Sprawozdanie II z laboratorium:
Uczenie Maszynowe i Sieci Neuronowe

Część I: Uczenie nadzorowane warstwowych sieci neuronowych
Część II: Rekurencyjne sieci neuronowe
Część III: Uczenie nienadzorowane warstwowych sieci neuronowych

21 lutego 2012

Prowadzący: dr inż. Maciej Komosiński

Autorzy:	**Krzysztof Urban**	inf84896	ISWD	krz.urb@gmail.com
	Tomasz Ziętkiewicz	inf84914	ISWD	tomek.zietkiewicz@gmail.com

Zajęcia poniedziałkowe; Krzysztof: 16:50, Tomasz: 13:30

1 Uczenie nadzorowane warstwowych sieci neuronowych

1.1 Różnice funkcjonalne pomiędzy sieciami jedno- i wielowarstwowymi oraz pomiędzy sieciami liniowymi a nieliniowymi (uczenie sieci warstwowych funkcji logicznej AND i funkcji różnicy symetrycznej XOR)

1. **Skonstruuj zbiór przykładów definiujący dwuargumentową funkcję ("bramkę") AND (File—New—Data set) i zachowaj go. Wszystkie 4 przykłady mają stanowić zbiór uczący. Jakie są klasy decyzyjne w tym zbiorze przykładów i jakie są ich liczności?**
   Zbiór uczący dla problemu "AND" pokazano w tabeli 1. Klasy decyzyjne to "0" i "1". Klasa "0" ma liczność 3 a klasa "1" - liczność 1.

Tabela 1: Zbiór uczący funkcji "AND"

Przykład	Zmienna 1	Zmienna 2	Decyzja
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

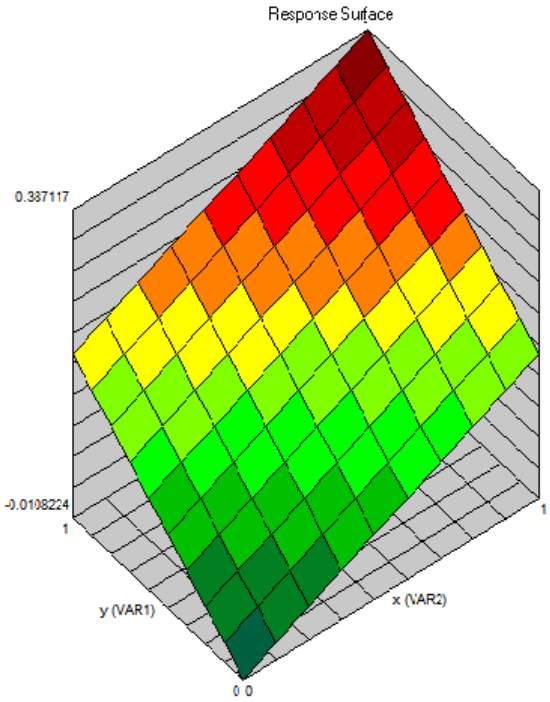
2. **Skonstruuj liniową sieć jednowarstwową o architekturze 2-1 (File—New—Network, Type=Linear, przycisk Advise). Uaktywnij okno wykresu błędu średniokwadratowego (Statistics—Training graph). Naucz sieć na problemie AND (Train—Multilayer perceptron—Back propagation). Obejrzyj funkcję odpowiedzi sieci (Run—Responce surface) i błędy dla poszczególnych przypadków (Statistics—Case errors).**

   Funkcję odpowiedzi sieci i błąd przez nią popełniany pokazano na rys. 1 i 2. Sieć nie radzi sobie z zadaniem - przez swą liniową naturę zawsze średnia wartość funkcji w punktach (0,1) i (1,0) będzie równa średniej wartości w punktach (0,0) i (1,1) przez co wagi mogą jedynie zmienić rozłożenie błędu między poszczególne przypadki uczące, ale zawsze zmniejszanine błędu w punktach (0,0), (0,1) i (1,0) będzie powodowało wzrost błędu w punkcie (1,1) i na odwrót. Przy zmniejszeniu prędkości uczenia błąd sieci można zredukować do ok 0.2.

3. **Spróbuj utworzyć sieć liniową dla problemu AND o liczbie warstw większej niż 2.**
   Porgram nie pozwala na utworzenie takiej sieci, ponieważ nie ma to sensu - po dodaniu dodatkowych warstw końcowa funkcja aktywacji będzie tylko kombinacją liniową funkcji poszczególnych neuronów. Dodatkowe warstwy są redundantne, ponieważ dla każdej liniowej sieci 3 i więcej warstwowej istnieje odpowiadająca jej sieć 2 warstwowa, której zachowanie jest identyczne.

4. **Przerób sieć na nieliniową sieć jednowarstwową (ustawiając Act fn w Edit—Network na Logistic) i naucz ją na tym samym problemie.**

   Sieć nielinowa poradziła sobie z problemem znacznie lepiej - błędy klasyfikacji bliskie zeru. W przeciwieństwie do sieci liniowej proces uczenia jest zbieżny - wraz z uczeniem błąd asymptotycznie maleje. Widać to na rysunku 4.



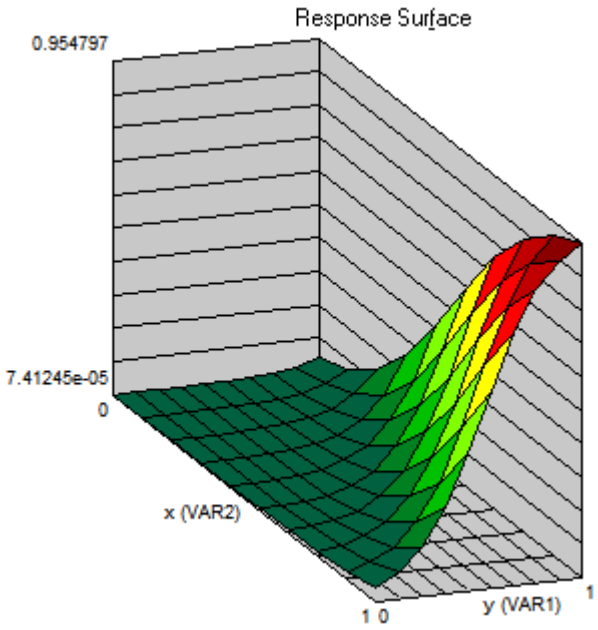
Rysunek 1: Funkcja odpowiedzi sieci liniowej. Problem "AND".

Rysunek 2: Błąd średniokwadratowy dla sieci liniowej. Problem "AND".

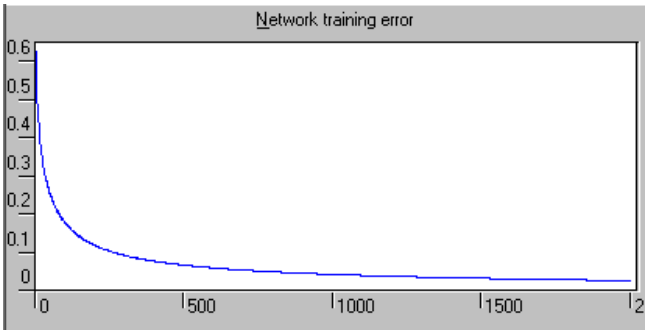
5. **Wejdź do edytora sieci (Network—Edit) i przypatrz się wagom neuronu wyjściowego. Wykreśl w przestrzeni wejść prostą, którą definiuje neuron wyjściowy i oceń, czy i jak realizuje ona separację klas decyzyjnych.**
   Prosta ta jest przedstawiona na rys. 6. Separuje ona prawidłowo punkty należące do klas 0 i 1. Jeśli za kryterium oceny separacji uznać maksymalizację minimalnej odległości prostej od separowanych punktów, to trzeba uznać, że realizuje ją dobrze, ponieważ odległość ta jest taka sama dla punktów (0,1), (1,0) i (1,1). Zmiana jakichkolwiek parametrów prostej, po której prosta ta wciąż separowałaby klasy, wiązałaby się ze zmniejszeniem minimalnej odległości punktów od prostej, więc ze względu na to kryterium prosta separująca jest optymalna.

6. **Skonstruuj zbiór przykładów definiujący dwuargumentową funkcję XOR i zachowaj go. Wszystkie 4 przykłady mają stanowić zbiór uczący.**

   Zbiór uczący dla funkcji XOR pokazano w tabeli 2.



Rysunek 3: Funkcja odpowiedzi sieci nieliniowej jednowarstwowej. Problem "AND".



Rysunek 4: Błąd średniokwadratowy dla sieci nieliniowej jednowarstwowej. Problem "AND".

7. **Skonstruuj liniową sieć jednowarstwową o architekturze 2-1 i naucz ją na problemie XOR.**

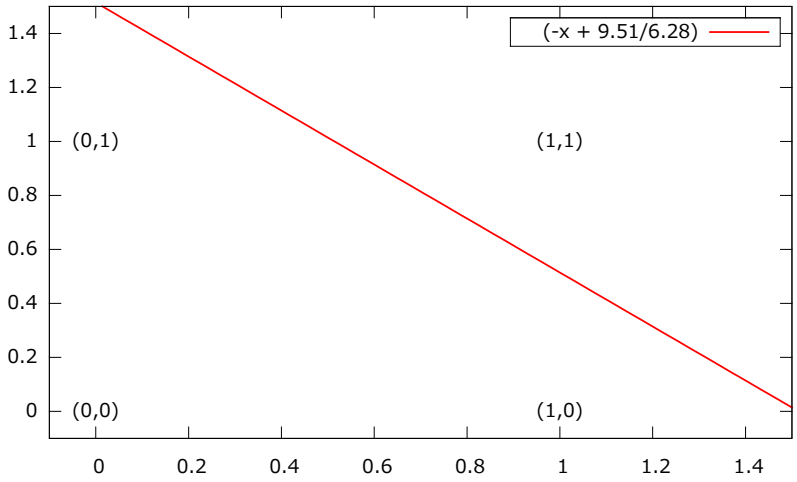
8. **Obejrzyj funkcję odpowiedzi sieci i błędy dla poszczególnych przypadków.**

   Jak widać na rysunkach 7, 8 i 9 sieć liniowa nie radzi sobie z zadaniem.

9. **Przerób sieć na nieliniową sieć jednowarstwową i naucz ją na tym samym problemie (zwróć uwagę, jak sieć stara się minimalizować błąd). Obejrzyj, jak zmienia się rozkład wag podczas procesu uczenia (Statistics—Weight distribution).**

   Błąd średniokwadratowy i funkcję odpowiedzi sieci nieliniowej jednowarstwowej dla problemu "XOR" przedstawiono na rysunkach 11 i 10. Sieć minimalizuje błąd w 3 punktach: (0,0), (0,1), (1,0), ale ponieważ ma tylko jedną warstwę nie może minimalizować błędu w czwartym punkcie: (1,1).

