Politechnika Rzeszowska

Wydział Elektrotechniki i Informatyki

Katedra Informatyki i Automatyki

Rzeszów, 2016r.

Sztuczna Inteligencja

Projekt

TEMAT: Zrealizować sieć neuronową LVQ uczącą się rozpoznawania rodzaju wina

Wykonał: Mariusz Bożek
2 EF-DI
P 01

Spis treści

1 Przedstawienie problemu

Temat realizowanego projektu dotyczy sieci neuronowej LVQ uczącej się rozpoznawania rodzaju wina, na podstawie danych pobranych z bazy Uniwersytetu Kalifornijskiego w Irvine[1]. Dane te są wynikiem analizy chemicznej win pochodzących z tego samego regionu we Włoszech, ale wytwarzanych z trzech różnych odmian winogron. Analiza określa ilość każdego z trzynastu składników znajdujących się w każdym z trzech rodzajów win.

Charakterystyka danych	wieloczynikowa
Ilość danych	178
Ilość atrybutów	13
Brakujące dane	nie
Obszar	fizyczno-chemiczne

Tabela 1: Opis danych

Atrybuty:

1. Rodzaj wina (dane wyjściowe)
2. Ilość alkoholu
3. Kwas jabłkowy
4. Osad
5. Alkaliczność osadu
6. Magnez
7. Ilość wszystkich fenoli
8. Flawonoidy
9. Fenole niefluidalne
10. Proantocyjanidyny
11. Intensywność koloru
12. Odcień
13. OD280 / OD315 rozwodnionego wina
14. Prolina

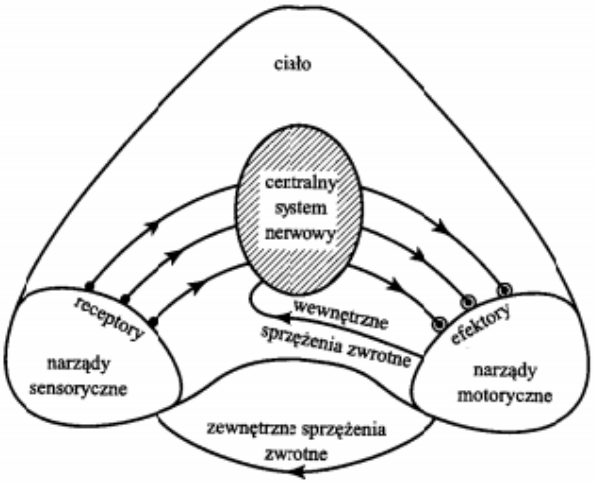
[1] http://archive.ics.uci.edu/ml/datasets/Wine

2 Opis sieci neuronowej i algorytmu uczenia

2.1 Czym jest sieć neuronowa?

Sztuczna sieć neuronowa (SSN) to zbiór jednostek obliczeniowym przetwarzających dane, komunikujących się ze sobą i pracujących równolegle. Z każdym połączeniem skoajrzona jest waga, która może zostać zmieniona w trakcie uczenia.

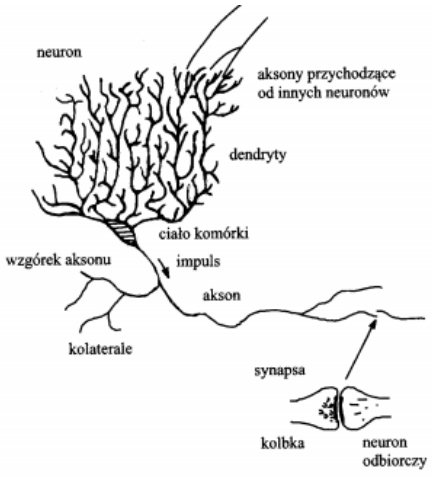
Ten matematyczny model został zainspirowany biologicznym działaniem neuronów istniejących w żywych istotach. Ludzki mózg składa się z około 10^{11} elementów przetwarzających sygnały zwanych neronami. Przekazują one pomiędzy sobą informacje w postaci impulsów elektrycznych za pomocą aksonów i synaps, których sieć ma gęstość około 10^4 na neuron. Sygnałów wejściowych do sieci dostarczają receptory. Sygnałami takimi mogą być bodźce z wnętrza ciała, jak również pobudzenia sensorów z zewnątrz organizmu. Bodźce te są przekazywane do sieci neuronowej, a następne przetwarzane są w centralnym systemie nerwowym. W wyniku tych działań otrzyumjemy sygnały strerujące efektorami, a więc określoną reakcję organizmu. Jest to trójstopniowy system, który steruje organizmem i jego działaniami.



Rysunek 1: Przepływ informacji w systemie nerwowym

2.2 Neuron biologiczny

Komórka nerwowa, inaczej neuron, to podstawowy element naturalnej sieci neuronowej. W typowej komórce rozróżniamy trzy podstawowe elementy: ciało komórki, akson i dentryty. Te ostatnie tworzą gęsto rozgałęzioną pajęczynę cienkich włókien wokół ciała neuronu. Informacja dociera do neuronu sąsiedniego za pośrednictwem aksonu, pojedyńczego włókna stanowiącego linię transmisyjną dla impulsów nerwowych.



