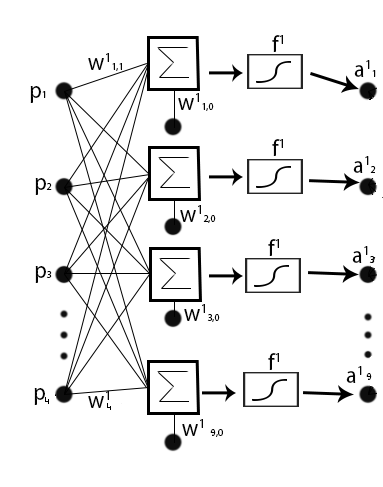
Celem budowanej sieci jest podział wzorców uczących na klasy obrazów zbliżonych do siebie i przyporządkowanie każdej klasie osobnego elementu wyjściowego. Podział na grupy odbywa się w takich sposób, by elementy w tej samej grupie były do siebie podobne a jednocześnie jak najbardziej odmienne od elementów z pozostałych grup. Poniżej przedstawiono zestaw danych służący do testowania oraz uczenia sieci neuronowej. Dane te podzielono pomiędzy dane uczące i testujące w stosunku 70% do 30 %.

Tab. 1 Zestaw danych

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Długość działki kielicha | Szerokość działki kielicha | Długość płatka | Szerokość płatka | *Rodzaj kwiatu* |
| 5.8 | 4.0 | 1.2 | 0.2 | *I. setosa* |
| 5.7 | 4.4 | 1.5 | 0.4 | *I. setosa* |
| 5.7 | 3.8 | 1.7 | 0.3 | *I. setosa* |
| 5.5 | 4.2 | 1.4 | 0.2 | *I. setosa* |
| 5.5 | 3.5 | 1.3 | 0.2 | *I. setosa* |
| 5.4 | 3.9 | 1.7 | 0.4 | *I. setosa* |
| 5.4 | 3.7 | 1.5 | 0.2 | *I. setosa* |
| 5.4 | 3.9 | 1.3 | 0.4 | *I. setosa* |
| 5.4 | 3.4 | 1.7 | 0.2 | *I. setosa* |
| 5.4 | 3.4 | 1.5 | 0.4 | *I. setosa* |
| 5.3 | 3.7 | 1.5 | 0.2 | *I. setosa* |
| 5.2 | 3.5 | 1.4 | 0.2 | *I. setosa* |
| 5.2 | 3.5 | 1.5 | 0.2 | *I. setosa* |
| 5.2 | 3.4 | 1.4 | 0.2 | *I. setosa* |
| 5.2 | 4.1 | 1.5 | 0.1 | *I. setosa* |
| 5.1 | 3.5 | 1.4 | 0.3 | *I. setosa* |
| 5.1 | 3.8 | 1.5 | 0.3 | *I. setosa* |
| 5.1 | 3.7 | 1.5 | 0.4 | *I. setosa* |
| 5.1 | 3.3 | 1.7 | 0.5 | *I. setosa* |
| 5.1 | 3.4 | 1.5 | 0.2 | *I. setosa* |
| 5.1 | 3.8 | 1.9 | 0.4 | *I. setosa* |
| 5.1 | 3.8 | 1.6 | 0.2 | *I. setosa* |
| 5.0 | 3.6 | 1.4 | 0.3 | *I. setosa* |
| 5.0 | 3.4 | 1.5 | 0.2 | *I. setosa* |
| 5.0 | 3.0 | 1.6 | 0.2 | *I. setosa* |
| 5.0 | 3.4 | 1.6 | 0.4 | *I. setosa* |
| 5.0 | 3.2 | 1.2 | 0.2 | *I. setosa* |
| 5.0 | 3.5 | 1.3 | 0.3 | *I. setosa* |
| 5.0 | 3.5 | 1.6 | 0.6 | *I. setosa* |
| 5.0 | 3.3 | 1.4 | 0.2 | *I. setosa* |
| 4.9 | 3.0 | 1.4 | 0.2 | *I. setosa* |
| 4.9 | 3.1 | 1.5 | 0.1 | *I. setosa* |
| 4.9 | 3.1 | 1.5 | 0.2 | *I. setosa* |
| 4.9 | 3.6 | 1.4 | 0.1 | *I. setosa* |
| 4.8 | 3.4 | 1.6 | 0.2 | *I. setosa* |
| 4.8 | 3.0 | 1.4 | 0.1 | *I. setosa* |
| 4.8 | 3.4 | 1.9 | 0.2 | *I. setosa* |
| 4.8 | 3.1 | 1.6 | 0.2 | *I. setosa* |
| 4.8 | 3.0 | 1.4 | 0.3 | *I. setosa* |
| 4.7 | 3.2 | 1.3 | 0.2 | *I. setosa* |
| 4.7 | 3.2 | 1.6 | 0.2 | *I. setosa* |
| 4.6 | 3.1 | 1.5 | 0.2 | *I. setosa* |
| 4.6 | 3.4 | 1.4 | 0.3 | *I. setosa* |
| 4.6 | 3.6 | 1.0 | 0.2 | *I. setosa* |
| 4.6 | 3.2 | 1.4 | 0.2 | *I. setosa* |
| 4.5 | 2.3 | 1.3 | 0.3 | *I. setosa* |
| 4.4 | 2.9 | 1.4 | 0.2 | *I. setosa* |
| 4.4 | 3.0 | 1.3 | 0.2 | *I. setosa* |
| 4.4 | 3.2 | 1.3 | 0.2 | *I. setosa* |
| 4.3 | 3.0 | 1.1 | 0.1 | *I. setosa* |
| 7.0 | 3.2 | 4.7 | 1.4 | *I. versicolor* |
| 6.9 | 3.1 | 4.9 | 1.5 | *I. versicolor* |
| 6.8 | 2.8 | 4.8 | 1.4 | *I. versicolor* |
| 6.7 | 3.1 | 4.4 | 1.4 | *I. versicolor* |
| 6.7 | 3.0 | 5.0 | 1.7 | *I. versicolor* |
| 6.7 | 3.1 | 4.7 | 1.5 | *I. versicolor* |
| 6.6 | 2.9 | 4.6 | 1.3 | *I. versicolor* |
| 6.6 | 3.0 | 4.4 | 1.4 | *I. versicolor* |
| 6.5 | 2.8 | 4.6 | 1.5 | *I. versicolor* |
| 6.4 | 3.2 | 4.5 | 1.5 | *I. versicolor* |
| 6.4 | 2.9 | 4.3 | 1.3 | *I. versicolor* |
| 6.3 | 3.3 | 4.7 | 1.6 | *I. versicolor* |
| 6.3 | 2.5 | 4.9 | 1.5 | *I. versicolor* |
| 6.3 | 2.3 | 4.4 | 1.3 | *I. versicolor* |
| 6.2 | 2.2 | 4.5 | 1.5 | *I. versicolor* |
| 6.2 | 2.9 | 4.3 | 1.3 | *I. versicolor* |
| 6.1 | 2.9 | 4.7 | 1.4 | *I. versicolor* |
| 6.1 | 2.8 | 4.0 | 1.3 | *I. versicolor* |