Rysunek 5: Błąd dla poszczególnych przypadków dla sieci nieliniowej jednowarstwowej. Problem "AND".



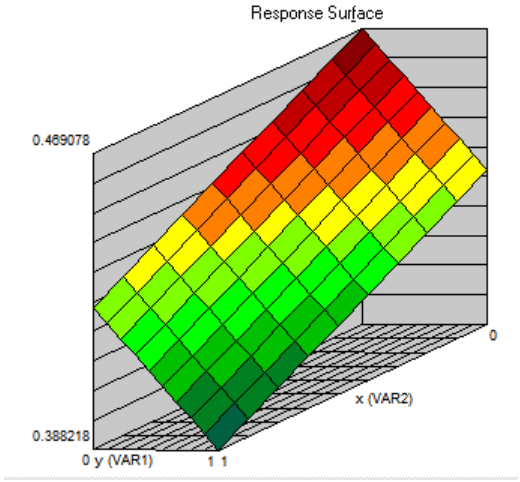
Rysunek 6: Prosta separująca klasy decyzyjne dla problemu "AND".

Tabela 2: Zbiór uczący funkcji "XOR"

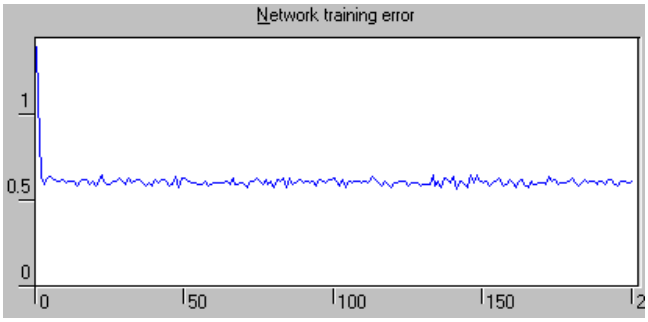
Przykład	Zmienna 1	Zmienna 2	Decyzja
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

10. **Skonstruuj nieliniową sieć dwuwarstwową o architekturze 2-2-1 (File—New—Network, Type=Multilayer perceptron) i naucz ją na problemie XOR. Obejrzyj funkcję odpowiedzi.**
Funkcję odpowiedzi i błąd średniokwadratowy przedstawiono na rys. 12 i 13. Jak widać, dzięki drugiej warstwie sieci funkcja odpowiedzi może lepiej dopasować się do danych uczących.

11. **Obejrzyj wagi sieci w edytorze sieci (Edit—Network). Jakie proste definiują**



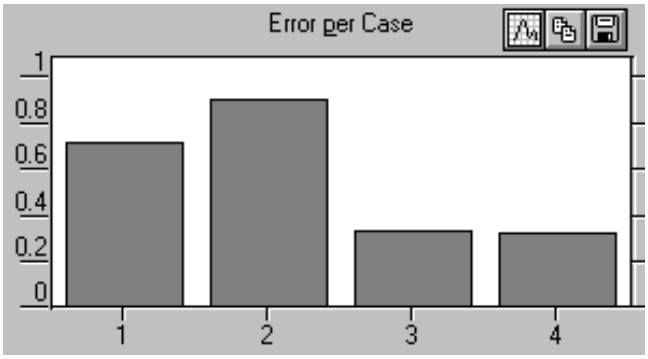
Rysunek 7: Funkcja odpowiedzi sieci liniowej dla problemu "XOR"



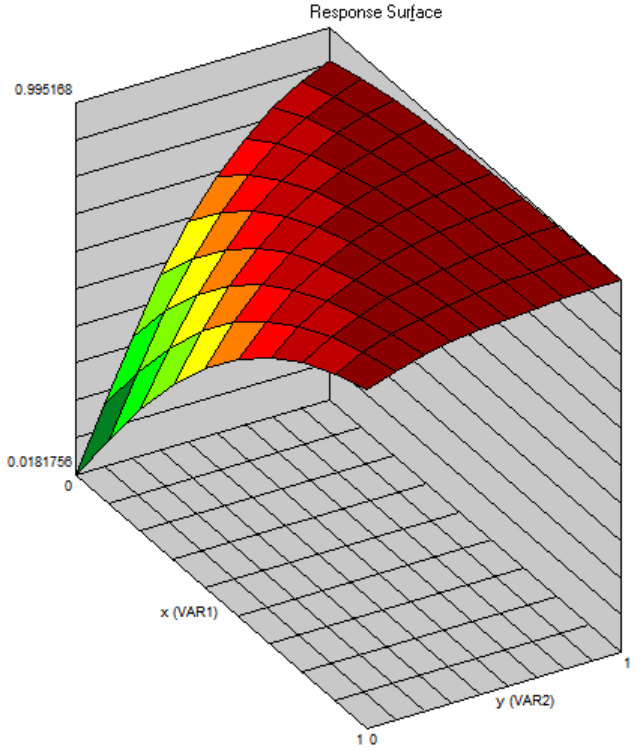
Rysunek 8: Błąd średniokwadratowy sieci liniowej dla problemu "XOR".

neurony w warstwie ukrytej (spróbuj je narysować w przestrzeni wejść) ? Jak można interpretować działanie neuronu wyjściowego ?
Proste definiowane przez neurony w warstwie ukrytej pokazano na rys. 14. Ponieważ wagi neuronu wyjściowego są zbliżone do wartości progu tego neuronu, to jego działanie można interpretować jako sumę logiczną: jeśli na jednym z wejść wartość jest bliska 1 lub większa, to na wyjściu neuronu wyjściowego również wartość będzie zbliżona do 1. Dzięki temu akumulowane są wyniki działania neuronów z warstwy ukrytej i zarówno dla wartości wejściowych (0,1) jak i (1,0) sieć daje prawidłową odpowiedź: 1.

12. **Obejrzyj, jak zmienia się rozkład wag podczas procesu uczenia. Jaka jest przyczyna takiego zachowania wag i jakie to może mieć konsekwencje (z "informatycznego" punktu widzenia) ?**
Na początku wagi zbliżają się do bardzo małych wartości bliskich zeru, co może doprowadzić do tego, że z powodu skończonej precyzji zapisu liczb rzeczywistych zostaną one wyzerowane.



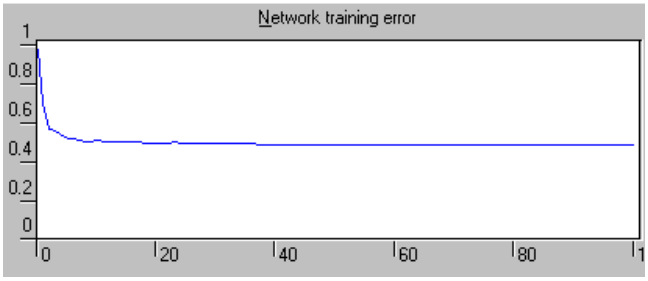
Rysunek 9: Błąd sieci liniowej dla problemu "XOR" dla poszczególnych przypadków .



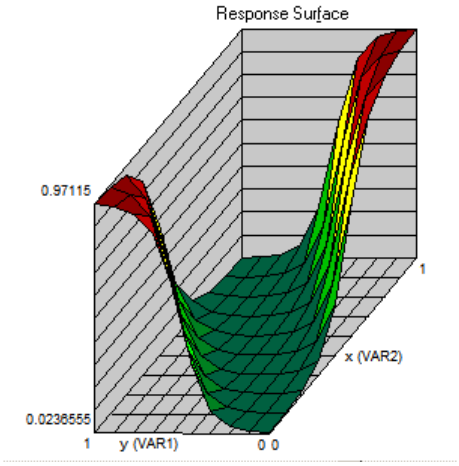
Rysunek 10: Funkcja odpowiedzi sieci nieliniowej jednowarstwowej dla problemu "XOR".

1.2 Obserwacja zjawiska przeuczenia na przykładzie zbioru PIMA.

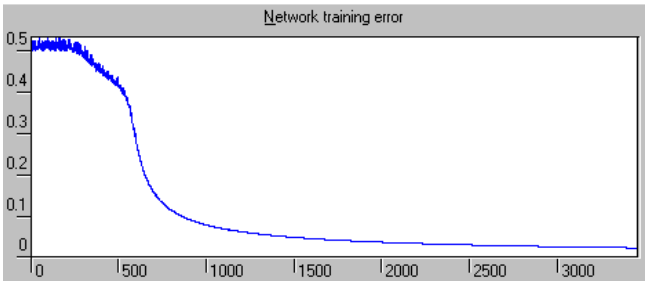
1. **Zapoznaj się z pochodzeniem zbiorów PIMA i BUPA (wyszukaj: "PIMA dataset", "BUPA dataset").**
   Zbiór PIMA zawiera dane dotyczące cukrzycy wśród indian Pima. Jest to zbiór o charakterze diagnostycznym: zawiera ogólne dane medyczne (8 atrbutów) i jeden bnarny atrybut decyzyjny (chory/zdrowy). Zbiór zawiera ukryte brakujące dane (nierzeczywiste wartości pewnych parametrów, np. ciśnienie krwi równe 0)
   Zbiór BUPA zawiera wyniki testów krwi które mogą wskazywać choroby wątroby, atrybut określający ilość sporzywanego dziennie alkoholu oraz jeden atrybut decyzyjny.



Rysunek 11: Błąd średniokwadratowy sieci nieliniowej jednowarstwowej dla problemu "XOR".



Rysunek 12: Funkcja odpowiedzi sieci nieliniowej 2-2-1 dla problemu "XOR".

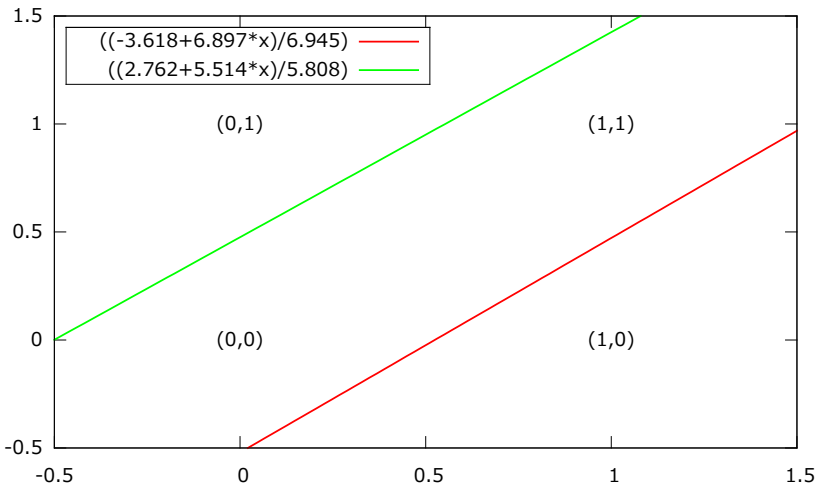


Rysunek 13: Błąd średniokwadratowy sieci nieliniowej 2-2-1 dla problemu "XOR"..