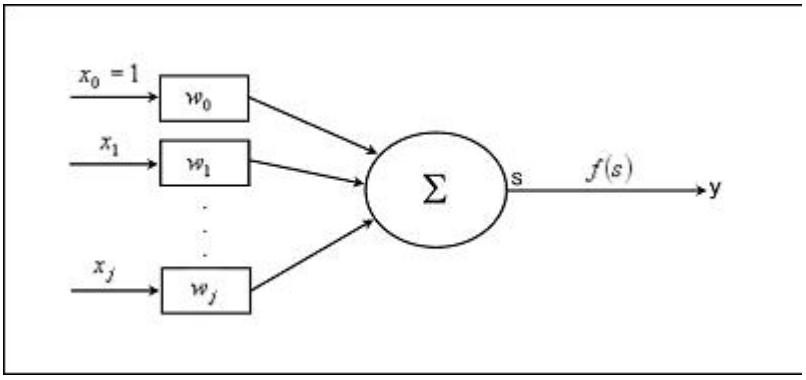
Rysunek 2: Schemat biologicznego neuronu

Poprzez złącze akson-dentryt (synapsę) przekazywane są syganły pomiędzy neuronami. W szczelinie pomiędzy synapsami przewodnictwo rzadko ma charakter elekryczny. Częściej polega to na uwolnieniu przez kolbkę odpowiedzniej substancji chemicznej. Ten zaś może wygenerować impuls do swojego aksonu, albo nie zareagować wcale.

Neuron reaguje stosownie do wartości wszystkich swoich pobudzeń skumulowanych w krótkim przedziale czasu . Neuron reaguje, jeżeli całkowity potencjał jego błony komórkowej osiąga określony poziom.

2.3 Sztuczny model neuronu

Na podstawie obserwacji działania naturalnych sieci neuronowych opracowano ich matematyczny model. Tak jak ich naturalne odpowiedniki, składają się z one z elementarnych komórek (neuronów). W uproszczeniu ziałanie takiego neuronu można opisać następująco: do neuronu na wejścia trafiają syganły. Każdy z nich ma jakąś wagę, która odpowiada za efektywność syanpsy. W neuronie obliczania jest ważona suma wejść i odejmowania jest wartość progowa. Następnie wynik tej sumy przekazywany jest jako argument funkcji aktywacji i jej wynik jest wyprowadzany na wyjście neuronu.



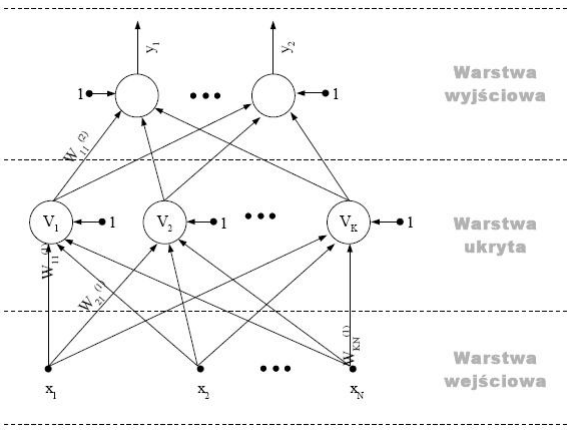
Rysunek 3: Model sztucznego neuronu

Na podstawie tej koncepcji, działanie pojedyńczego neuronu można opisać za pomocą wzoru:

$$y = f(\sum_{j=0}^{N} x_j * w_j)$$

2.4 Topologia sieci neuronowej

W SSN wyróżniamy grupę neuronów, które otrzymują wartości prezentowane sieci (neurony wejściowe; Rysunek 3), oraz grupę, która stonowi odpowiedź sieci (neurony wyjściowe). Wszystkie pozostałe elementy przetwarzające należą do tak zwanej warstwy ukrytej neuronów.



Rysunek 4: Prosty schemat sztucznej sieci neuronowej

Ogólnie wyróżnia się dwa typy architektur SSN:

1. **sieci jednokierunkowe** (ang. feedforwarded) tj. sieci o jednym kierunku przepływu sygnałów. Szczególnym przypadkiem architektury jednokierunkowej jest sieć warstwowa, reprezentująca zdecydowanie najpopularniejszą topologię;
2. Inne, np. **sieci rekurencyjne** (feedback, bidirectional) tj. sieci ze sprzężeniami zwrotnymi (sieć Hopfielda) albo sieci uczenia się przez współzawodnictwo (Kohonena)

Zasady łączenia neuronów między sobą

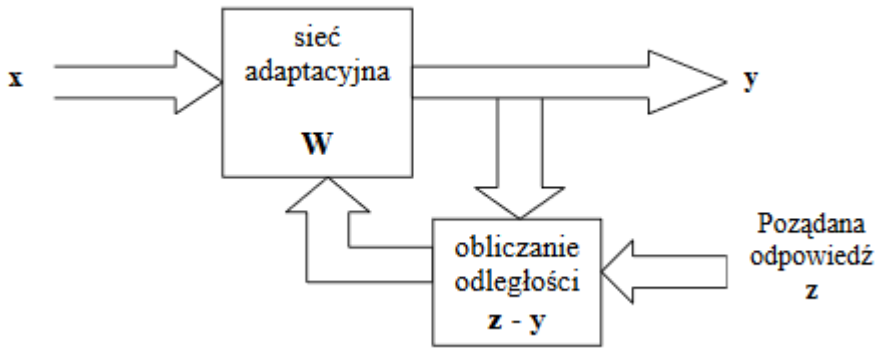
- "każdy z każdym",
- połączenia między kolejnymi warstwami w sieciach warstwowych
- tylko z pewną grupą neuronów, najczęściej z tzw. sąsiedztwem.