| 6.1 | 2.8 | 4.7 | 1.2 | *I. versicolor* |
| 6.1 | 3.0 | 4.6 | 1.4 | *I. versicolor* |
| 6.0 | 2.2 | 4.0 | 1.0 | *I. versicolor* |
| 6.0 | 2.9 | 4.5 | 1.5 | *I. versicolor* |
| 6.0 | 2.7 | 5.1 | 1.6 | *I. versicolor* |
| 6.0 | 3.4 | 4.5 | 1.6 | *I. versicolor* |
| 5.9 | 3.0 | 4.2 | 1.5 | *I. versicolor* |
| 5.9 | 3.2 | 4.8 | 1.8 | *I. versicolor* |
| 5.8 | 2.7 | 4.1 | 1.0 | *I. versicolor* |
| 5.8 | 2.7 | 3.9 | 1.2 | *I. versicolor* |
| 5.8 | 2.6 | 4.0 | 1.2 | *I. versicolor* |
| 5.7 | 2.8 | 4.5 | 1.3 | *I. versicolor* |
| 5.7 | 2.6 | 3.5 | 1.0 | *I. versicolor* |
| 5.7 | 3.0 | 4.2 | 1.2 | *I. versicolor* |
| 5.7 | 2.9 | 4.2 | 1.3 | *I. versicolor* |
| 5.7 | 2.8 | 4.1 | 1.3 | *I. versicolor* |
| 5.6 | 2.9 | 3.6 | 1.3 | *I. versicolor* |
| 5.6 | 3.0 | 4.5 | 1.5 | *I. versicolor* |
| 5.6 | 2.5 | 3.9 | 1.1 | *I. versicolor* |
| 5.6 | 3.0 | 4.1 | 1.3 | *I. versicolor* |
| 5.6 | 2.7 | 4.2 | 1.3 | *I. versicolor* |
| 5.5 | 2.3 | 4.0 | 1.3 | *I. versicolor* |
| 5.5 | 2.4 | 3.8 | 1.1 | *I. versicolor* |
| 5.5 | 2.4 | 3.7 | 1.0 | *I. versicolor* |
| 5.5 | 2.5 | 4.0 | 1.3 | *I. versicolor* |
| 5.5 | 2.6 | 4.4 | 1.2 | *I. versicolor* |
| 5.4 | 3.0 | 4.5 | 1.5 | *I. versicolor* |
| 5.2 | 2.7 | 3.9 | 1.4 | *I. versicolor* |
| 5.1 | 2.5 | 3.0 | 1.1 | *I. versicolor* |
| 5.0 | 2.0 | 3.5 | 1.0 | *I. versicolor* |
| 5.0 | 2.3 | 3.3 | 1.0 | *I. versicolor* |
| 4.9 | 2.4 | 3.3 | 1.0 | *I. versicolor* |
| 7.9 | 3.8 | 6.4 | 2.0 | *I. virginica* |
| 7.7 | 3.8 | 6.7 | 2.2 | *I. virginica* |
| 7.7 | 2.6 | 6.9 | 2.3 | *I. virginica* |
| 7.7 | 2.8 | 6.7 | 2.0 | *I. virginica* |
| 7.6 | 3.0 | 6.6 | 2.1 | *I. virginica* |
| 7.4 | 2.8 | 6.1 | 1.9 | *I. virginica* |
| 7.3 | 2.9 | 6.3 | 1.8 | *I. virginica* |
| 7.2 | 3.6 | 6.1 | 2.5 | *I. virginica* |
| 7.2 | 3.2 | 6.0 | 1.8 | *I. virginica* |
| 7.2 | 3.0 | 5.8 | 1.6 | *I. virginica* |
| 7.1 | 3.0 | 5.9 | 2.1 | *I. virginica* |
| 6.9 | 3.2 | 5.7 | 2.3 | *I. virginica* |
| 6.9 | 3.1 | 5.4 | 2.1 | *I. virginica* |
| 6.9 | 3.1 | 5.1 | 2.3 | *I. virginica* |
| 6.8 | 3.0 | 5.5 | 2.1 | *I. virginica* |
| 6.8 | 3.2 | 5.9 | 2.3 | *I. virginica* |
| 6.7 | 2.5 | 5.8 | 1.8 | *I. virginica* |
| 6.7 | 3.3 | 5.7 | 2.1 | *I. virginica* |
| 6.7 | 3.1 | 5.6 | 2.4 | *I. virginica* |
| 6.7 | 3.3 | 5.7 | 2.5 | *I. virginica* |
| 6.7 | 3.0 | 5.2 | 2.3 | *I. virginica* |
| 6.5 | 3.0 | 5.8 | 2.2 | *I. virginica* |
| 6.5 | 3.2 | 5.1 | 2.0 | *I. virginica* |
| 6.5 | 3.0 | 5.5 | 1.8 | *I. virginica* |
| 6.5 | 3.0 | 5.2 | 2.0 | *I. virginica* |
| 6.4 | 2.7 | 5.3 | 1.9 | *I. virginica* |
| 6.4 | 3.2 | 5.3 | 2.3 | *I. virginica* |
| 6.4 | 2.8 | 5.6 | 2.1 | *I. virginica* |
| 6.4 | 2.8 | 5.6 | 2.2 | *I. virginica* |
| 6.4 | 3.1 | 5.5 | 1.8 | *I. virginica* |
| 6.3 | 3.3 | 6.0 | 2.5 | *I. virginica* |
| 6.3 | 2.9 | 5.6 | 1.8 | *I. virginica* |
| 6.3 | 2.7 | 4.9 | 1.8 | *I. virginica* |
| 6.3 | 2.8 | 5.1 | 1.5 | *I. virginica* |
| 6.3 | 3.4 | 5.6 | 2.4 | *I. virginica* |
| 6.3 | 2.5 | 5.0 | 1.9 | *I. virginica* |
| 6.2 | 2.8 | 4.8 | 1.8 | *I. virginica* |
| 6.2 | 3.4 | 5.4 | 2.3 | *I. virginica* |
| 6.1 | 3.0 | 4.9 | 1.8 | *I. virginica* |
| 6.1 | 2.6 | 5.6 | 1.4 | *I. virginica* |
| 6.0 | 2.2 | 5.0 | 1.5 | *I. virginica* |
| 6.0 | 3.0 | 4.8 | 1.8 | *I. virginica* |
| 5.9 | 3.0 | 5.1 | 1.8 | *I. virginica* |
| 5.8 | 2.7 | 5.1 | 1.9 | *I. virginica* |
| 5.8 | 2.8 | 5.1 | 2.4 | *I. virginica* |
| 5.8 | 2.7 | 5.1 | 1.9 | *I. virginica* |
| 5.7 | 2.5 | 5.0 | 2.0 | *I. virginica* |
| 5.6 | 2.8 | 4.9 | 2.0 | *I. virginica* |
| 4.9 | 2.5 | 4.5 | 1.7 | *I. virginica* |