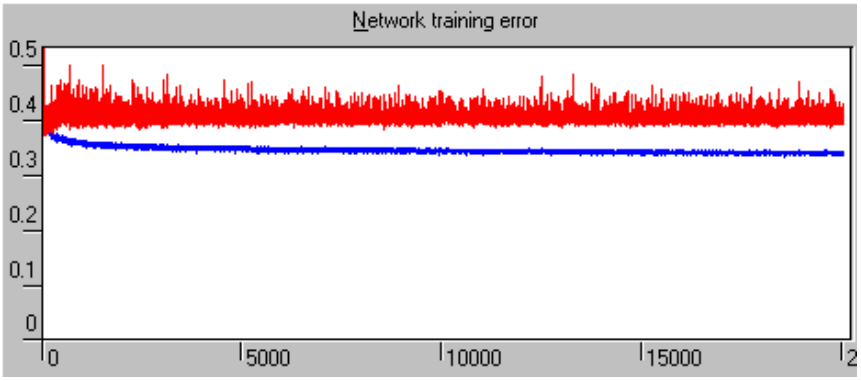
Zbiór nie posiada brakujących wartości.

2. **Wczytaj zbiór PIMA i skonstruuj dla niego dwuwarstwową sieć nieliniową o architekturze 8-4-1.**

3. **Przeprowadź uczenie algorytmem wstecznej propagacji błędu (może być bardzo długie, np. 20 000 epok); obserwuj przebieg błędu dla zbioru uczącego i weryfikującego.**
   Błąd pokazano na rys. 15. Jak widać po krótkim czasie następuje zjawisko przeuczenia: błąd dla zbioru weryfikującego jest większy od błędu dla zbioru uczącego.



Rysunek 14: Proste separująca klasy decyzyjne dla problemu "XOR". Neurony warstwy ukrytej.



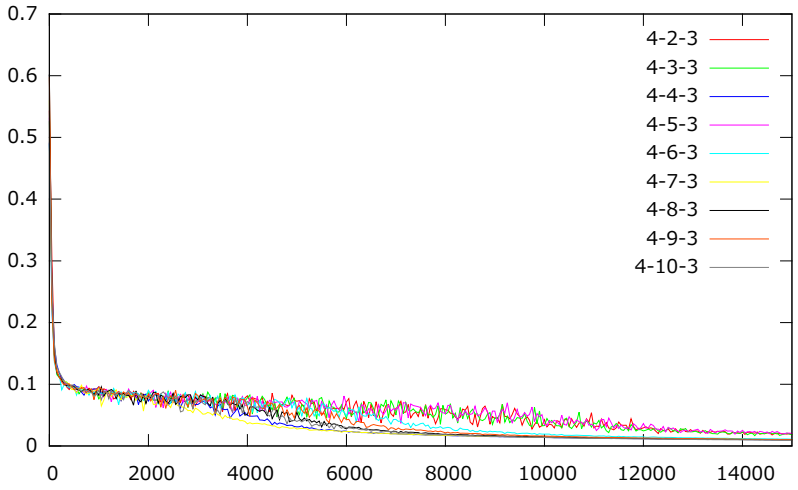
Rysunek 15: Błąd średniokwadratowy dla zbioru uczącego (kolor niebieski) i weryfikującego (kolor czerwony) dla przykładu PIMA

1.3 Dobór liczby neuronów w warstwie ukrytej na przykładzie zbioru IRIS.

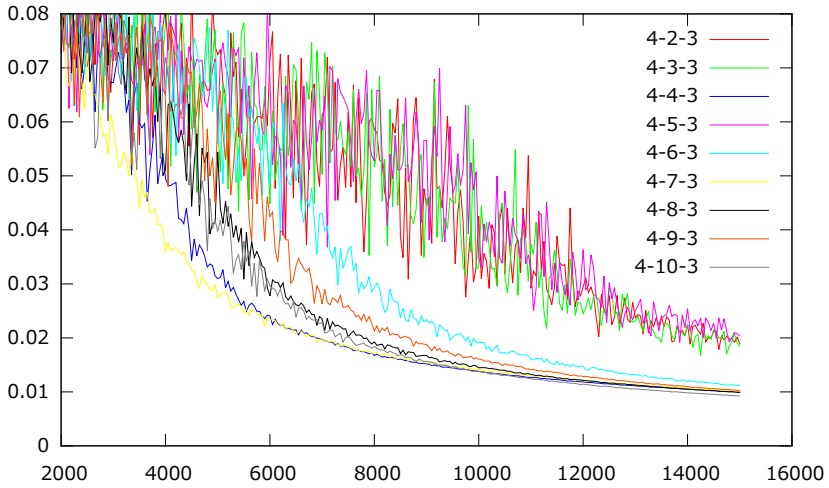
1. **Dla zbioru IRIS przeprowadź co najmniej 5 eksperymentów uczenia nieliniowych sieci dwuwarstwowych o architekturach 4-n-3, dla n zmieniającego się w przedziale [2,10], przy wyłączonym kryterium warunku stopu na zbiorze weryfikacyjnym. Po każdym eksperymencie kopiuj przebieg błędu przez schowek do Notatnika.**

2. **Przedstaw w gnuplocie na jednym wykresie przebieg błędu uczenia dla kolejnych wartości n.**
   Przebieg błędu uczenia przedstawiono na rys. 16.

3. **Czy istnieje jednoznaczna zależność pomiędzy n a przebiegiem błędu średniokwadratowego? Czy biorąc pod uwagę tylko przebeg błędu dla zbioru**



Rysunek 16: Błąd średniokwadratowy popełniany przez sieć dla różnych szerokości warstwy ukrytej.



Rysunek 17: Błąd średniokwadratowy popełniany przez sieć dla różnych szerokości warstwy ukrytej - przybliżenie.

uczącego można ustalić optymalną liczbę neuronów w warstwie ukrytej? Jeśli tak, to ile ona wynosi dla tego zbioru przykładów?
Nie, nie isrtnieje taka jednoznaczna zależność. Wielkości błędu dla różnych ilości neuronów w warstwie ukrytej są bardzo zbliżone. Jedynie dla n równego 2, 3, lub 5 błąd jest nieco większy, jednak ze względu na brak monotoniczności funkcji błędu względem ilości neuronów, wydaje się, że jest to przypadkowa różnica. Na podstawie samego przebiegu błędu nie można ustalić optymalnej ilości neuronów w warstwie ukrytej.

1.4 Sterowanie rozmiarem obszaru niepewnych odpowiedzi (braku odpowiedzi) sieci na granicach klas decyzyjnych (podczas testowania sieci)

1. **Utwórz i naucz sieć na zbiorze PIMA (architektura 8-4-1).**

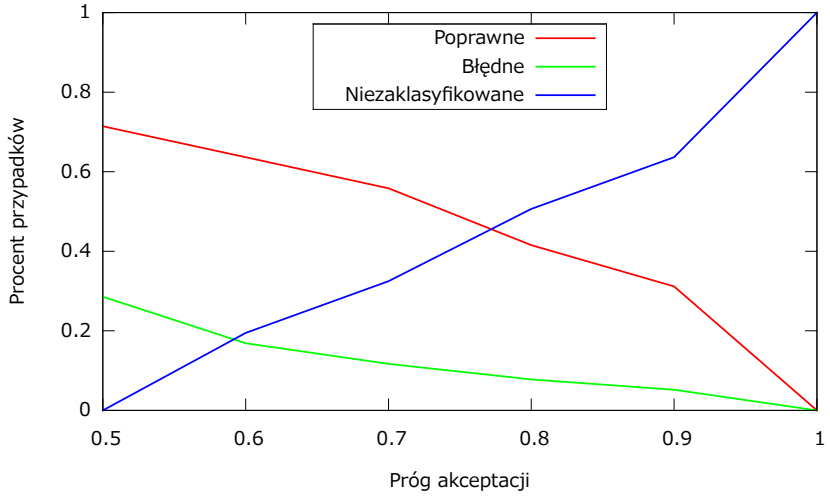
2. **Przy użyciu gnuplota sporządź wykres (trzy przebiegi na jednym wykresie) zależności procentu przypadków z jednego ze zbiorów (uczącego / weryfikującego / testującego):**

   - **poprawnie zaklasyfikowanych (do wszystkich klas decyzyjnych razem),**
   - **niepoprawnie zaklasyfikowanych,**
   - **niezaklasyfikowanych w funkcji progu Accept, dla następujących wartości Accept i Reject:**

Tabela 3: Wartości parametrów Accept i Reject w zadaniu 2

Accept	Reject
0.5	0.5
0.6	0.4
0.7	0.3
0.8	0.2
0.9	0.1
1.0	0.0

Wykres pokazano na rysunku 18.



Rysunek 18: Zależność procentowej ilości przypadków zaklasyfikowanych poprawnie, błędni i niezaklasyfikowanych od progu akceptacji, dla zbioru testującego.