2.5 Charakterystyka procesu uczenia sieci

Odpowiedź sieci dla danego wejścia zależy od struktury połączeń i współczynników wagowych. Ogólnie rzecz biorąc sposoby połaczeń w danej sieci pozostają bez zmian, ale korekcji ulegają wagi, aby sieć mogła interpretować wiele funkcji. To w jaki sposób wagi ulegną modyfikacji zalezy od zadania, jakie zostanie przedstawione sieci oraz informacji dostępnych do nauki sieci.
Wyróżnia się dwa podstawowe sposoby uczenia sieci:

- uczenie nadzorowane (ang. supevised learning)
- uczenie nienadzorowane (ang. unsupervised learnig)

2.5.1 Uczenie nazdorowane



Rysunek 5: Schemat uczenia nadzorowanego

Dany jest zbiór przykładów uczących składający się z par wejście-wyjście (x_j,z_j), gdzie z_j jest pożądaną odpowiedzią sieci na sygnały wjeściowe x_j. Zadanie sieci jest nauczyć się możliwie jak najdokładniej funkcji przybliżającej powiązanie wejścia z wyjściem.
Odległość pomiędzy pożądaną a rzeczywistą odpowiedzią sieci jest miarą błędu, która jest używana do modyfikacji wag połączeń pomiędzy neuronami.

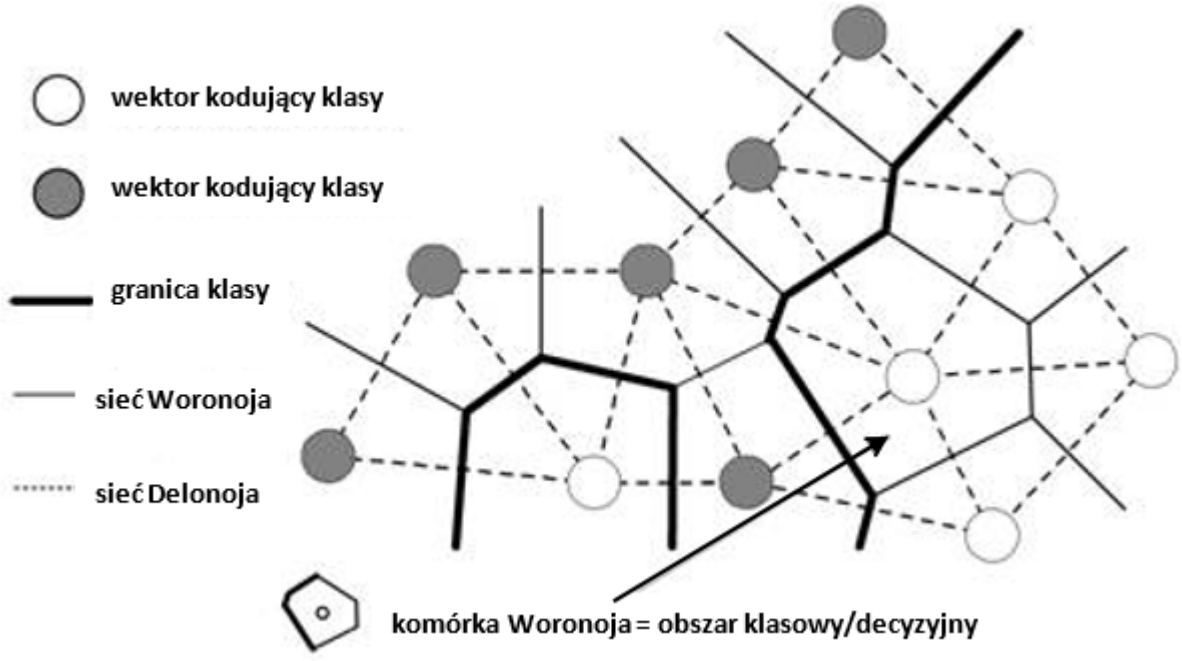
2.5.2 Uczenie nienadzorowane

Tutaj nie są znane konkretne przyporządkowania zestawów wejść. Sieć neuronowa stara się uporządkować pododne zestawy wejść w grupy. Wówczas tworzy się mapa, na której umieszczane są odpowiednie rekordy wejściowe.