Proces uczenia odbywa się przy pomocy uczenia rywalizującego, które jest metodą uczenia sieci samoorganizujących. Podczas uczenia neurony uczą się rozpoznawać dane i zbliżają się odpowiednio do obszarów zajmowanych przez te dane. Po wejściu każdego wektora uczącego wybierany jest tylko jeden neuron, najbliższy prezentowanemu wzorcowi. Neurony rywalizują między sobą i zwycięża ten neuron, którego wartość jest największa. Zwycięski neuron przyjmuje na wyjściu wartość 1, a pozostałe neurony 0. Jest to uczenie bez nauczyciela.

Neuron zwycięski ma prawo uaktualnić swoje wagi wg jednej z dwóch zasad: WTA(Winner Takes All), WTM(Winner Takes Most). W tym projekcie została wykorzystana pierwsza zasada.



Rys 1. Schemat sieci neuronowej

Wielokrotne testy różnych konfiguracji wykazały, że dla danego problemu najlepszymi wynikami cechuje się sieć zbudowana z 9 neuronów, która w dodatku biorąc pod uwagę ilość neuronów charakteryzuje się małym skomplikowaniem.

1. Wybór za zakresu 0 do 1  
2. Normalizacja wektorów danych wejściowych  
3. Wybór początkowych wartości wag, jako niewielkich liczb losowych z zakresu -1 do 1  
4. Dla danego wektora uczącego obliczmy odpowiedź sieci. A więc dla pojedynczego neuronu obliczana jest suma ilorazów sygnałów wejściowych i wag - uki.   
5. Wybierany jest neuron, dla którego obliczona suma jest największa i zaktualizowanie jego wag według wzoru:

gdzie:

η to współczynnik uczenia wybierany z zakresu od 0 do 1

6. Znormalizowanie wartości nowego wektora wag  
7. Zwycięski neuron daje odpowiedź na swoim wyjściu 1, a pozostałe 0  
8. Wczytanie kolejnego wektora uczącego

1. **Otrzymane wyniki**

Proces badania kwiatów został podzielony na trzy etapy: badanie rozpoznawania kwiatu ze względu na wielkość kielicha, badanie rozpoznawania kwiatu ze względu na wielkość płatków oraz badanie rozpoznawania kwiatu ze względu na wszystkie kryteria. Na podstawie danych uczących utworzono poniższe wykresy stanowiące odniesienie, na podstawie którego weryfikowano wyniki uzyskane przez sieć neuronową.