3. **Czy na podstawie otrzymanego wykresu można zasugerować jakąś optymalną wartość obu progów dla tego zbioru przykładów i tej sieci?**

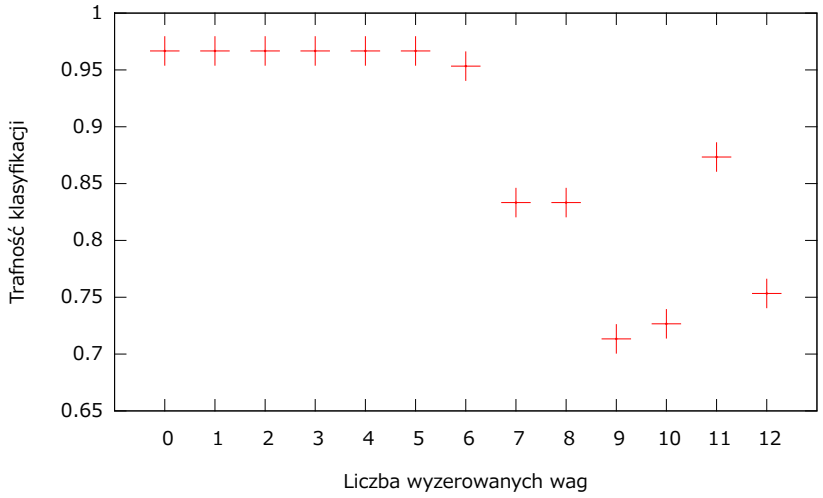
Tak, na podstawie wykresu (rys. 18) można zasugerować optymalną wartość, ale w ogólności zależy ona od specyfiki zadania klasyfikacji. Decydent musi zdecydować jakie jest maksymalne akceptowalne przez niego prawdopodobieństwo błędnej klasyfikacji i tak wyznaczyć próg akceptacji, żeby procent przypadków klasyfikowanych błędnie, odczytany z wykresu 18 był nie większy od tego maksymalnego akceptowalnego prawdopodobieństwa pojawienia się takich błędów. Jeszcze lepiej byłoby rozważyć ilość błędów false-positive i false-negative i dobrać próg akceptacji zgodnie z kosztami tych pomyłek. W przypadku zbioru PIMA koszt pomyłki wydaje się niewielki. Dla progu accept równego 0.5 otrzymujemy aż 80% trafnych klasyfikacji i tyklo 20% błędnych. Podwyższanie progu powoduje delikatny spadek liczby błędnych klasyfikacji przy gwałtownym spadku liczby klasyfikacji prawidłowych i gwałtownym wzroście liczby przypadków niesklasyfikowanych. Dlatego wydaje się, że optymalną wartością będzie tutaj 0.5.

1.5 Badanie odporności sieci na uszkodzenia

1. **Skonstruuj sieć dwuwarstwową o architekturze 4-3-3 dla problemu IRIS i naucz ją.**

2. **Otwórz edytor sieci (Edit—Network). Wyszukaj (w obu warstwach: ukrytej i wyjściowej!) i usuń z sieci (tj. wyzeruj) wagę najmniejszą co do wartości bezwzględnej (pomiń wiersz Threshold, zawierający progi neuronów). Przeprowadź ponownie testowanie sieci i zapisz wynik.**

3. **Kroki z punku 3 powtórz ok. 10 razy, kolejno usuwając coraz większe wagi i notując trafność klasyfikowania. Sporządź wykres zależności trafności klasyfikowania w funkcji ilości usuniętych (najmniejszych) wag.**
   Wykres pokazano na rysunku 19.



Rysunek 19: Wpływ uszkodzeń sieci na trafność klasyfikacji.

4. **Czy odporność sieci na uszkodzenia (usunięcia wag) jest wysoka?**
   Tak. Widać to na rysunku 19. Bez żadnej szkody dla trafności klasyfikacji można wyzerować pierwsze 5 najmniejszych wag - ich wartości były bliskie zeru. Po zerowaniu kolejnych wag trafność spada, by nagle, po usunięciu wagi 11 znów wzrosnąć.

5. **Czy jesteś w stanie na podstawie tak "zredukowanej" sieci powiedzieć coś o ważności poszczególnych atrybutów opisujących przykłady?**

Tabela 4: Wagi neuronów po wyzerowaniu 11 z nich.

	h1# 01	h1#02	h1#03	Set	Ver	Vir
Threshold	-1.931	8.862	1.698	-2.993	1.659	4.573
SLENGTH	0	0	0			
SWIDTH	0	-3.083	-3.994			
PLENGTH	0	7.648	5.271			
PWIDTH	0	10.05	4.444			
h1#01				0	0	0
h1#02				0	-8.377	8.924
h1#03				-7.039	7.538	0

Tak, w przypadku sieci (pokazanej w tabeli 4), w której wyzerowano 11 z 21 wag i która osiąga trafność klasyfikacji tylko ok 10% gorszą od pełnej sieci wyzerowane zostały wszystkie wagi związane z atrybutem *"slength"*. To sugeruje, że atrybut ten nie niesie wiele informacji z punktu widzenia przeprowadzanej klasyfikacji.

6. **Czy w konsekwencji "przerzedzenia" sieci można usunąć niektóre neurony? Jak należy to zrobić? (przemyśl dokładnie).**
Wagi pierwszego neuron z pierwszej warstwy ukrytej (h1#01) zostały całkowicie wyzerowane (zarówno wagi na jego wejściach jak i wszystkie wagi na wejściach następnej warstwy, do których jest on podłączony), co widać w tabeli 4. Dlatego wydaje się, że można by go usunąć bez wpływu na działanie sieci. Jednak, ponieważ w analizowanej sieci używana była nieliniowa (sigmoidalna) funkcja przenosząca (w terminologii SNN: activation function. W [2] mianem "funkcja aktywacji" określa się tę funkcję, która w SNN jest nazywana PSP - Post Synaptic Potential function), to wspomniany neuron można "bezkarnie" usunąć jedynie, jeśli wyzerowane są wszystkie wagi wejść, do których jest on połączony - wyzerowanie jedynie wejść tego neuronu nie wystarcza, ponieważ wartość funkcji aktywacji $f(0) = \frac{1}{1+e^{-0}} = \frac{1}{2} \neq 0$, a właśnie wartość 0 pojawiłaby się na odpowiednich wejściach neuronów z 3 warstwy gdyby usunąć całkowicie neuron h1#01.

1.6 Eksperymentalny dobór rozmiaru zbioru weryfikującego

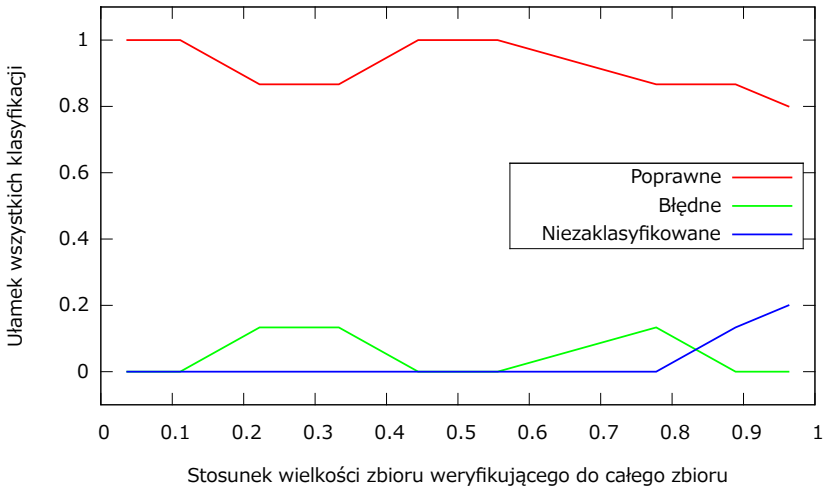
Dla zbioru IRIS, PIMA lub BUPA przeprowadź eksperyment uczenia i testowania, zmieniając proporcje pomiędzy zbiorem uczącym i weryfikującym.

1. **Zacznij od następujących proporcji zbioru uczącego, weryfikującego i testującego 8:1:1.**

2. **Przeprowadź uczenie (liczba epok $\geqslant 300$). Przetestuj sieć, zapisz trafność klasyfikowania, błąd klasyfikowania i procent przykładów niezaklasyfikowanych.**

3. **Sukcesywnie zwiększaj rozmiar zbioru weryfikującego, zmniejszając rozmiar uczącego (pole Training w oknie Edit—Data set), do proporcji 1:8:1 włącznie.**

4. **Sporządź wykres zależności trafności klasyfikowania, błędu klasyfikowania i procentu przykładów niezaklasyfikowanych w funkcji proporcji rozmiaru zbioru weryfikującego do liczby przykładów, jakie mamy do dyspozycji podczas uczenia (czyli rozmiar zbioru uczącego + rozmiar zbioru weryfikującego). Czy można znaleźć jakąś wielkość optymalną?**
Na podstawie wykresu z rysunku 20 można przypuszczać, że optymalny stosunek wielkości zbioru trenującego i weryfikującego to 1:1 - wtedy jest najmniej klasyfikacji nietrafnych i niepewnych a najwięcej poprawnych. Dla skrajnego przypadku stosunek zbioru trenującego do weryfikującego 8:1 - sieć jest tak samo dobra i wydaje się, że taka wielkość będzie bardziej odpowiednia dla małych zbiorów uczących, w których przydzielenie większej liczby przykładów do zbioru weryfikującego spowodowałoby zbytnie zredukowanie zbioru uczącego.

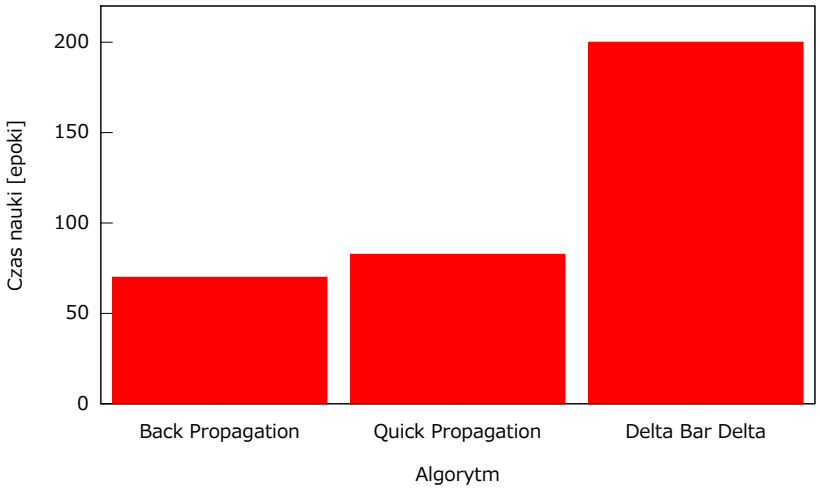


Rysunek 20: Częstość klasyfikacji poprawnych, błędnych i niezaklasyfikowanych w funkcji proporcji rozmiaru zbioru weryfikującego do rozmiaru zbioru użytego podczas uczenia (zbiór *IRIS*).