2.6 Metoda LVQ

2.6.1 Ogólnie o metodzie

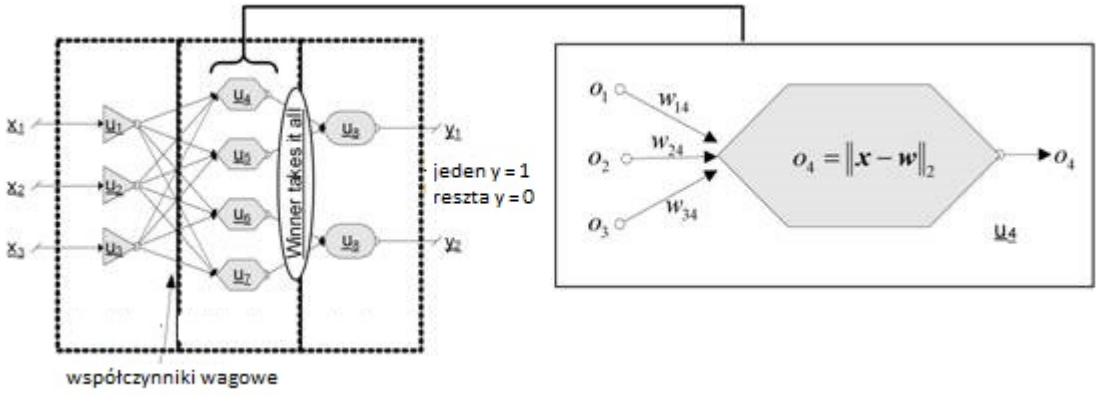
Sztuczne Sieci Neuronowe zazwyczaj służą do rozwiązywania problemów natury statystycznej, gdzie klasy wzorca nakładają się i muszą uwzględniać optymalną lokalizację granic decyzji. **Adaptacyjne kwantowanie wektorowe** (ang. Learning Vector Quantisation LVQ) definiuje bardzo dobre ich przybliżenia i ma małą złożoność obliczeniową.
Można ją zastosować do takich zadań jak rozpoznawanie obrazów, klasyfikacji wieloklasowej oraz do kompresji danych. W przeciwieństwie do Probabilistycznych sieci neuronowych czy nienadzorowanej wersji VQ, LVQ nie zaokroągla funkcji gęstości dla zbioru próbek klasowych, tylko definiuje granice klas na podstawie prototypów, których parametry określane są za pomocą algorytmów "najbliższy sąsiad" i "zwycięzca bierze wszystko".
Na początku przestrzeń próbek pokrywana jest wektorami kodującymi. Każdy znich ma reprezentować obszar oznaczony daną klasą. Każdu taki wektor może być uważany za prototyp elementu danej klasy.



Rysunek 6: Teselacja przestrzeni wejściowej w regiony klasowe(decyzyjne)

Obszar danych wejściowych jest dzielony na komórki decyzyjne. Podział dokonuje się na płaszczyznach prostopadłych do linii łączących dwa wektory kodujące (liniach sieci Delonoja; kreska przerywana na Rysunku 6). Płaszczyzny te są w przypadku dwóch wymiarów prostymi (linia ciągła), tworzącymi sieć Woronoja.

2.6.2 Topologia sieci LVQ



Rysunek 7: Topologia sieci LVQ

Sieci LVQ należą do typu feedforward i posiadają jedną ukrytą warstwę neuronów, która jest w pełni połączona z wartstwą wejściową. Za wektor kodujący można uznać cały ukryty neuron albo wektor wag dla wszyystkich wejść danego neuronu.
Uczenie polega na modyfikacji wag zgodnie z regułami adaptacyjnymi, co powoduje zmianę pozycji wektora kodującego w przestrzeni wejściowej.Granice klas są odcinkami bedącymi symetralnymi linii sieci Delonoja. Przez to zmieniają się podczas uczenia. Teselacja spowodowana wektorami kodującymi jest optymalna tylko wtedy, gdy dane w jednej komórce rzeczywiście należą do jednej klasy. Zwycięską kategorią, zdefiniowaną zazwyczaj jako 'c', dla której wektor kodujący ma najmniejszą euklidesowską odległość od x:

$$|x - m_c| = min_i|x - m_i|$$

gdzie: x - współrzędna wektora wejściowego(signal vector)
m_i - współrzędna wektora kodującego (codebook vector)

Na podstawie tej zależności porównywany jest wektor wejściowy z reprezentantami klas. Mała odległość oznacza duże prawdobodobieństwo należenia danej próbki do reprezentanta klasy rodzimej dla najbliższego wektora kodującego.

2.7 Algorytmy uczenia

2.7.1 Podstawowy algorymt LVQ1

Podstawowy algorytm LVQ działa na zasadzie nagradzania dobrze sklasyfikowanych próbek przesuwając wektor kodujący w stronę danego wektora wejściowego. Natomniast złe klasyfikacje są "karane" oddaleniem go w przeciwnym kierunku. Wektor przesunięcia zależy od wartości współczynnika uczenia α_t.
LVQ1 jest najprostszym algorytmem stosowanym w sieciach LVQ. W niektórych przypadkach nie pozwala on na osiągnięcie zadowalającej dokładności sieci. Gdy z powodu losowego ustawiania wag początkowych dany wektor kodujący jest źle zlakalizowany, to algorytm nie ma miechanizmów pozwalających na na poprawienie jakości i szybkości pozbywania się takich złych przyporządkowań. Jego dobrą stroną jest szybki spadek błędu klasyfikacji w początkowych fazach uczenia.