Wykres 1. Podział na grupy ze względu na wielkość działki kielicha

Wykres 2. Podział na grupy ze względu na wielkość płatka

Na podstawie wielu testów sieci neuronowej wyłoniono liczbę grup, na które jest w stanie sieć podzielić kwiatki biorąc pod uwagę kryterium wielkości działki kielicha, wielkości płatka oraz wszystkie kryteria razem. Możemy zauważyć, iż pokrywają się one z powyższymi wykresami, co pozwala stwierdzić, iż przygotowana przeze mnie sieć jest w stanie poradzić sobie z zadaniem grupowania kwiatów według różnych parametrów. W poniższej tabeli ukazano liczbę grup wskazaną przez moją sieć neuronową dla każdej z rozważanych przeze mnie cech:

Tab 2. Liczba aktywnych neuronów podczas badania różnych kryterium podziału kwiatów

|  |  |
| --- | --- |
| Ilość grup na które sieć neuronowa podzieliła kwiaty w zależności od badanych cech (ilość aktywnych neuronów przy badaniu każdego z kryterium) | |
|
| Kryterium wielkości działki kielicha | 2 |
| Kryterium wielkości płatka | 3 |
| Wszystkie kryteria razem | 2 |

W poniższej tabeli przedstawiono procent poprawnych odpowiedzi sieci w procesie uczenia podczas 50 epoki. Jak widać największe zróżnicowanie wyników mamy w przypadku kryterium wielkości kielicha oraz kryterium wielkości płatka. W przypadku, gdy bierzemy pod uwagę wszystkie kryteria moja sieć nie ma większych problemów z pogrupowaniem kwiatów, zawsze wskazując poprawną odpowiedź. W przypadku kryterium wielkości działki kielicha, gdzie to rozróżniamy podział na dwie grupy widzimy, iż wraz ze wzrostem wartości współczynnika uczenia procent poprawnego pogrupowania kwiatów wzrasta. Zaobserwowano tutaj małe spadki poprawności jak w przypadku, kiedy to współczynnik uczenia ma wartość 0,0001 i 0,001 oraz w przypadku współczynnika równego 0,01 gdzie zachodzi gwałtowny spadek poprawności odpowiedzi sieci. W przypadku grupowania kwiatów pod względem wielkości płatka, kiedy to sieć musi rozróżnić 3 różne grupy widzimy, iż procent poprawnych odpowiedzi jest znacznie niż w dwóch pozostałych przypadkach. W tym kryterium poziom poprawnych odpowiedzi wzrasta do pewnego pułapu wraz ze wzrostem wartości współczynnika uczenia, aż do momentu, w którym się stabilizuje i nie zmienia już swojej wartości.

Tab 3. Procent poprawnych odpowiedzi sieci podczas procesu uczenia przy liczbie epok równej 50

|  |  |  |  |
| --- | --- | --- | --- |
| Wartość współczynnika uczenia | Procent poprawnych odpowiedzi podczas procesu uczenia przy liczbie epok równej 50 | | |
|
| Kryterium wielkości działki kielicha | Kryterium wielkości płatka | Wszystkie kryteria razem |
|
| 0,0001 | 73,33% | 33,33% | 100,00% |
| 0,001 | 69,52% | 73,33% | 100,00% |
| 0,005 | 100,00% | 66,66% | 100,00% |
| 0,01 | 68,88% | 67,61% | 100,00% |
| 0,09 | 100,00% | 66,66% | 100,00% |
| 0,1 | 100,00% | 66,66% | 100,00% |

W poniższej tabeli przedstawiono procent poprawnych odpowiedzi sieci w procesie testowania. Dane testujące nie zawierają się w danych uczących, dlatego też otrzymane wyniki pokazują, czy sieć jest gotowa na rozróżnianie i grupowanie kwiatów z którymi wcześniej się nie zapoznała. Jak możemy zauważyć ponownie największe zróżnicowanie wyników mamy w przypadku kryterium wielkości kielicha oraz kryterium wielkości płatka. W przypadku, gdy bierzemy pod uwagę wszystkie kryteria sieć nie ma większych problemów z pogrupowaniem kwiatów z którymi to wymiarami nigdy wcześniej nie mogła się zapoznać, zawsze wskazując poprawną odpowiedź. W przypadku kryterium wielkości działki kielicha, gdzie to rozróżniamy podział na dwie grupy widzimy, iż wraz ze wzrostem wartości współczynnika uczenia procent poprawnego pogrupowania kwiatów wzrasta. Zaobserwowano tutaj małe spadki poprawności jak w przypadku, kiedy to współczynnik uczenia ma wartość 0,0001 i 0,001 oraz w przypadku współczynnika równego 0,01 gdzie zachodzi gwałtowny spadek poprawności odpowiedzi sieci. Wyniki te znajdują swoje odzwierciedlenie w odpowiedziach podczas procesu uczenia. W przypadku grupowania kwiatów pod względem wielkości płatka, kiedy to sieć musi rozróżnić 3 różne grupy widzimy, iż procent poprawnych odpowiedzi jest znacznie niż w dwóch pozostałych przypadkach. Widzimy tutaj jednak że pomimo słabszych odpowiedzi sieci przy współczynniku uczenia 0,01 podczas procesu uczenia, w momencie testowania sieć ta bardzo dobrze radzi sobie z grupowaniem kwiatów, z którymi nie zapoznała, co pozwala stwierdzić, iż wyższa wartość współczynnika uczenia może w sposób wymierny przełożyć się na efektywność naszej sieci przy identyfikacji kwiatów z poza danych uczących. W tym kryterium poziom poprawnych odpowiedzi wzrasta do pewnego pułapu wraz ze wzrostem wartości współczynnika uczenia, aż do momentu, w którym się stabilizuje i nie zmienia już swojej wartości.

Tab 4. Procent poprawnych odpowiedzi sieci podczas procesu testowania

|  |  |  |  |
| --- | --- | --- | --- |
| Wartość współczynnika uczenia | Procent poprawnych odpowiedzi podczas procesu testowania | | |
|
| Kryterium wielkości działki kielicha | Kryterium wielkości płatka | Wszystkie kryteria razem |
|
| 0,0001 | 93,33% | 33,33% | 100,00% |
| 0,001 | 84,44% | 73,33% | 100,00% |
| 0,005 | 95,55% | 71,11% | 100,00% |
| 0,01 | 80,00% | 84,44% | 100,00% |
| 0,09 | 95,55% | 80,00% | 100,00% |
| 0,1 | 95,55% | 84,44% | 100,00% |

1. **Analiza wyników**

Analizę wyników przeprowadzono osobno dla każdego z badanych kryterium. Każde z nich oparto o wartości iloczynu skalarnego uzyskiwanego przez zwycięskie neurony podczas procesu identyfikacji i przydziału do grupy każdego z kwiatów podczas procesu testowania. W powyższych tabelach można było zobaczyć procentowe zestawienie poprawności odpowiedzi mojej sieci podczas grupowania kwiatów na podstawie różnych kryteriów, a także w różnych procesach: uczenia jak i testowania. Wyniki zamieszczone w tabelach są bezpośrednio powiązane z wykresami znajdującymi się poniżej odnoszących się do testowania odpowiedzi sieci po przeprowadzonym procesie uczenia grupowania kwiatów ze względu na wszystkie kryteria razem to znaczy: wielkość działki kielicha jak i wielkość płatków.