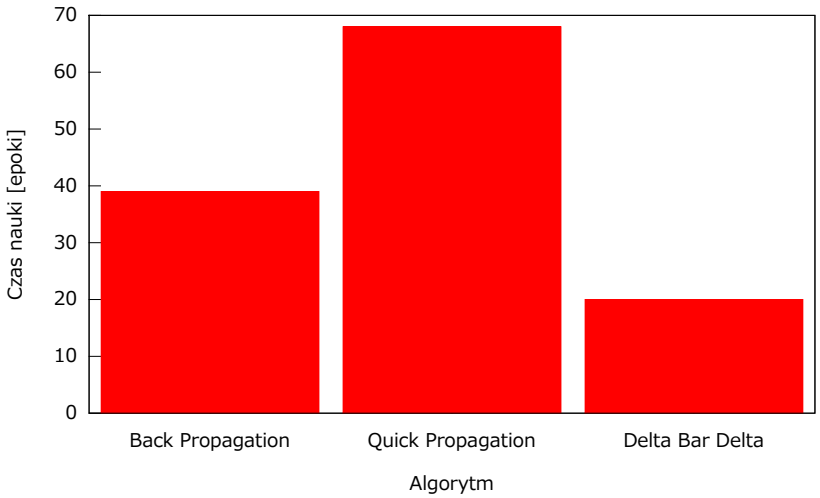
1.7 Porównanie 'vanilla' backpropagation z bardziej wyrafinowanymi algorytmami uczenia nadzorowanego sieci warstwowych

Dla zbioru PIMA lub BUPA przeprowadź eksperyment uczenia i testowania, używając poza standardowym (vanilla) algorytmem backpropagation innych algorytmów uczenia (np. QuickPropagation, Delta-bar-Delta). Dla zapewnienia porównywalności wyników, w poszczególnych eksperymentach zachowaj tą samą konfigurację sieci i warunki stopu. W miarę możliwości uśrednij wyniki z kilku przebiegów dla każdego algorytmu. Czy któryś z algorytmów jest wyraźnie lepszy? Porównaj liczbę epok uczenia.
Uwaga. Algorytmy QuickProp i DeltaBarDelta mogą okazać się trochę niestabilne (w ogólności łatwiej niż backprop wpadają w minima lokalne). W takim przypadku trzeba się trochę pobawić parametrami.

Dlat estowanych zbiorów danych najlepszym algorytmem okazał się zwykły *Vanilla Back Propagation* (rysunki 21 i 22. Co prawda *Delta-Bar-Delta* kończył działanie szybciej, ale osiągał znacznie gorsze wyniki, co widać na rysunku 23.



Rysunek 21: Średni czas uczenia (w epokach) sieci aż do osiągnięcia kryterium stopu dla różnych algorytmów uczenia. (zbiór *IRIS*).

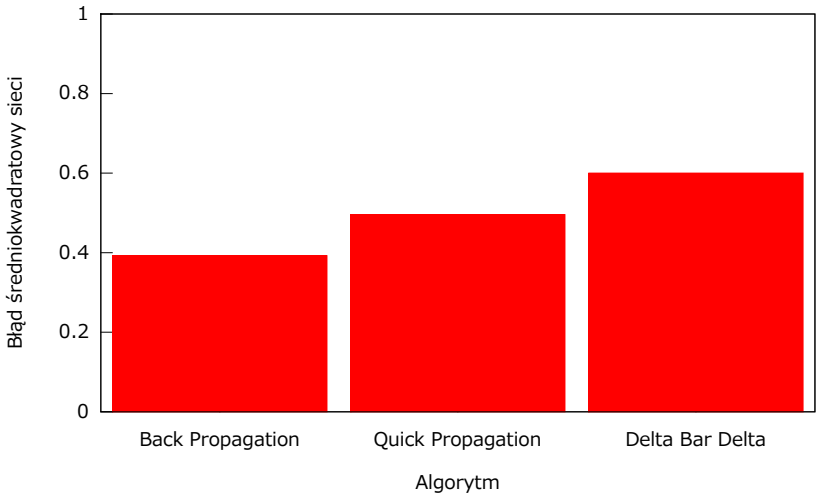


Rysunek 22: Średni czas uczenia (w epokach) sieci aż do osiągnięcia kryterium stopu dla różnych algorytmów uczenia. (zbiór *PIMA*).

1.8 Inne sieci warstwowe – sieci z jednostkami o symetrii kołowej (RBF)

1. **Zbuduj sieć RBF dla problemu IRIS. Otwórz edytor sieci (Edit—Network). Zwróć uwagę na różnice w polach Act fn i PSP in w porównaniu z perceptronami.** Funkcja *Act* to funkcja wykładnicza (w perceptronach: f. logistyczna) a funkcja *PSP* to funkcja *RBF*.

2. **Naucz sieć. Uczenie składa się z trzech etapów: ustalania centrów poszczególnych neuronów, ustalenia promieni (deviation), które decydują o stopniu**



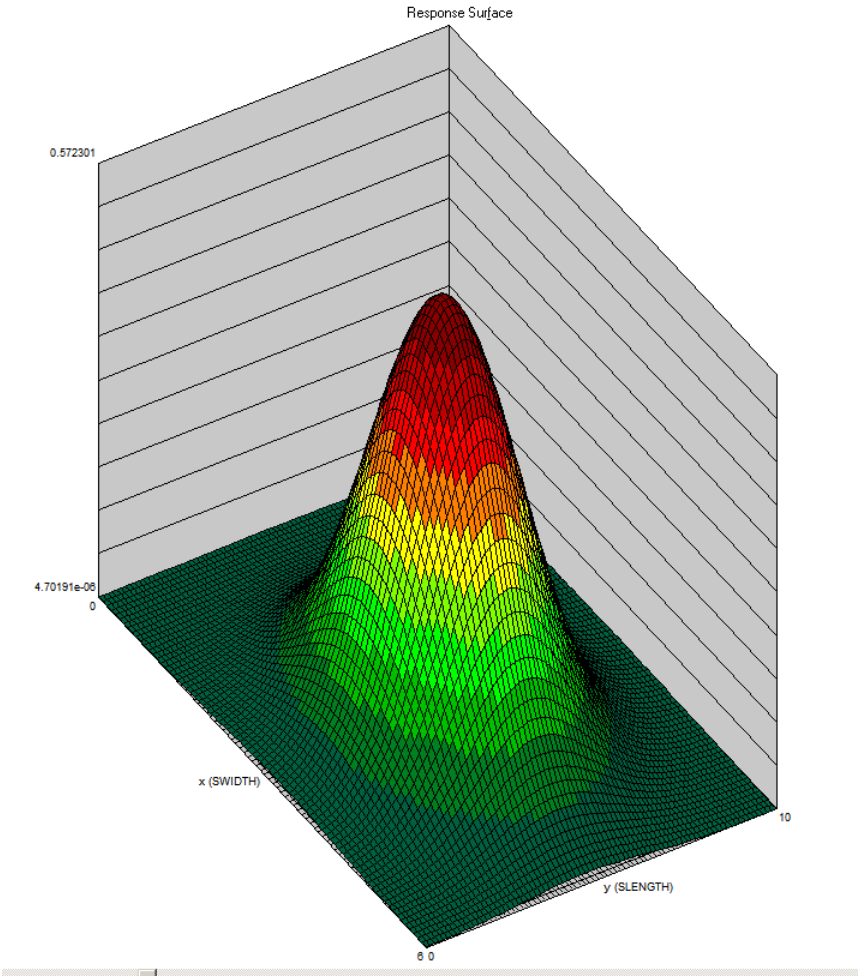
Rysunek 23: Średni błąd średniokwadratowy po osiągnięciu warunku stopu dla różnych algorytmów uczenia. (zbiór *PIMA*).

"spłaszczenia" funkcji Gaussowskich, oraz obliczenia wag neuronu wyjściowego. Zwróć uwagę na czas uczenia.
Podczas uczenia użyto grupowania k-means w celu ustalenia centrów neuronów, heurystyki k-nearest z k=14 w celu wyznaczenia promieni. Wagi neuronów wyjściowych zostały obliczone za pomocą pseudoinwersji.

3. **Obejrzyj funkcję odpowiedzi wybranych neuronów w warstwie ukrytej i neuronów w warstwie wyjściowej (Run—Response Surface). Możesz zmieniać też zmienne niezależne (atrybuty). Zauważ różnice w stosunku do sieci z jednostkami logistycznymi.**
Funckję odpowiedzi dla jednego z neuronów warstwy ukrytej pokazano na rys. 24 a z warstwy wyjściowej na rys. 25.

4. **Zastanów się, jakie konsekwencje ma niewłaściwy dobór promieni. Co będzie się działo, gdy promienie będą za duże? Co, gdy za małe? (związek z generalizacją).**
Im szerszy promień tym więcej przykładów obejmuje swoim zasięgiem każdy z neuronów w warstwie ukrytej i z tym mniejszą rozdzielczością odróżnia od siebie różne przykłady. Jeśli promień będzie duży, to zdolność sieci do generalizacji (klasyfikacji nieznanych przykładów) będzie większa, ale jednocześnie sieć będzie mniej szczegółowa - mniejsza będzie różnica w odpowiedzi dla dwóch różnych przykładów. W skrajnym przypadku każde z centrów będzie odpowiadać na każdy z przykładów i ich odpowiedzi będą bardzo podobne - wtedy warstwa ukryta sieci praktycznie nie działa i taką sieć można by zastąpić 2 warstwową sięcią liniową.
Jeśli promienie będą małe, to funkcje gausowskie będą bardziej szpiczaste i sieć może utracić zdolność do generalizacji - nowe przykłady mogą nie pasować do żadnego z centrów i w rezultacie nie będą rozróżnialne - nie zostaną przydzielone do żadnej z klas, albo zostaną przydzielone do przypadkowej klasy.