1. Przedstaw sieci kolejny wektor wejściowy X_n

2. Porównaj klasy $C_X n$ przypisanej wektorowi X_n i klasy $C_W i$ przypisanej do wektora W_c (wektor kodujący).

3. Dokonaj kolekty wag neuronu zgodnie z zależnością:

   - Jeśli $C_X n = C_W i, to W_c(t) + \alpha_t[X_i - W_c(t)]$
   - Jeśli $C_(Xn) \neq C_W i, to W_c(t) - \alpha_t[X_i - W_c(t)]$

4. Pozostałe wektory nie ulegają zmianie.

2.7.2 Algorytm LVQ2.1

Jego działanie jest zbliżone do pierwszej wersji. Różnica polega na tym, że w procesie uczenia korzystamy z dwóch wektorów wzorcowych W_i oraz W_z, które sa najbliższe wektorowi X. Pierwszy z nich należy do tej samej klasy co X, a drugi do innej. Następnie oba wektory są odpowiednio traktowane: ten należący do tej samej klasy jest przybliżany do próbki X, a ten z innej jest oddalany.

1. Wyznacz okno do którego wpada wektor wejściowy zgodnie z zależnością:

   - $\min(d_i/d_z, d_z/d_i) > s, s = (i - w)/(i + w), w = 0.2..0.3$

2. Adaptuj odpowiednio wagi:

   - $W_z(t + 1) = W_z(t) - \alpha_t[X_k - W_z(t)]$
   - $W_i(t + 1) = W_i(t) + \alpha_t[X_k - W_i(t)]$
     Gdzie:
   - z - wektor o innej klasie niż X
   - i - wektor o tej samej klasie co X
   - d_z, d_i - euklidesowe odległości od X dla wektora z oraz i
   - - parametr nazywany szerokością "okna"

Algorytm ten jest najczęściej stosowany dopiero po uczeniu sieci metodą LVQ1.

2.7.3 Algorytm LVQ3

Poprzedni algorytm nie uwzględniał sprawdzania warunku, czy wektor W_z w trakcie procesu uczenia nie zmienia swego położenia tak, że przestaje być dobrym aproksymatorem klasy, do której należy. Potrzeba wprowadzenia takiego zabezpiecznenia na wypadek tej sytuacji doprowadziła do stworzenia wersji trzeciej tej metody.

Algorytm ten wprowadza dodatkową zależność:

$$W_k(t + 1) = W_k(t) + \varepsilon\alpha(t)[X(t) - W_k(t)]$$

Gdzie:

$k\epsilon$\{z,i\} gdy X, W_z, W_i należą do tej samej klasy.

Optymalną wartość ε znajduje się drogą prób i błędów. Jest ona wprost proporcjonalna do szerokości "okna" (parametr w). Udoskonalenie dodane do tej metody w wielu przypadkach poprawia zbieżność w stosunku do algorytmu LVQ2.1

3 Wykonanie

3.1 Przygotowanie danych

Dane zostały pobrane ze strony podanej przez prowadzącego. Plik zawiera 14 kolumn, gdzie pierwsza to klasy (target), do których przypisane są odpowiednie wartości z pozostałych trzynastu kolumn. Tak zapisane dane zostały zaimportowane do programu MatLab (wersja 7.1) i rozdzielone do dwóch tablic (P i T), które zostały transponowane.
Kolejnym krokiem była normalizacja danych wejściowych zgodnie ze wzorem:

$$(ymax - ymin) * \frac{x - xmin}{xmax - xmin} + ymin$$

Normalizację wykonuje się, aby ujednolić dane wejściowe. Taki zabieg poprawia jakość uczenia sieci.

3.2 Sprzęt

Wszystkie eksperymety zostały wykonanie na komputerze typu PC o następujących parametrach:

- Procesor Intel Core i3 1.90GHz
- RAM: 4.00 GB
- System Operacyjny: Windows 7 Pro
- Wersja Matlaba: 7.1 (2012r.)

3.3 Eksperymenty

3.3.1 Eksperyment 1: Zestawienie wyników przy iteracyjnej zmianie wszystkich parametrów

Wykorzystując skrypt, który przedstawiony jest w punkie (3.4) zmieniałem w pętli parametry uczenia, to jest: ilość neuronów w warstwie ukrytej (S1), liczbę epok, oraz współczynnik uczenia.
Skrypt wykonywał się około sześciu godzin. Otrzymałem łącznie 3500 tysiąca wyników. Najwyższy wynik skuteczności uczenia otrzymałem dla parametrów:

- S1 = 85
- Współczynnik uczenia = 0,3
- Liczba epok: 2000

I wynosił on 88,20%. Z kolei najniższy wynik pojawił się przy parametrach:

- S1 = 5
- Współczynnik uczenia = 0,1
- Liczba epok: 7000

Wyniósł on wtedy tylko 47,75%

W poniższych tabelach jest pokazanych kolejno 12 nagorszych i najlepszych wyników.