Wykres 3. Wartości iloczynu skalarnego podczas procesu testowania kryterium ogólnego dla neuronu zwycięskiego dla współczynnika uczenia równego 0,0001

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić dwie grupy kwiatów. W przypadku tego wykresu powstałego przy małej wartości współczynnika uczenia widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup jest duża i z łatwością możemy je odszukać na wykresie.

Wykres 4. Wartości iloczynu skalarnego podczas procesu testowania kryterium ogólnego dla neuronu zwycięskiego dla współczynnika uczenia równego 0,001

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić dwie grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż poprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup się zmniejszyła co utrudnia nam rozpoznanie grup na podstawie wykresu, jednak dla sieci nie jest to żadne utrudnienie i w dalszym stopniu jest ona efektywna.

Wykres 5. Wartości iloczynu skalarnego podczas procesu testowania kryterium ogólnego dla neuronu zwycięskiego dla współczynnika uczenia równego 0,005

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić dwie grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż poprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup znowu się zmniejszyła co ponownie utrudnia nam rozpoznanie grup na podstawie wykresu, jednak dla sieci nie jest to żadne utrudnienie i w dalszym stopniu jest ona efektywna.

Wykres 6. Wartości iloczynu skalarnego podczas procesu testowania kryterium ogólnego dla neuronu zwycięskiego dla współczynnika uczenia równego 0,01

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić dwie grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż poprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup znowu się zmniejszyła co ponownie utrudnia nam rozpoznanie grup na podstawie wykresu, jednak dla sieci nie jest to żadne utrudnienie i w dalszym stopniu jest ona efektywna.

Wykres 7. Wartości iloczynu skalarnego podczas procesu testowania kryterium ogólnego dla neuronu zwycięskiego dla współczynnika uczenia równego 0,09

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić dwie grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż poprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup się zwiększyła i nawet z wykresu jesteśmy w stanie odczytać grupowanie kwiatów. W przypadku pracy programu dalej różnice są wystarczające i otrzymana sieć cechuje się świetną efektywnością.

Wykres 8. Wartości iloczynu skalarnego podczas procesu testowania kryterium ogólnego dla neuronu zwycięskiego dla współczynnika uczenia równego 0,1

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić dwie grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż poprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup znajduje się na podobnym poziomie jak uprzednio, a sieć cechuje się dalej 100% poprawnością odpowiedzi.

Poniższe wykresy odnoszą się do wartości współczynnika skalarnego uzyskiwanego przez zwycięski neuron w procesie testowania przydziału do grupy na podstawie kryterium wielkości działki kielicha.

Wykres 9. Wartości iloczynu skalarnego podczas procesu testowania kryterium wielkości działki kielicha dla neuronu zwycięskiego dla współczynnika uczenia równego 0,0001

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić dwie grupy kwiatów. W przypadku tego wykresu powstałego przy małej wartości współczynnika uczenia widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup jest znacznie mniejsza niż w przypadku identyfikacji na podstawie wszystkich kryterium i tej samej wartości współczynnika uczenia mimo to jednak sieć ta charakteryzuje się dużą skutecznością wynoszącą 93,33%.

Wykres 10. Wartości iloczynu skalarnego podczas procesu testowania kryterium wielkości działki kielicha dla neuronu zwycięskiego dla współczynnika uczenia równego 0,001

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić dwie grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż uprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup jest znacznie mniejsza niż w przypadku identyfikacji na podstawie wszystkich kryterium i tej samej wartości współczynnika uczenia, a także mniejsza w porównaniu z poprzednim wykresem co znalazło swoje odzwierciedlenie w gorszej efektywności sieci wynoszącej 84,44%.

Wykres 11. Wartości iloczynu skalarnego podczas procesu testowania kryterium wielkości działki kielicha dla neuronu zwycięskiego dla współczynnika uczenia równego 0,005

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić dwie grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż uprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup jest większa niż uprzednio, a także w porównaniu z pierwszym współczynnikiem uczenia, dlatego też nasza się cechuje się lepszą skutecznością wynoszącą w tym przypadku 95,55%.

Wykres 12. Wartości iloczynu skalarnego podczas procesu testowania kryterium wielkości działki kielicha dla neuronu zwycięskiego dla współczynnika uczenia równego 0,01

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić dwie grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż poprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup znowu się zmniejszyła co ponownie utrudnia nam rozpoznanie grup na podstawie wykresu. W przypadku tego współczynnika procent poprawnych odpowiedzi podczas procesu uczenia widocznie zmalał w porównaniu z wcześniejszym współczynnikiem, co posiada swoje odwzajemnienie na tym wykresie, gdzie wszystkie punkty posiadają wartości bardzo dla siebie zbliżone, a sama sieć osiągnęła najgorszą skuteczność, równą jednak zadowalające 80%.

Wykres 13. Wartości iloczynu skalarnego podczas procesu testowania kryterium wielkości działki kielicha dla neuronu zwycięskiego dla współczynnika uczenia równego 0,09

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić dwie grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż uprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup jest większa niż uprzednio, a także, iż jest ona bardzo zbliżona do tych uzyskiwanych przy współczynniku uczenia równym 0,005, co znajduje swoje uzasadnienie w takiej samej skuteczności sieci równej 95,55%.