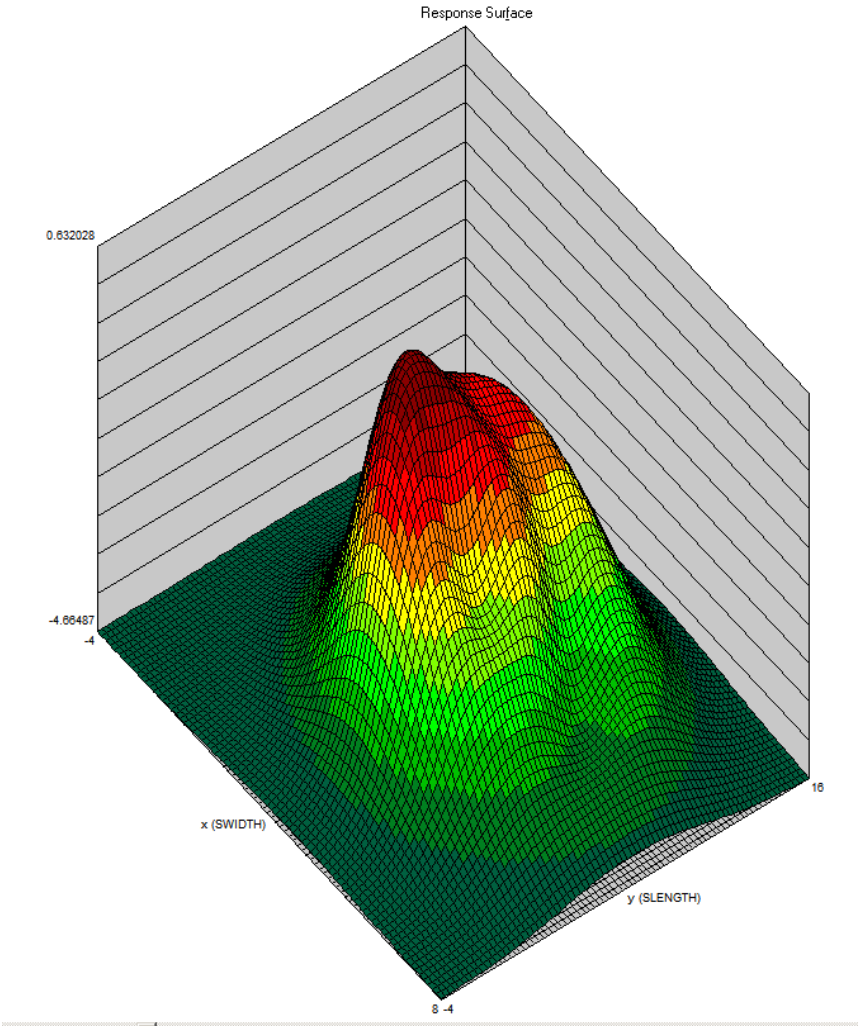
5. **Przetestuj sieć dla różnych metod uczenia, tj. dla różnych inicjalizacji centrów (Sample, K-means) i różnych sposobów ustalania odchyleń (Explicit,**



Rysunek 24: Funkcja odpowiedzi neuronu z warstwy ukrytej.

Isotropic, K-nearest). Najlepiej miej otwarty edytor sieci; obserwuj jakie konsekwencje ma wybranie poszczególnych sposobów uczenia.
Inicjalizacja centrów wpływa na wektory wag neuronów RBF. Inicjalizacja przez losowe próbkowanie (sample) sprawia, że algorytm staje się niedeterministyczny w tym sensie, że po każdym uczeniu sieci otrzymujemy inną sieć. Zjawisko to nie występuje w przypadku wybrania centrów używając grupowania k-means.
Sposób ustalania odchyleń wpływa na wagi w warstwie wyjściowej oraz na progi w warstwie ukrytej - im większy promień tym mniejszy próg - zgodnie z tym co napisano w dopowiedzi na poprzednie zadanie - tym więcej przykładów obejmie swoim zasięgiem (da dla nich niezerową odpowiedź) każdy z neuronów RBF.

6. **Porównaj sieć RBF z siecią typu Multilayer Perceptron na trudniejszym zbiorze przykładów (PIMA lub BUPA).**
Porównanie sieci RBF i MLP na przykładzie zbioru PIMA, widoczne na rys. 26, pokazuje że klasyczna sieć neuronowa (MLP) radzi sobie nieco lepiej niż sieć RBF, jednak różnice te nie są znaczące. W przypdku zbioru BUPA (rys. 27) lepsza okazała się sieć RDF, przy czym różnica w trafności klasyfikacji była podobna jak w przypadku zbioru



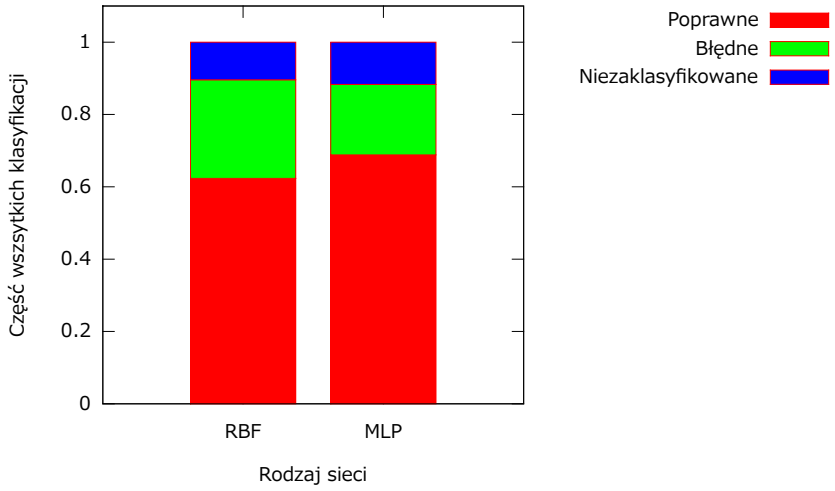
Rysunek 25: Funkcja odpowiedzi neuronu z warstwy ukrytej.

PIMA.

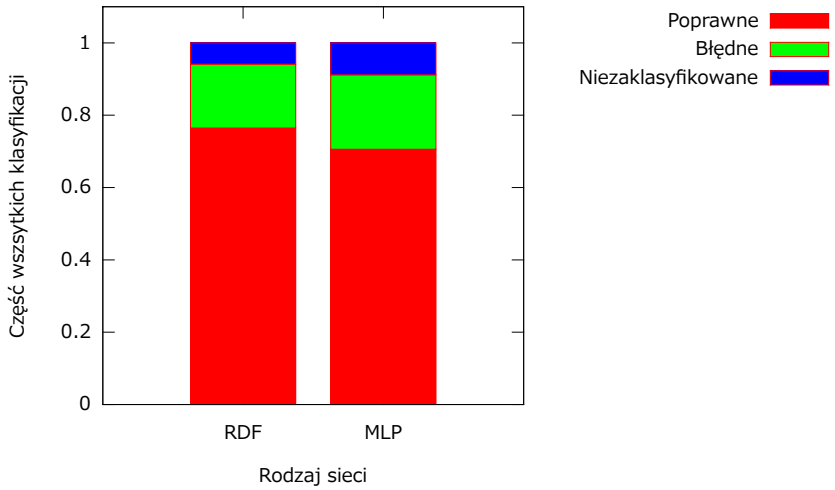
1.9 Uzyskiwanie jak najwyższej trafności

Manipulując:

- **architekturą sieci,**
- **prędkością uczenia (można zmieniać dynamicznie),**
- **zakresem wartości używanych do inicjalizacji wag sieci,**
- **warunkami stopu (statycznymi i/lub dynamicznymi),**
- **rodzajem algorytmów,**

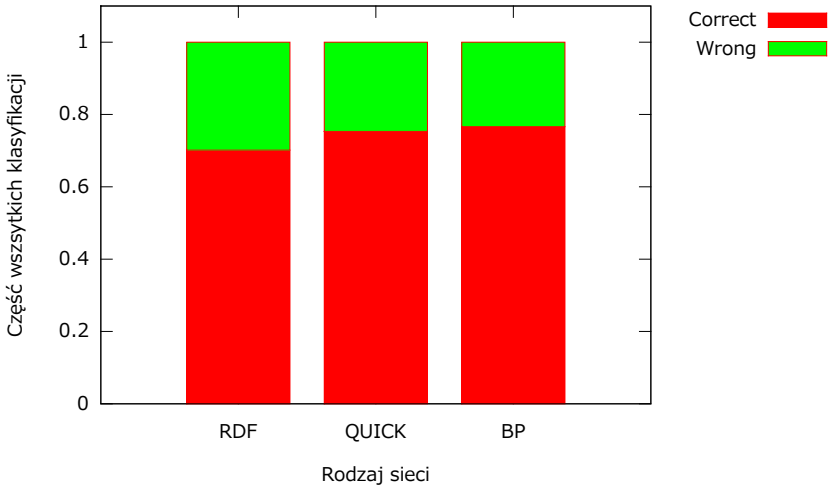


Rysunek 26: Trafność klasyfikacji zbioru PIMA przez sieć Multilayer perceptron (MLP) i sieć RDF.



Rysunek 27: Trafność klasyfikacji zbioru BUPA przez sieci Multilayer perceptron (MLP) i sieć RDF.

spróbuj uzyskać jak największą trafność klasyfikowania na zbiorze testującym dla zbioru PIMA lub BUPA.
Największą trafność klasyfikacji zbioru PIMA - 0.76 - udało się otrzymać stosując zwykły algorytm Back Propagation o parametrach: prędkość uczenia: 0.1, moment: 0.33. Nie jest to wynik średni, lecz najlepszy uzyskany w kilku próbach. Jest on porównywalny z wynikami klasyfikacji PIMA raportowanymi w [1].



Rysunek 28: Trafność klasyfikacji zbioru PIMA przez sieć RDF oraz sieci MLP uczone algorytmami Back Propagation (BP) i Quick Propagation (QP).

2 Rekurencyjne sieci neuronowe

2.1 Kodowanie sieci neuronowych o dowolnej topologii

- **Napisz w dowolnym języku/narzędziu sparametryzowany generator warstwowych-sieci-feed-forward albo każdy-z-każdym dla f0 lub f1.**

```
from random import random
import sys
'''
Simple NN generator
Given number of neurons in each layer it generates feed-forward network in f1 no
First layer is input layer: number of neurons in the first layer is equal to num
'''
def generate(neuron_type, layers):
    print """org:
name: Anonymous framstick
genotype:~"""
    genotype = "X"
    inputs_no = 0
    for layer, layer_size in enumerate(layers):
        if(layer > 0):
            inputs_no = layers[layer - 1]
        first_input = 1
        for neuron in range(layer_size):
            genotype += "[%s" % (neuron_type, )
            for input_neuron in range(first_input, first_input + inputs_no):
                weight = random()
                genotype += ",%d:%f" % (-input_neuron, weight)
            genotype += "]"
            first_input += 1
```

```
        print genotype
        print "~\n"


if __name__ == '__main__':
    if sys.argv[1] == '-h' or sys.argv[1] == '--help':
        print """
        Simple NN generator
        Given number of neurons in each layer it generates
        feed-forward network in f1 notation, with random weights.

        usage:
            nn_generator.py N a b c
        where:
            N - number of genotypes to generate
            a - number of neurons in first layer
            b - number of neurons in second layer
            c - ...
         example:
            nn_generator.py 5 3 4 1
         will generate 5 genotypes, each with neural network of architecture 3-4-
    else:
        N = int(sys.argv[1])
        layers = [int(c) for c in sys.argv[2:]]
        neuron_type = "Nu"
        for i in range(0,N):
            generate(neuron_type, layers)
```