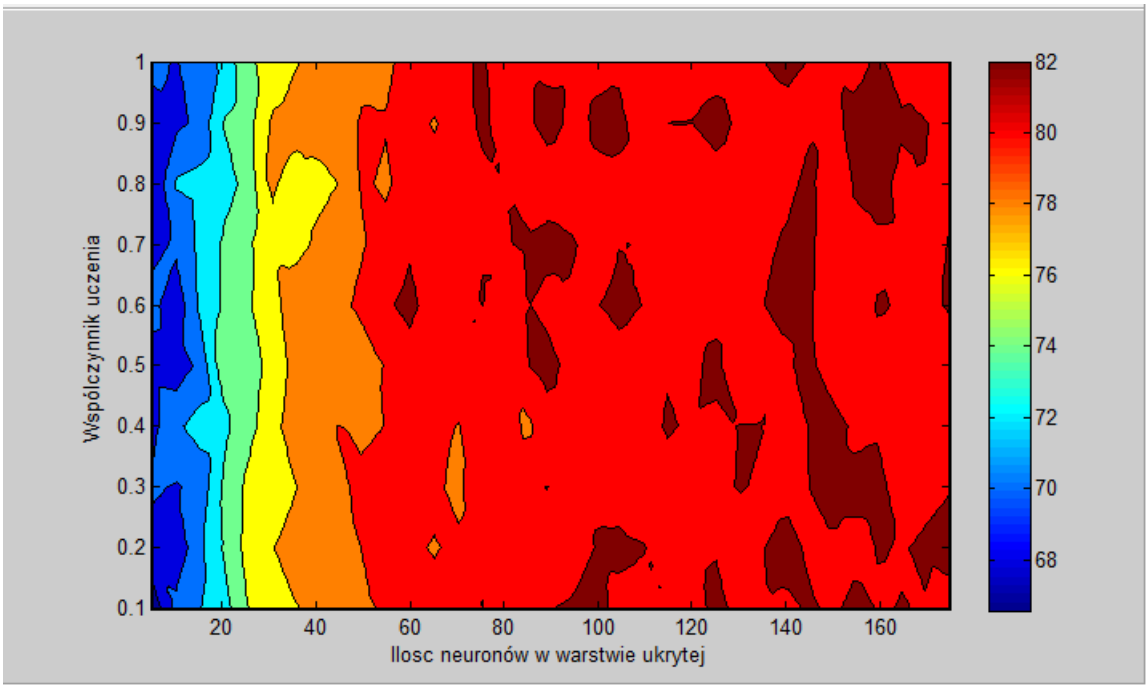
Liczba neuronów	Współczynnik uczenia	Liczba epok	Skuteczność uczenia(%)
5	0,1	7000	47,75
10	0,6	9000	56,74
5	0,8	3000	57,30
5	0,1	10000	60,11
10	0,5	3000	61,24
5	0,4	5000	62,92
10	0,2	6000	62,92
10	0,9	5000	62,92
10	0,6	3000	63,48
5	0,6	6000	64,04
5	0,9	3000	64,04
5	0,2	1000	64,61

Tabela 2: Najgorsze wyniki przy zmianie wszystkich parametrów

Liczba neuronów	Współczynnik uczenia	Liczba epok	Skuteczność uczenia(%)
145	0,5	6000	86,52
150	0,5	9000	86,52
160	0,4	1000	86,52
160	0,6	3000	86,52
160	0,8	4000	86,52
70	0,9	7000	87,08
110	0,1	5000	87,08
115	0,3	9000	87,08
140	0,2	8000	87,08
155	0,3	10000	87,08
125	0,2	6000	87,64
125	0,5	1000	87,64
85	0,3	2000	88,20

Tabela 3: Najlepsze wyniki przy zmianie wszystkich parametrów

Jak możemy zauważyć skuteczność uczenia zależy tylko od zmiany liczby neuronów w warstwie ukrytej. Przy tych samych wynikach (zarówno tych wysokich jak i niskich) występują te same wartości współczynnika uczenia i liczby epok. Możemy zauważyć to także na poniższym wykresie(rysunek 8):



Rysunek 8: Zależność skuteczności uczenia od liczby neuronów i współczynnika uczenia

Jak widzimy na tym wykresie, wysoka skuteczność uczenia zaczyna się w okolicach 50-55 neuronów w warstwie ukrytej, co stanowi niecałą 1/3 ilość wejść (178).

3.3.2 Eksperyment 2: Iteracyjna zmiana tylko dwóch parametrów: liczby neuronów i współczynnika uczenia

Jak zauważyliśmy podczas poprzedniego eksperymentu, skuteczność uczenia zmienia się tylko w zależności od liczby neuronów w warstwie ukrytej. Postanowiłem sprawdzić, jakie wyniki będzie dawała sieć przy zmianie tylko dwóch parametrów: liczby neuronów i współczynnika uczenia, przy liczbie epok równej 10000.
W Wyniku tego eksperymentu otrzymałem 4250 wyników. W tym przypadku najgorszy wynik sieć osiągneła przy parametrach:

- S1 = 10
- Współczynnik uczenia = 0,27
- Skuteczność uczenia = 63,48

Najlepszy natomiast osiągnęła przy parametrach:

- S1 = 176
- Współczynnik uczenia = 0,29
- Skuteczność uczenia = 88,76

Pięć najlepszych i najgorszych wyników zostało przedstawionych w następnych tabelach.