Wykres 14. Wartości iloczynu skalarnego podczas procesu testowania kryterium wielkości działki kielicha dla neuronu zwycięskiego dla współczynnika uczenia równego 0,1

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić dwie grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż uprzednio widzimy, iż różnica iloczynów skalarnych jest bardzo zbliżona do tych uzyskiwanych przy współczynniku uczenia równym 0,005 oraz 0,09, co znajduje swoje uzasadnienie w takiej samej skuteczności sieci równej 95,55%.

Poniższe wykresy odnoszą się do wartości współczynnika skalarnego uzyskiwanego przez zwycięski neuron w procesie testowania przydziału do grupy na podstawie kryterium wielkości płatków.

Wykres 15. Wartości iloczynu skalarnego podczas procesu testowania kryterium wielkości płatków dla neuronu zwycięskiego dla współczynnika uczenia równego 0,0001

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Trudno tutaj mówić o wyróżnieniu jakiejkolwiek z grup kwiatów. W przypadku tego wykresu powstałego przy małej wartości współczynnika uczenia widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup jest zbyt mała, aby możliwe było efektywne grupowanie kwiatów. Znajduje to swoje odzwierciedlenie w efektywności sieci wynoszącej w tym przypadku jedynie 33,33%.

Wykres 16. Wartości iloczynu skalarnego podczas procesu testowania kryterium wielkości płatków dla neuronu zwycięskiego dla współczynnika uczenia równego 0,001

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić trzy grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż poprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup jest zdecydowanie większa niż ostatnio, w wyniku czego sieć o wiele lepiej radzi sobie z grupowaniem kwiatów osiągając skuteczność na poziomie 73,33%.

Wykres 17. Wartości iloczynu skalarnego podczas procesu testowania kryterium wielkości płatków dla neuronu zwycięskiego dla współczynnika uczenia równego 0,005

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić trzy grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż poprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup jest minimalnie mniejsza niż ostatnio, w wyniku czego sieć radzi sobie minimalnie gorzej z grupowaniem kwiatów osiągając skuteczność na poziomie 71,11%.

Wykres 18. Wartości iloczynu skalarnego podczas procesu testowania kryterium wielkości płatków dla neuronu zwycięskiego dla współczynnika uczenia równego 0,01

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić trzy grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż poprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup jest zdecydowanie większa niż ostatnio, a także zauważalnie większa niż w przypadku współczynnika uczenia równego 0,001 w wyniku czego sieć o wiele lepiej radzi sobie z grupowaniem kwiatów osiągając skuteczność na poziomie 84,44%.

Wykres 18. Wartości iloczynu skalarnego podczas procesu testowania kryterium wielkości płatków dla neuronu zwycięskiego dla współczynnika uczenia równego 0,09

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić trzy grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż poprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup jest minimalnie mniejsza niż ostatnio, w wyniku czego sieć radzi sobie minimalnie gorzej z grupowaniem kwiatów osiągając skuteczność na poziomie 80%.

Wykres 19. Wartości iloczynu skalarnego podczas procesu testowania kryterium wielkości płatków dla neuronu zwycięskiego dla współczynnika uczenia równego 0,1

Powyższy wykres przedstawiający wartości iloczynu skalarnego dla wygrywającego neuronu odzwierciedla przypisanie różnych kwiatów do poszczególnych grup. Możemy na nim wyróżnić trzy grupy kwiatów. W przypadku tego wykresu powstałego przy większej wartości współczynnika uczenia niż poprzednio widzimy, iż różnica iloczynów skalarnych dla poszczególnych grup jest minimalnie większa niż ostatnio, a także porównywalna z przypadkiem gdy współczynnik uczenia miał wartość 0,01 w wyniku czego sieć osiąga taki sam rezultat ze skutecznością na poziomie 84,44%.

1. **Wnioski**

* Efektywność sieci Kohonena opartej na regule WTA, jej zdolność do wskazywania poprawnych odpowiedzi jest powiązana z jej budową. Po wielokrotnych testach różnych konfiguracji stwierdzono, że dla powyższego problemu najlepszą skuteczność przy jednoczesnym małym rozbudowaniu sieci uzyskano przy ilości neuronów równej 9.
* Efektywność procesu uczenia dla sieci neuronowej jest silnie powiązana z wartością współczynnika uczenia sieci. Należy na podstawie wielokrotnych testów dobrać takie wartości współczynników, aby uniknąć bardzo powolnego procesu uczenia, ale też, należy uważać, żeby nasza sieć się nie przeuczyła, a bardzo szybka modyfikacja wag nie prowadziła do błędnych odpowiedzi sieci. W przypadku mojej sieci zauważyłem, że często wyższa wartość współczynnika pomimo gorszych rezultatów w procesie uczenia osiągała ona lepsze wyniki w procesie testowania opartym na danych nie zawierających się w procesie uczenia. Pozwala to więc stwierdzić, iż taka sieć będzie lepiej przygotowana do rozwiązywania problemów, grupowania kwiatów z którymi nie miała do tej pory styczności.
* Sieć Kohonena oparta o regułę WTA sprawia, iż pomimo dużej liczby neuronów tylko kilka z nich pozostaje aktywnych i bierze aktywny udział w rozwiązywanie danego zadania. Nic jednak nie stoi na przeszkodzie, aby pozostałe neurony uaktywniły się przy innym zadaniu co pozwala wnioskować, iż sieć potrafi w dobry sposób adaptować się do nowych zadań, co zresztą wynika pośrednio z bardzo dobrych wyników testowania przeprowadzonych na innych próbkach niż te dostarczone w procesie uczenia sieci.
* Wraz ze zwiększająca się wartością współczynnika uczenia możemy zaobserwować mniejsze różnice w wartościach iloczynu skalarnego uzyskiwanego przez zwycięski neuron. Pomimo tego, że wartości te są bardzo do siebie zbliżone sieć jest w stanie dalej poprawnie identyfikować grupy. W przypadku jednak gdy wartość współczynnika będzie zbyt duża, wartości te staną się zbyt sobie bliskie i sieć stanie się nieefektywna.