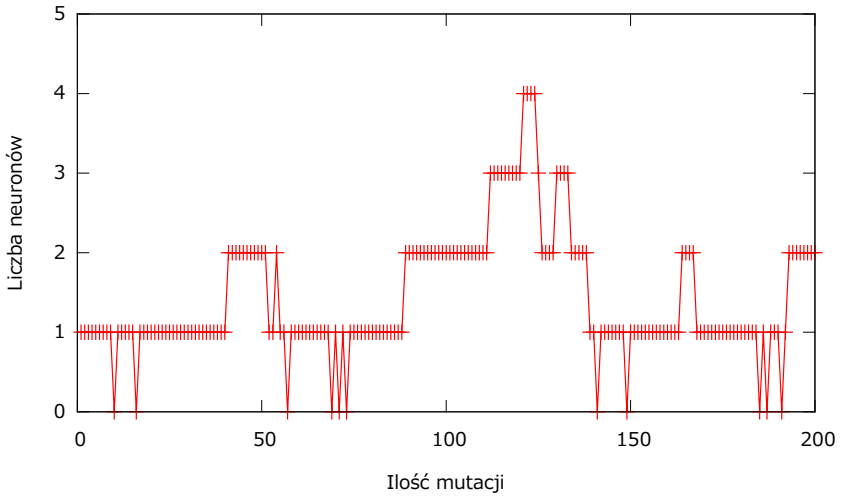
2.2 Optymalizacja wag i topologii

1. **Mutacja**

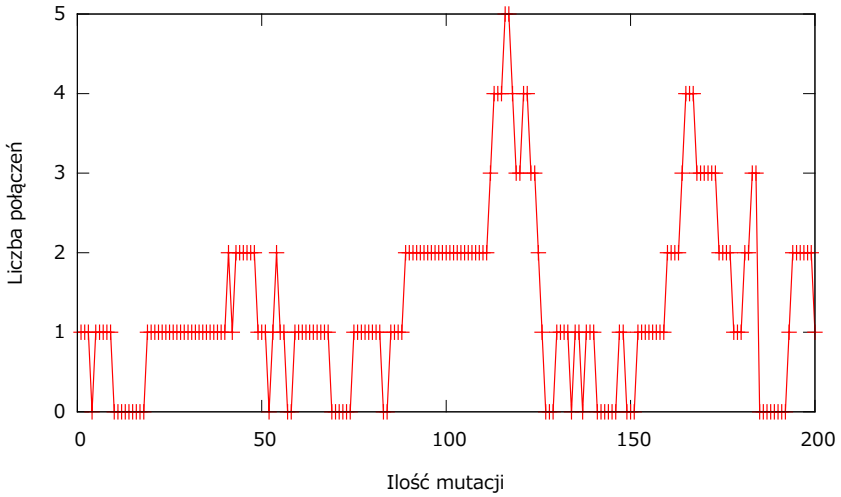
   - **Stwórz kilkanaście razy sekwencję 200 mutantów zaczynając od sieci z jednym neuronem. Co można powiedzieć o tych sekwencjach? Pomocniczo zrób wykresy liczby neuronów i połączeń neuronowych w funkcji n.**
     Na rysunkach 29 oraz 30 pokazano odpowiednio ilość neuronów i ilość połączeń między nimi w funkcji ilość mutacji. Widać na nich, że w trakcie mutacji neurony i połączenia między nimi są zarówno dodawane jak i odejmowane, sama mutacja bez presji selekcyjnej nie prowadzi w żadnym kierunku.

2. **Krzyżowanie**

   - **Wybierz jedną z reprezentacji: f0 lub f1.**
     Wybrano reprezentację f1.
   - **Stwórz dwie sieci neuronowe o tej samej topologii, różniące się tylko wagami.**
     Utworozno sieci Feed-Forward o topologii 3-4-3 pokazane na rysunku 31.
   - **Dokonaj ich krzyżowania. Czy rezultat spełnia postulaty dobrego krzyżowania podane w poprzednim ćwiczeniu?**
     Sieci widoczne na rysunkach 33, 34 i 35 są niespójne: w rezultacie krzyżowania



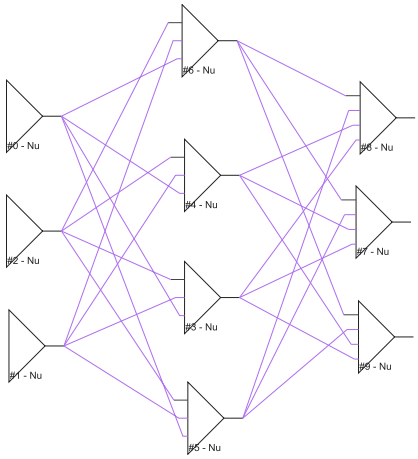
Rysunek 29: Ilości neuronów w funkcji ilości mutacji.



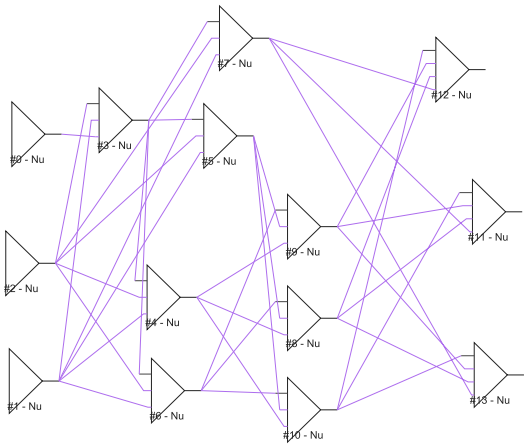
Rysunek 30: Ilości połączeń między neuronami w funkcji ilości mutacji.

powstały dwie niepołączone ze sobą sieci. Nie zawsze też jest zachowana liczba wejść i wyjść sieci (jak w przypadku sieci pokazanej na rys. 35). Obie wymienione wady przeprowadzonego krzyżowania sprawiają, że bez nałożenia ograniczeń na dopuszczalne powstałe genotypy nie powinno ono być stosowane. Z drugiej strony, sieci o niewłaściwej liczbie wejś/wyjść w wyniku selekcji byłyby odrzucane z populacji, więc użycie takiego krzyżowania jest możliwe.

- **Powtórz tę operację. Czy krzyżowanie jest deterministyczne?**
  Nie, krzyżowanie nie jest deterministyczne. Sieci widoczne na rysunkach 32 - 35 powstały w wyniku krzyżowania tych samych dwóch sieci o architekturach 3-4-3 (rys.31), ale różnią się od siebie.
- **Powtórz to zadanie (krzyżowanie) dla pary sieci neuronowych o bardzo**



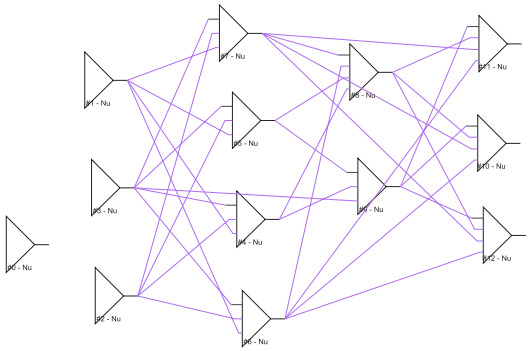
Rysunek 31: Sieć o architekturze 3-4-3, "rodzic" sieci prezentowanych na następnych rysunkach.



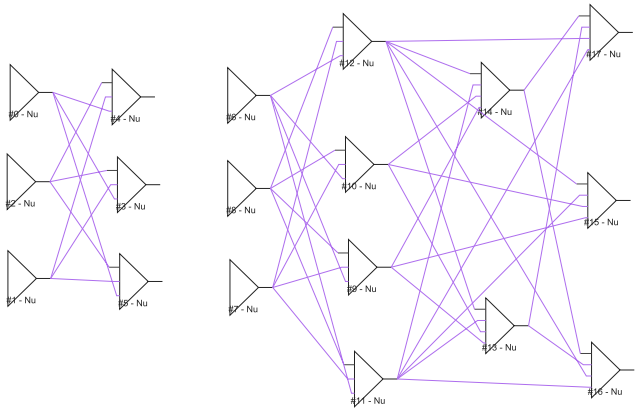
Rysunek 32: Potomek powstały w wyniku krzyżowania dwóch sieci o architekturze 3-4-3 widocznej na rys.31.

różnych topologiach

W wyniku krzyżowania dwóch bardzo różnych sieci: sieci feed-forward o topologii 6-12-8 oraz "podłużnej" sieci ze sprzężeniem zwrotnym pokazanej na rys. 36 powstała dość ciekawa sieć (pokazana na rys.37), która łączy w sobie cechy obu tych sieci: z jednej strony ma mało wejść i wiele sprzężeń zwrotnych (jak sieć ze sprzężeniem), z drugiej ma dużo wyjść - jak sieć 6-12-8.



Rysunek 33: Potomek powstały w wyniku krzyżowania dwóch sieci o architekturze 3-4-3 widocznej na rys.31.



Rysunek 34: Potomek powstały w wyniku krzyżowania dwóch sieci o architekturze 3-4-3 widocznej na rys.31.

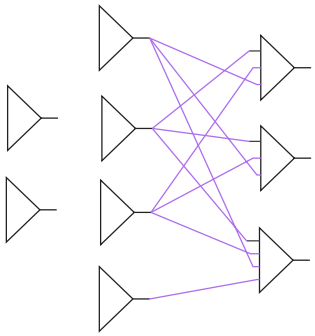
3 Uczenie nienadzorowane warstwowych sieci neuronowych

3.1 Projektowanie i testowanie prostej sieci typu SOM

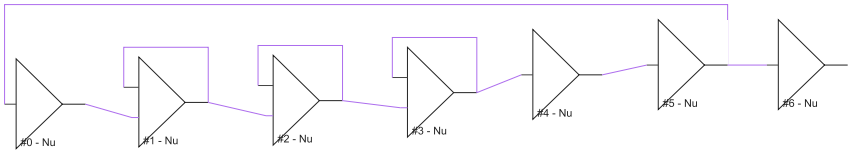
1. **Utwórz sieć typu mapa odwzorowania cech istotnych (SOM) o wymiarach 5x4 (sugerowanych przez StatsticaNN) dla zbioru IRIS: File—New—Network, potem wybierz Kohonen, Advise, Create.**

2. **Otwórz edytor sieci (Edit—Network). Jakiego typu neurony są stosowane w warstwie wyjściowej (okienko z ilustracją sieci to sugeruje) ?**
   Stosowane są neurony z radialną funkcją PSP (jak w sieci RBF) oraz pierwiastkową funkcją aktywacji (Act).