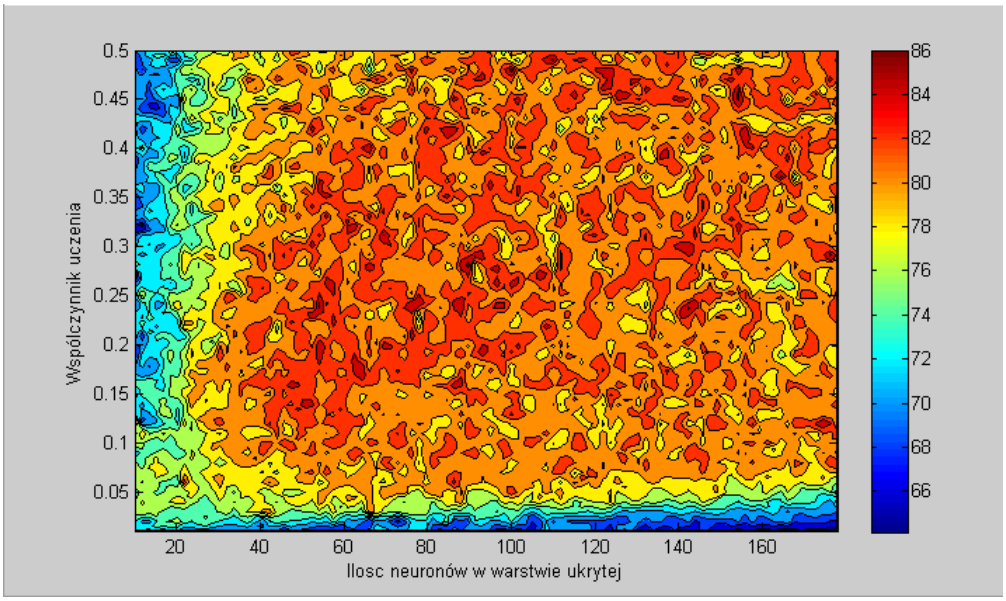
Liczba neuronów	Współczynnik uczenia	Skuteczność uczenia (%)
10	0,27	63,48
12	0,32	63,48
166	0,01	64,04
154	0,01	64,61
160	0,01	64,61

Tabela 4: Najgorsze wyniki przy zmianie liczby neuronów i współczynnika uczenia

Liczba neuronów	Współczynnik uczenia	Skuteczność uczenia (%)
154	0,46	87,08
164	0,37	87,08
174	0,12	87,08
70	0,31	87,64
176	0,29	88,76

Tabela 5: Najgorsze wyniki przy zmianie liczby neuronów i współczynnika uczenia

Jak możemy zauważyć tym razem, w najniższych wartościach skuteczności uczenia możemy spotkać dużą ilość neuronów warstwy ukrytej. Świadczy to o tym, że sieć LVQ ma także małą podatność na zmianę liczby neuronów, szczególnie w górnych wartościach. Jak widać na wykresie (Rysunek 9.) najgorsze wyniki występują przy niewielkiej liczbie neuronów i małym współczynniku uczenia (niebieski kolor).



Rysunek 9: Zależność skuteczności uczenia od liczby neuronów i współczynnika uczenia (Eksperyment 2)

3.3.3 Eksperyment 3. Iteracyjna zmiana współczynnika uczenia i liczby epok

Wyciągając wnioski z dwóch poprzednich eksperymentów postanowiłem przeprowadzić jeszcze jeden. Tym razem liczbę neuronów pozostawiłem taką samą (85, najlepszy wynik z pierwszego eksperymentu) i zmieniałem tylko liczbę epok i współczynnik uczenia.
Uzyskałem w ten sposób następujące wyniki:

- Liczba epok: 1000
- Skuteczność uczenia (średnia, dla wszystkich współczynników uczenia): 74,39%

- Liczba epok: 2000
- Skuteczność uczenia: 78,88%

- Liczba epok: 3000
- Skuteczność uczenia: 79,61%

- Liczba epok: 4000
- Skuteczność uczenia: 80,94%

- Liczba epok: 5000
- Skuteczność uczenia: 81,35%

- Liczba epok: 6000
- Skuteczność uczenia: 80,76%

- Liczba epok: 7000
- Skuteczność uczenia: 81,69%

- Liczba epok: 8000
- Skuteczność uczenia: 81,51%

- Liczba epok: 9000
- Skuteczność uczenia: 81,47%

- Liczba epok: 10000
- Skuteczność uczenia: 81,62%

Eksperyment ten przyniósł tylko potwierdzenie wniosków z poprzednich dwóch. Średnia skuteczność uczenia przy 85 neuronach wyniosła około 81% .Najniższa wartość wystąpiła przy 1000 epokach (74,38%), a najwyższa przy 10000 (81,62%). Jak widać, zmiana liczby epok czy współczynnika uczenia ma rzeczywiście niewielki wpływ na działanie sieci LVQ.