1. **Kod programu**

**Layer.h:**

#pragma once

#include<vector>

#include"Neuron.h"

using namespace std;

class Layer

{

public:

Layer(int numberOfNeurons, int amountOfDendrites, int amountOfOutputs, double learningCoefficient);

vector<Neuron> neurons;

int getNumberOfNeurons() { return \_numberOfNeurons; }

void changeNeuronWeight(bool testing);

double getScalarProduct(int index) { return \_scalarProducts[index]; }

void gatherScalarProducts();

int getWinnerNeuronIndex(){ return \_winnerNeuronIndex; }

private:

void foundMaxScalarProduct(bool testing);

int \_numberOfNeurons;

vector<double> \_scalarProducts;

int \_winnerNeuronIndex;

};

**Layer.cpp:**

#include"Layer.h"

Layer::Layer(int numberOfNeurons, int amountOfDendrites, int amountOfOutputs, double learningCoefficient)

{

\_numberOfNeurons = numberOfNeurons;

neurons.resize(numberOfNeurons);

for (int i = 0; i < numberOfNeurons; i++)

neurons[i].Neuron::Neuron(amountOfDendrites, amountOfOutputs, learningCoefficient);

}

void Layer::changeNeuronWeight(bool testing)

{

gatherScalarProducts();

foundMaxScalarProduct(testing);

}

void Layer::gatherScalarProducts()

{

\_scalarProducts.clear();

for (int i = 0; i < \_numberOfNeurons; i++)

\_scalarProducts.push\_back(neurons[i].countScalarProduct());

}

void Layer::foundMaxScalarProduct(bool testing)

{

double tmp = \_scalarProducts[0];

\_winnerNeuronIndex = 0;

for (int i = 1; i < \_scalarProducts.size(); i++)

{

if (tmp < \_scalarProducts[i])

{

\_winnerNeuronIndex = i;

tmp = \_scalarProducts[i];

}

}

neurons[\_winnerNeuronIndex].processOutput();

if (testing == false)

neurons[\_winnerNeuronIndex].countNewWeights();

}

**Neuron.h:**

#pragma once

#include <iostream>

#include <vector>

using namespace std;

class Neuron

{

public:

void createInputs(int amountOfDendrites, int amountOfSynapses);

void createInput() { \_inputs.push\_back(0); }

void createWeight(int index) { \_weights.push\_back(0); }

int getInputsAmount() { return \_inputs.size(); }

int getWeightsAmount() { return \_weights.size(); }

double getInput(int index) { return \_inputs[index]; }

void setInput(int index, double value) { \_inputs[index] = value; }

double getSynapse(int index) { return \_weights[index]; }

void setSynapse(int index, double value) { \_weights[index] = value; }

double getSumOfAllInputs() { return \_sumOfAllInputs; }

double getOutputValue() { return \_outputValue; }

double processSingleInput(int index) { return \_inputs[index] \* \_weights[index]; }

void processOutput();

void countNewWeights();

double countScalarProduct();

Neuron();

Neuron(int amountOfDendrites, int amountOfOutputs, double learningCoefficient);

private:

double countFirstWeight();

void normalizeWeights();

vector<double> \_inputs;

vector<double> \_weights;

double \_sumOfAllInputs;

double \_outputValue;

double \_learningCoefficient;

};

**Neuron.cpp:**

#include "Neuron.h"

#include <time.h>

#include <math.h>

Neuron::Neuron()

{

\_inputs.resize(0);

\_weights.resize(0);

\_sumOfAllInputs = 0.0;

\_outputValue = 0.0;

\_learningCoefficient = 0.0;

}

Neuron::Neuron(int amountOfDendrites, int amountOfOutputs, double learningCoefficient)

{

createInputs(amountOfDendrites, amountOfOutputs);

normalizeWeights();

\_learningCoefficient = learningCoefficient;

\_sumOfAllInputs = 0.0;

\_outputValue = 0.0;

}

void Neuron::createInputs(int amountOfDendrites, int amountOfOutputs)

{

for (int j = 0; j < amountOfDendrites; j++)

{

\_inputs.push\_back(0);

\_weights.push\_back(countFirstWeight());

}

}

double Neuron::countScalarProduct()

{

\_sumOfAllInputs = 0.0;

for (int i = 0; i < getInputsAmount(); i++)

\_sumOfAllInputs += \_inputs[i] \* \_weights[i];

return \_sumOfAllInputs;

}

void Neuron::processOutput()

{

double beta = 1.0;

\_outputValue = (1.0 / (1.0 + (exp(-beta \* \_sumOfAllInputs))));

}

void Neuron::countNewWeights()

{

for (int i = 0; i < getWeightsAmount(); i++)

\_weights[i] = \_weights[i] + (\_learningCoefficient \* (\_inputs[i] - \_weights[i]));

normalizeWeights();

}

//private methods

double Neuron::countFirstWeight()

{

double max = 1.0;

double min = 0.0;

double weight = ((double(rand()) / double(RAND\_MAX)) \* (max - min)) + min;

return weight;

}

void Neuron::normalizeWeights()

{

double vectorLength = 0.0;

for (int i = 0; i < getWeightsAmount(); i++)

vectorLength += pow(\_weights[i], 2);

vectorLength = sqrt(vectorLength);

for (int i = 0; i < getWeightsAmount(); i++)