3. **Testowanie sieci. Obejrzyj najpierw, jak „odpowiadają" poszczególne neurony na kolejne przykłady ze zbioru uczącego (Run—Activations). Potem obejrzyj odpowiedzi sieci w diagramie, który zachowuje jej topologię (Run—Topological Map). Jak grupują się przykłady ? Zwróć uwagę, że podczas przeglądania przykładów mapą topologiczną, można samemu nazywać przykłady (górne**



Rysunek 35: Potomek powstały w wyniku krzyżowania dwóch sieci o architekturze 3-4-3 widocznej na rys.31.



Rysunek 36: Sieć ze sprzężeniem zwrotnym.

pole bez nazwy) i/lub neurony (dolne pole bez nazwy); działa także prawy przycisk myszki.
Na rysunku 38 przedstawiono mapę topologiczną dla problemu IRIS utworzoną za pomocą sieci SOM. Etykiety przypisano według klasy przykładów najczęściej mapowyanych na dany neuron. Widać, że przestrzenne rozłożenie neuronoów odpowiada podobieństwom między przykładami, do których są one podobne - przykłady należące do tej samej klasy są zgrupowane.

3.2 Analiza rzeczywistego problemu przy użyciu sieci SOM

Dane: Zbiór PROTEIN.STA, dostępny lokalnie w katalogu Statistica (Examples). Opisuje on spożycie protein różnego pochodzenia w wybranych krajach Europy. Dane nie są podzielone na klasy decyzyjne.

Zbuduj sieć dla tego problemu (sugerowany rozmiar warstwy wyjściowej 4x4 lub nawet 3x3, bo mało przykładów). Naucz ją i przetestuj. Czy da się wyciągnąć i uzasadnić jakieś wnioski dotyczące podobieństwa poszczególnych krajów pod względem spożycia protein? Można, podobnie jak dla IRIS, spróbować z siecią SOM jednowymiarową. Zbudowano sieć SOM o topologii 4x3. Jak widać na rysunku 39 sieć bardzo dobrze odwzorowuje podobieństwa między państwami (jeśli założyć, że państwa położone blisko siebie cechują się względnie podobnymi nawykami żywieniowymi). Na przykład wszystkie kraje przypisane do wierzchołka o etykiecie "Europa Zachodnia" są rzeczywiście zaliczane do krajów Europy Zachodniej (wg. klasyikacji ONZ). Podobnie było z krajami należącymi do Bałkanów, krajami Skandynawskimi jak i śródziemnomorskimi (Półwysep Iberyjski i Włochy).

Dziwne wydaje się to, że np. Polska, która jako jedyna zajmuje swój wierzchołek, nie jest do niego bardzo podobna - poziom aktywacji jej wierzchołka gdy przypadek "Polska" jest prezentowany na wejście sieci wynosi aż 0.52 i jest zbliżony do poziomu aktywacji wielu

innych węzłów (rys. 40. Można z tego wysnuć wniosek, że Polska kuchnia nie jest specyficzna, bierze po trochu z innych kuchni. Inaczej jest np. w przypadku kuchni Norweskiej(rys. 41): aktywacja wierzchołka "Skandynawia" jest niska - wynosi 0.38 (czyli przypadek Norwegia jest podobny do tego wierzchołka). Jednocześnie inne wierzchołki są zdecydowanie mniej podobne - najbardziej NRD - 0.67. Można to interpretować w ten sposób, że kuchnia Norweska jest typową kuchnią Skandynawską i jest specyficzna na tle wszystkich kuchni Europejskich.

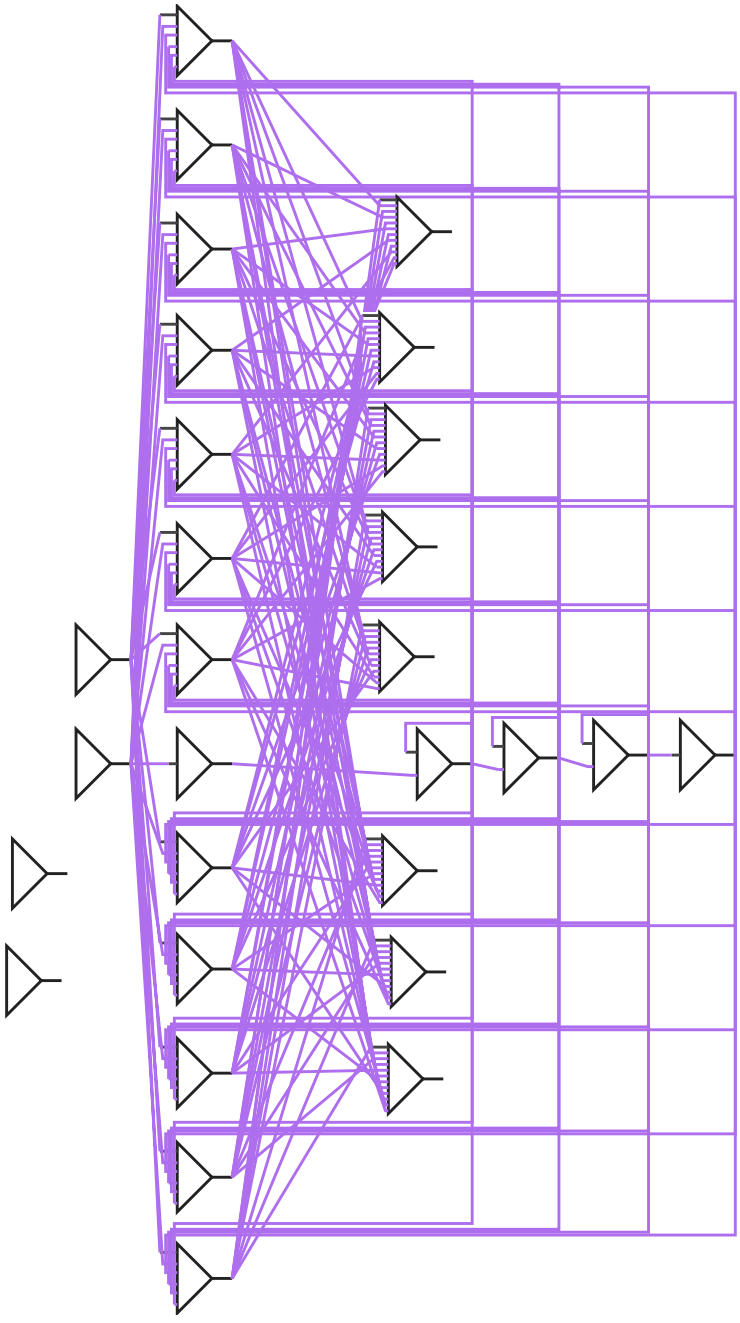
3.3 Projektowanie i testowanie prostej sieci typu SOM

Zobacz, co się dzieje, gdy opcja "Shuffle cases" jest wyłączona. Sprawdź wpływ parametrów algorytmu uczącego (blisko "skrajnych" wartości parametrów) na przebieg i efekty uczenia. Dla szybkości uczenia równej 1 wszystkie przykłady zostały przypisane do zaledwie 3 wierzchołków a błąd uczenia był bardzo duży (ok. 0.8). Im mniejsza prędkość uczenia tym mniejszy błąd, ale tym wolniej osiągane warunki stopu. Obserwując wykres błędu sieci i eksperymentując prędkością uczenia można dojść do wniosku, że w przypadku zbioru *PROTEIN* optymalna prędkość wynosi ok 0.1.
Funkcja "Shuffle cases" nie ma istotengo wpływu na błąd uczenia.

References

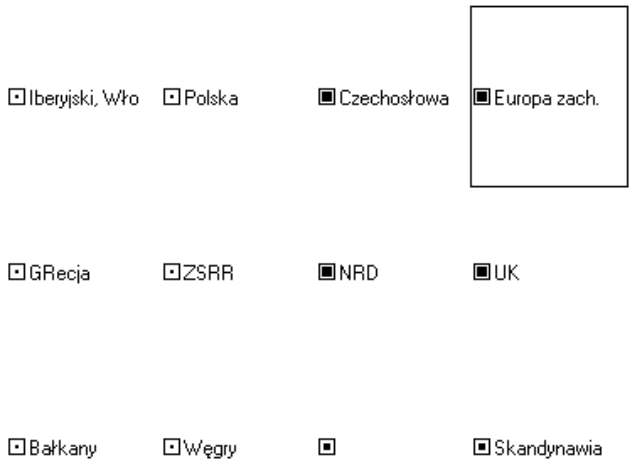
[1] Włodzisław Duch. *Datasets used for classification: comparison of results*. 2010. URL: `http://www.is.umk.pl/projects/datasets.html#Diabetes`.

[2] Jerzy Stefanowski Krzysztof Krawiec. *Uczenie maszynowe i sieci neuronowe*. 2nd ed. Wydawnictwo Politechniki Poznańskiej, 2004.



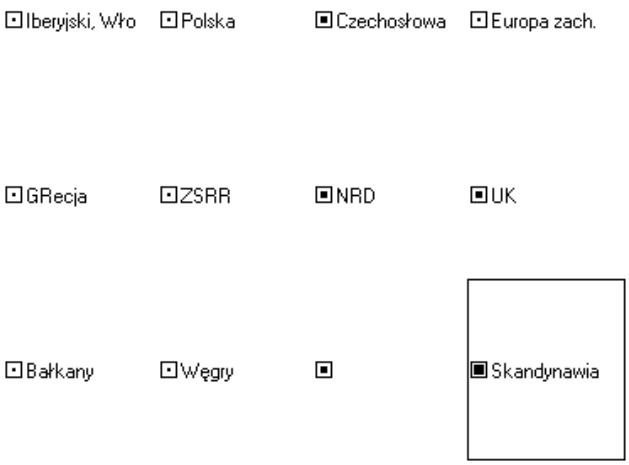
Rysunek 37: Potomek powstały w wyniku krzyżowania sieci feed-forward o architekturze 6-12-8 i sieci ze sprzężeniem zwrotnym pokazanej na rys.**??**.

Rysunek 38: Mapa topologiczna dla dproblemu IRIS



Rysunek 39: Mapa topologiczna dla problemu PROTEINS. Wierzhołki, do których przypisano więcej niż 2 kraje zostały opisane w poniższy sposób: Europa zachodnia: Austria, Belgia, Francja, Irlandia, Holandia, Szwajcaria, RFN; Bałkany: Albania, Bułgaria, Rumunia, Jugosławia; Skandynawia: Dania, Finlandia, Szwecja, Norwegia; Półwysep Iberyjski: Hiszpania, Portugalia.
.

Rysunek 40: Mapa topologiczna dla problemu PROTEINS. Pokazana aktywacja dla przykładu Polska.
.



Rysunek 41: Mapa topologiczna dla problemu PROTEINS. Pokazana aktywacja dla przykładu Norwegia.
.