3.4 Skrypt

```
%LVQ Neural network training

clear all

nntwarn off
load wine.data

P = wine(:,2:14);
T = wine(:,1);

P=P';
T=T';
maxP=max(P);
minP=min(P);
Pn=zeros(size(P));
for i=1:length(maxP),
     Pn(:,i) = (1-(-1))*(P(:,i)-minP(i))/(maxP(i)-minP(i)) + (-1);
%      (ymax-ymin)*(x-xmin)/(xmax-xmin) + ymin;
end

nr =1;
S1=10:2:length(T),
for liczba_epok=[1000:1000:10000]
    for wsp_uczenia=[0.1:0.05:0.5]


TP = [1000 liczba_epok wsp_uczenia];
CVTrainTarget = ind2vec(T);
CVTrain = P;
[w1, w2]= initlvq(CVTrain, S1, CVTrainTarget);

[w1, w2] = trainlvq(w1, w2, CVTrain, CVTrainTarget, TP);
a = simulvq(CVTrain, w1, w2);
a = vec2ind(a);

[T' a' (T-a)' (abs(T-a)>0.5)']
skutecznosc_uczenia = (1-sum(abs(T-a)>0.5)/length(P))*100;

    tab_w(nr,1)=S1;
        tab_w(nr,1)=wsp_uczenia;
        tab_w(nr,2)=liczba_epok;
        tab_w(nr,3)=skutecznosc_uczenia;
        nr = nr+1;
end;
end;
```

3.5 Opis funkcji użytych w skrypcie

- **ind2vec** - funkcja zwracająca macierz zawierająca dokładnie jedną jedynkę w każdej kolumnie. Pozostałe współrzędne każdego wektora są zawsze zerowe, konwertuje wskaźniki na wektory np.:
  ind = [1 3 2 3]
  vec(1,1)=1
  (3,2)=1
  (2,3)=1
  (3,4)=1

- **vec2ind** - funkcja zwracająca indeksy współrzędnych wektorów kolumnowych, które mają jednostkową wartość. Pozostałe współrzędne każdego wektora są zawsze zerowe, np.:
  vec =
  1 0 0 0
  0 0 1 0
  0 1 0 1
  ind =[1 3 2 3]

- **initlvq(CVTrain, S1, CVTrainTarget)** - przyjmuje dane wejściowe i zwraca ustawienia inicjalizacji współczynników wagowych warstw neuronów z rywalizacją (konkurencyjna warstwa),

- **[w1, w2] = trainlvq (w1, w2, CVTrain, CVTrainTarget, TP)** - szkolenie sieci (W1 - wagi na konkurencyjnej warstwie, W2 - wagi warstwy liniowej, CVTrainTarget - macierz docelowa, TP - parametry treningowe), uczenie jest przeprowadzane przez tak zwany trening w przeciwieństwie do konwencjonalnych metod programistycznych SSN nie są projektowane na zasadach "jeśli to" ,a właśnie trenowane.,

- **simulvq(CVTrain, w1, w2)** - używana do symulacji przeszkolenia sieci.

4 Podsumowanie i wnioski

Po wykonaniu tych eksperymentów stwierdzam, że najlepszy dobór parametrów to:
Eksperyment 1

- Liczba neuronów: 85
- Współczynnik uczenia: 0,3
- Liczba epok: 2000
- Skuteczność: 88,20%

Eksperyment 2

- Liczba neuronów: 176
- Współczynnik uczenia: 0,29
- Liczba epok: 10000
- Skuteczność: 88,76%

Eksperyment 3

- Liczba neuronów: 85
- Współczynnik uczenia: 0,22
- Liczba epok: 10000
- Skuteczność: 88,20%

Rozpoznawanie rodzaju wina na podstawie danych opisanych w rozdziale 1 za pomocą sieci LVQ zostało zrealizowanie, choć z nie do końca zadowalającym efektem. Maksymalną sprawność uczenia jaką udało to 88,75%. Śmiało można stwierdzić, że na sprawnośc uczenia większego wpływu nie miała liczba epok większa niż 1000. Największy za to wpływ miała zmiana liczby neuronów. Jednakże powyżej zwiększanie powyżej 60 nie dawało widocznej zmiany w wynikach sieci (wykres; Rysunek 8). Jeśli chodzi o współczynnik uczenia to najbardziej opytmalną wartością jest ok 0,3. Jak zauważyłem, zwiększanie tego współczynnika powyżej 0,5 nie daje pożądanych rezulatów.
Pomimo długości wykonywania skryptu dla pierwszego eksperymentu (ok 6 godz.) można stwierdzić, że metoda LVQ jest jedną z szybszych metod uczenia sieci neuronowych metodą uczenia nadzorowanego. Czas ten zależał raczej od zagęszczenia pętli (uzyskałem 3500 wyników).
Matlab Jest bardzo dobrym narzędziem do tworzenia i badania działania sieci neuronowych. Udostępnia od wiele przydatnych funkcji, które można wykorzystać do symulacji zachowań SSN.

5 Spis rysunków, tabel i wykresów

Spis rysunków

Spis tabel

6 Bibliografia

- http://www.cs.put.poznan.pl/jstefanowski/aed/TPDANN.pdf
- http://4programmers.net/Z_pogranicza/Sztuczne_sieci_neuronowe_i_algorytmy_genetyczne
- http://www.neural-forecasting.com/lvq_neural_nets.htm
- http://www.mathworks.com/help/nnet/ref/lvqnet.html
- "The Handbook of Brain Theory and Neural Networks", Teuvo K. Kohonen, *Learning Vector Quantization*
- Marek Jan Kasperski, *Sztuczna Inteligencja*, Gliwice, 2015.