

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

8 września 2011

Prowadzący: dr inż. Maciej Komosiński

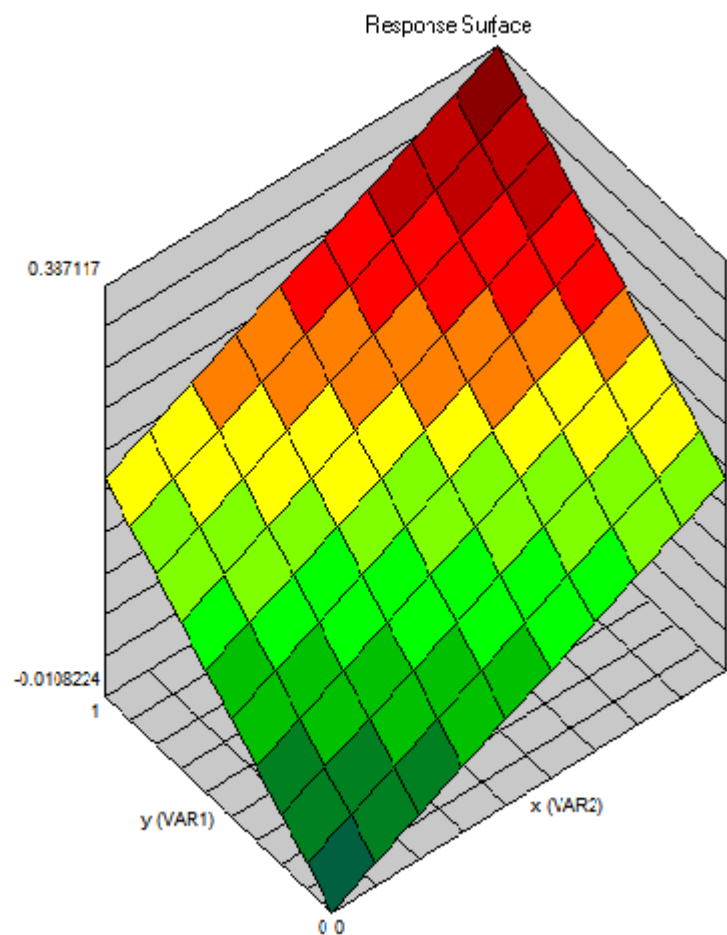
Autorzy: **Krzysztof Urban** inf84896 ISWD krz.urb@gmail.com
Tomasz Ziętkiewicz inf84914 ISWD tomek.zietkiewicz@gmail.com

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 liczebności?
2. Wyszukaj w Pomocy hasło "Verification" i przeczytaj wszystkie pięć znalezionych tematów.
3. Wyobraź sobie (narysuj) pożądaną funkcję odpowiedzi sieci (trójwymiarowy wykres zależności wyjścia od dwóch wejść).
4. 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—Response surface) i błędy dla poszczególnych przypadków (Statistics—Case errors).
5. Spróbuj utworzyć sieć liniową dla problemu AND o liczbie warstw większej niż 2.
6. Przerób sieć na nieliniową sieć jednowarstwową (ustawiając Act fn w Edit—Network na Logistic) i naucz ją na tym samym problemie.
7. 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.
8. Skonstruuj zbiór przykładów definiujący dwuargumentową funkcję XOR i zachowaj go. Wszystkie 4 przykłady mają stanowić zbiór uczący.
9. Wyobraź sobie (narysuj) pożądaną funkcję odpowiedzi sieci.
10. Skonstruuj liniową sieć jednowarstwową o architekturze 2-1 i naucz ją na problemie XOR.
11. Obejrzyj funkcję odpowiedzi sieci i błędy dla poszczególnych przypadków.
12. 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).

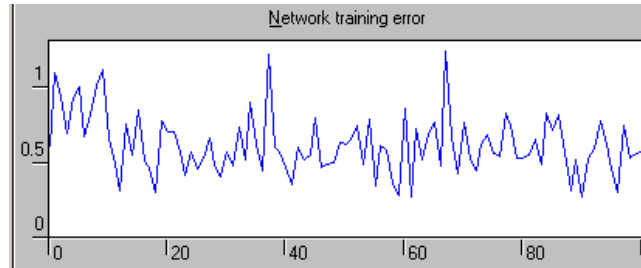


Rysunek 1: Funkcja odpowiedzi sieci.