\_weights[i] /= vectorLength;

}

**main.cpp:**

#include <iostream>

#include <vector>

#include "Layer.h"

#include <time.h>

#include <fstream>

using namespace std;

void setInputValue(Neuron& neuron, vector< vector<double> > inputData, int numberOfEntrances, int inputDataRow);

void learn(Layer& layer, vector< vector<double> > inputData);

void test(Layer& layer, vector< vector<double> > inputData);

void readLearningData(vector< vector<double> > &learningInputData, int numberOfEntrances, int amountOfOutputs);

void readTestingData(vector< vector<double> > &testingInputData, int numberOfEntrances, int amountOfOutputs);

fstream OUTPUT\_LEARNING\_FILE;

fstream OUTPUT\_TESTING\_FILE;

fstream OUTPUT\_TESTING\_NEURON;

fstream LEARNING\_DATA;

fstream TESTING\_DATA;

int main()

{

srand(time(NULL));

vector< vector<double> > learningInputData;

vector< vector<double> > testingInputData;

int numberOfNeurons = 9;

int numberOfEntrances = 4;

int numberOfOutputs = 1;

double learningCoefficient = 1.0;

Layer kohonenNetwork(numberOfNeurons, numberOfEntrances, numberOfOutputs, learningCoefficient);

readLearningData(learningInputData, numberOfEntrances, numberOfOutputs);

readTestingData(testingInputData, numberOfEntrances, numberOfOutputs);

do

{

cout << "1. Learn" << endl;

cout << "2. Test" << endl;

cout << "3. Exit" << endl;

int choice;

cin >> choice;

switch (choice)

{

case 1:

OUTPUT\_LEARNING\_FILE.open("output\_learning\_data.txt", ios::out);

for (int epochNumber = 1, i = 0; i < 50; i++, epochNumber++)

{

learn(kohonenNetwork, learningInputData);

OUTPUT\_LEARNING\_FILE << "EPOCH NUMBER: " << epochNumber << endl;

}

break;

case 2:

OUTPUT\_TESTING\_FILE.open("output\_testing\_data.txt", ios::out);

OUTPUT\_TESTING\_NEURON.open("output\_testing\_neuron.txt", ios::out);

test(kohonenNetwork, testingInputData);

break;

case 3:

OUTPUT\_LEARNING\_FILE.close();

OUTPUT\_TESTING\_FILE.close();

return 0;

}

} while (true);

return 0;

}

void setInputValue(Neuron& neuron, vector< vector<double> > inputData, int numberOfEntrances, int row)

{

for (int i = 0; i < numberOfEntrances; i++)

neuron.setInput(i, inputData[row][i]);

}

void learn(Layer& layer, vector< vector<double> > inputData)

{

static int counter = 0;

for (int inputDataRow = 0; inputDataRow < inputData.size(); inputDataRow++)

{

for (int i = 0; i < layer.getNumberOfNeurons(); i++)

{

setInputValue(layer.neurons[i], inputData, layer.neurons[i].getInputsAmount(), inputDataRow);

layer.neurons[i].countScalarProduct();

}

layer.changeNeuronWeight(false);

if (counter == 35)

{

counter = 0;

OUTPUT\_LEARNING\_FILE << "Next flower" << endl;

}

OUTPUT\_LEARNING\_FILE << layer.getWinnerNeuronIndex() << endl;

counter++;

}

}

void test(Layer& layer, vector< vector<double> > inputData)

{

static int counter = 0;

for (int inputDataRow = 0; inputDataRow < inputData.size(); inputDataRow++)

{

for (int i = 0; i < layer.getNumberOfNeurons(); i++)

{

setInputValue(layer.neurons[i], inputData, layer.neurons[i].getInputsAmount(), inputDataRow);

layer.neurons[i].countScalarProduct();

}

if (counter == 15)

{

counter = 0;

OUTPUT\_TESTING\_NEURON << "Next flower" << endl;

}

layer.changeNeuronWeight(true);

OUTPUT\_TESTING\_FILE << layer.neurons[layer.getWinnerNeuronIndex()].getSumOfAllInputs() << endl;

OUTPUT\_TESTING\_NEURON << layer.getWinnerNeuronIndex() << endl;

counter++;

}

}

void readLearningData(vector< vector<double> > &learningInputData, int numberOfEntrances, int amountOfOutputs)

{

LEARNING\_DATA.open("learning\_data\_all.txt", ios::in);

vector<double> row;

do

{

row.clear();

for (int i = 0; i < numberOfEntrances; i++)

{

double inputTmp = 0.0;

LEARNING\_DATA >> inputTmp;

row.push\_back(inputTmp);

}

double vectorLength = 0.0;

for (int i = 0; i < numberOfEntrances; i++)

vectorLength += pow(row[i], 2);

vectorLength = sqrt(vectorLength);

for (int i = 0; i < numberOfEntrances; i++)

row[i] /= vectorLength;

learningInputData.push\_back(row);

} while (!LEARNING\_DATA.eof());

LEARNING\_DATA.close();

}

void readTestingData(vector< vector<double> > &testingInputData, int numberOfEntrances, int amountOfOutputs)

{

TESTING\_DATA.open("testing\_data\_all.txt", ios::in);

vector<double> row;

while (!TESTING\_DATA.eof())

{

row.clear();

for (int i = 0; i < numberOfEntrances; i++)

{

double inputTmp = 0.0;

TESTING\_DATA >> inputTmp;

row.push\_back(inputTmp);

}

double vectorLength = 0.0;

for (int i = 0; i < numberOfEntrances; i++)

vectorLength += pow(row[i], 2);

vectorLength = sqrt(vectorLength);

for (int i = 0; i < numberOfEntrances; i++)

row[i] /= vectorLength;

testingInputData.push\_back(row);

}

TESTING\_DATA.close();

}