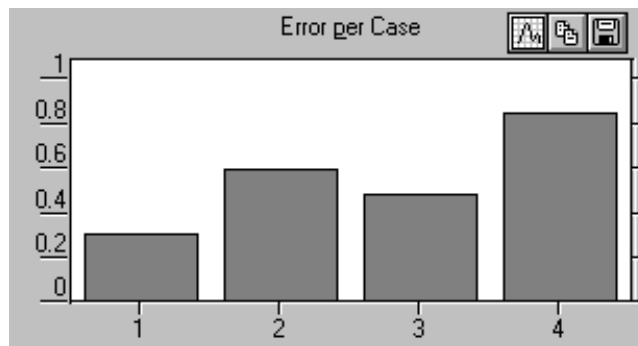
13. Skonstruuj nieliniową sieć dwuwarstwową o architekturze 2-2-1 (File—New—Network, Type=Multilayer perceptron) i naucz ją na problemie XOR. Obejrzyj funkcję odpowiedzi.
14. Obejrzyj wagi sieci w edytorze sieci (Edit—Network). Jak je definiują neurony w warstwie ukrytej (spróbuj je narysować w przestrzeni wejść) ? Jak można interpretować działanie neuronu wyjściowego ?
15. 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) ?

1.2 Obserwacja zjawiska przeuczenia na przykładzie zbioru PIMA.

1. Zapoznaj się z pochodzeniem zbiorów PIMA i BUPA (wyszukaj: "PIMA dataset", "BUPA dataset").



Rysunek 2: Błąd średniokwadratowy.

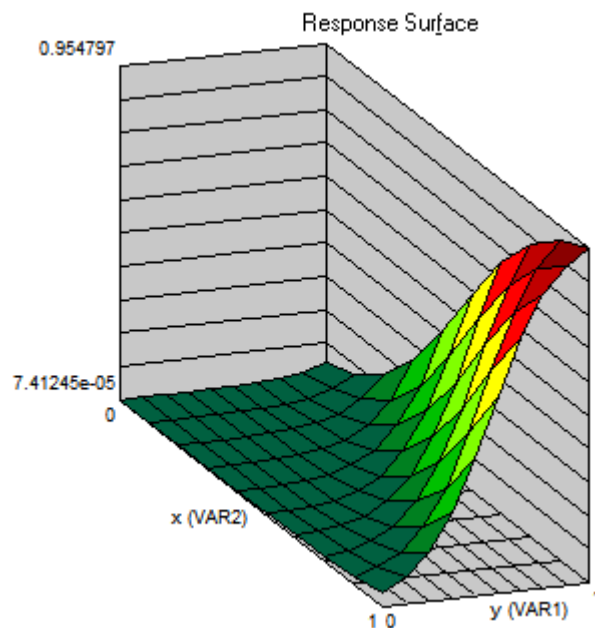


Rysunek 3: Błąd dla poszczególnych przypadków.

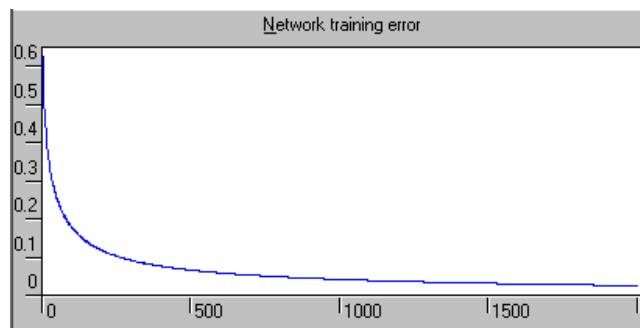
2. Wczytaj zbiór PIMA i skonstruuj dla niego dwuwarstwową sieć nieliniową o architekturze 8-4-1.
3. Obejrzyj ustawienia w okienku Pre/Post Processing.
4. Wyłącz "inkrementacyjne" warunki stopu ustawiając np. zero lub dużą wartość parametru Window w oknie Stopping Conditions.
5. Przeprowadź uczenie algorytmem wstecznej propagacji błędu (może być bardzo długie, np. 20 000 epok); obserwuj przebieg błędu dla zbioru uczącego i weryfikującego.

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.



Rysunek 4: Funkcja odpowiedzi sieci.

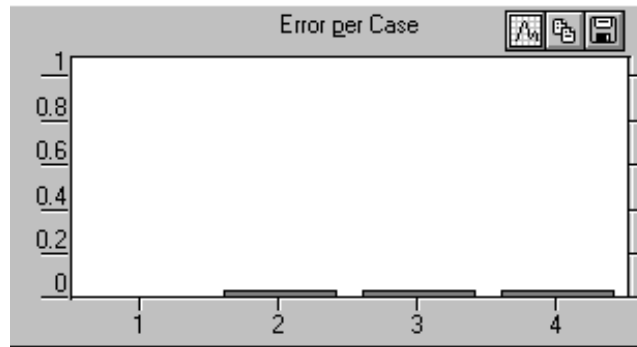


Rysunek 5: Błąd średniokwadratowy.

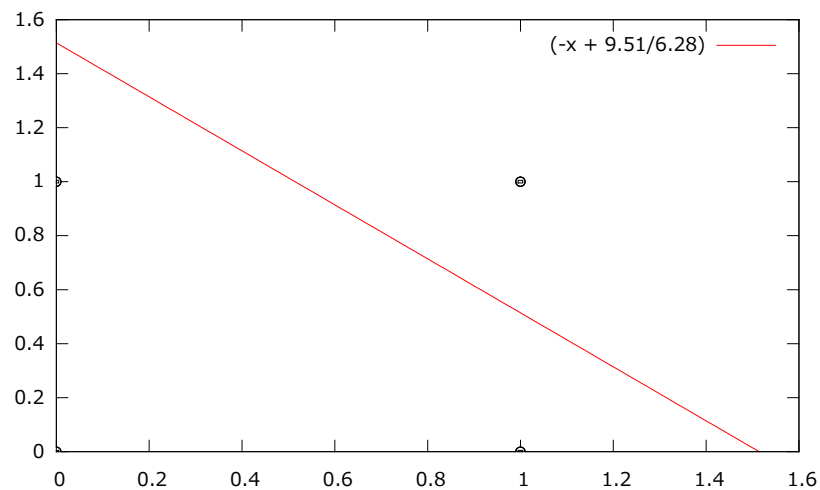
3. Czy istnieje jednoznaczna zależność pomiędzy n a przebiegiem błędu średniokwadratowego? Czy biorąc pod uwagę tylko przebieg błędu dla zbioru uczącego można ustalić optymalną liczbę neuronów w warstwie ukrytej? Jeśli tak, to ile ona wynosi dla tego zbioru przykładów?

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. Przeprowadź klasyfikowanie przykładów ze zbioru testującego (okno Stati-



Rysunek 6: Błąd dla poszczególnych przypadków.



Rysunek 7: Prosta separująca klasy decyzyjne.

stics—Classification). Znajdź w tym oknie macierz pomyłek (ang. confusion matrix).

- Otwórz okno Edit—Pre-Post Processing. Pola Accept i Reject definiują, jakie wartości muszą mieć neurony w warstwie wyjściowej, aby móc zinterpretować ich stan jako decyzję o przypisaniu przykładu do odpowiedniej klasy decyzyjnej. Choć nie jest to kontrolowane przez program, powinno zachodzić $\text{Accept} \neq \text{Reject}$
- 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:

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

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

1.5 Badanie odporności sieci na uszkodzenia

1. Skonstruuj sieć dwuwarstwową o architekturze 4-3-3 dla problemu IRIS i naucz ją.
2. Przeprowadź testowanie i zapisz trafność klasyfikowania.
3. 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.
4. Kroki z punktu 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.
5. Czy odporność sieci na uszkodzenia (usunięcia wag) jest wysoka?
6. Czy jesteś w stanie na podstawie tak “zredukowanej” sieci powiedzieć coś o ważności poszczególnych atrybutów opisujących przykłady?
7. Czy w konsekwencji “przerzedzenia” sieci można usunąć niektóre neurony? Jak należy to zrobić? (przemyśl dokładnie).

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. Zaczynij od następujących proporcji zbioru uczącego, weryfikującego i testującego 8:1:1.
2. Przeprowadź uczenie (liczba epok $\zeta=300$). 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?

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 tę 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.

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.
2. Naucz sieć. Uczenie składa się z trzech etapów: ustalania centrów poszczególnych neuronów, ustalenia promieni (deviation), które decydują o stopniu “spłaszczenia” funkcji Gaussowskich, oraz obliczenia wag neuronu wyjściowego. Zwróć uwagę na czas uczenia.
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.
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).
5. Przetestuj sieć dla różnych metod uczenia, tj. dla różnych inicjalizacji centrów (Sample, K-means) i różnych sposobów ustalania odchyleń (Explicit, Isotropic, K-nearest). Najlepiej miej otwarty edytor sieci; obserwuj jakie konsekwencje ma wybranie poszczególnych sposobów uczenia.
6. Porównaj sieć RBF z siecią typu Multilayer Perceptron na trudniejszym zbiorze przykładów (PIMA lub BUPA).

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,

spróbuj uzyskać jak największą trafność klasyfikowania na zbiorze testującym dla zbioru PIMA lub BUPA.

Jeśli możesz, porównaj uzyskany wynik z trafnością otrzymaną przy użyciu drzew decyzyjnych.

Aby zapewnić porównywalność wyników i powtarzalność eksperymentu, nie zmieniaj oryginalnego przyporządkowania przykładów do zbiorów uczącego, weryfikującego i testującego.

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.

2.2 Optymalizacja wag i topologii

1. Mutacja.

- Ustaw parametry mutacji (Experiment-Genetics) w f0 i f1 tak, żeby dotyczyły tylko dodawania/usuwania neuronów i dodawania/usuwania połączeń. Ustaw "Neurons to add" na jeden rodzaj neuronu – Nu.
- 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. W konsoli:

```
//make mutant sequence.
//ensure there is one genotype in the gene pool!
var n,ile=200;
GenePools.group=0; //select first group
for(n=1;n<=ile;n++)
{
    GenePools.genotype=n-1; //select n-th genotype as ancestor
    GenePools.mutateSelected();
    Genotype.name="mutant "+n; //set its name to consecutive number
    GenePools.copySelected(0);
    //Simulator.print(""+n+" "+Genotype.nnsiz);
}
```

2. Krzyżowanie

- Wybierz jedną z reprezentacji: f0 lub f1.
- Stwórz dwie sieci neuronowe o tej samej topologii, różniące się tylko wagami.
- Dokonaj ich krzyżowania. Czy rezultat spełnia postulaty dobrego krzyżowania podane w poprzednim ćwiczeniu?

W konsoli:

```
//cross over two neural networks.  
//ensure there are two genotypes in the gene pool!  
GenePools.group=0; //select first group  
GenePools.genotype=0; //select first genotype  
GenePools.crossoverSelected(1); //cross over with the second genotype  
Genotype.name="child"; //set descendant's name  
GenePools.copySelected(0);  
}
```

- Powtórz tę operację. Czy krzyżowanie jest deterministyczne?
- Powtórz to zadanie (krzyżowanie) dla pary sieci neuronowych o bardzo różnych topologiach.

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 StatisticaNN) 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) ?
3. Zwróć uwagę, jak zdefiniowany jest błąd popełniany przez sieć (Error function) (Suma po wszystkich przykładach odległości pomiędzy przykładem a najbliższym mu wektorem wag neuronów w warstwie wyjściowej =, uczenie nienadzorowane!)
4. Zobacz, jak można zmieniać topologię sieci manewrując parametrem Width dla drugiej warstwy.
5. Naucz sieć (Train—Kohonen) przy domyślnych ustawieniach parametrów. Zwróć uwagę, że mimo nienadzorowanego charakteru uczenia ma sens wyświetlanie przebiegu błędu.
6. Co więcej, można używać zbioru weryfikującego i oprzeć na nim warunek stopu. Zbadaj, jak uczy się sieć np. z warunkiem stopu Minimum improvement—Verification = 0.01, Window = 10 (wydaje się to sensowne, bo widać, że minimum lokalne znajdowane jest bardzo szybko i nie ma potrzeby uczyć sieci aż przez 100 epok).

7. 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 pole bez nazwy) i/lub neurony (dolne pole bez nazwy); działa także prawy przycisk myszki.
8. Poza tym można też obejrzeć statystykę „wygrywania” współzawodnictwa przez poszczególne neurony w poszczególnych podzbiorach (Statistics—Win frequencies).
9. W podobny sposób naucz i przeanalizuj dla tego samego zbioru sieć o innej topologii (szczególnie ciekawe jest to dla sieci o topologii „jednowymiarowej”, gdzie warstwa wyjściowa ma np. 20 neuronów ułożonych w rzędzie, parametr Width = 1 dla warstwy 2 w edytorze sieci).

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.

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.