Akademia Ekonomiczna w Krakowie

Ryszard Tadeusiewicz, Grażyna Paliwoda-Pękosz, Paweł Lula

Metody sztucznej inteligencji i ich zastosowania w ekonomii i zarządzaniu

Kraków, 2003

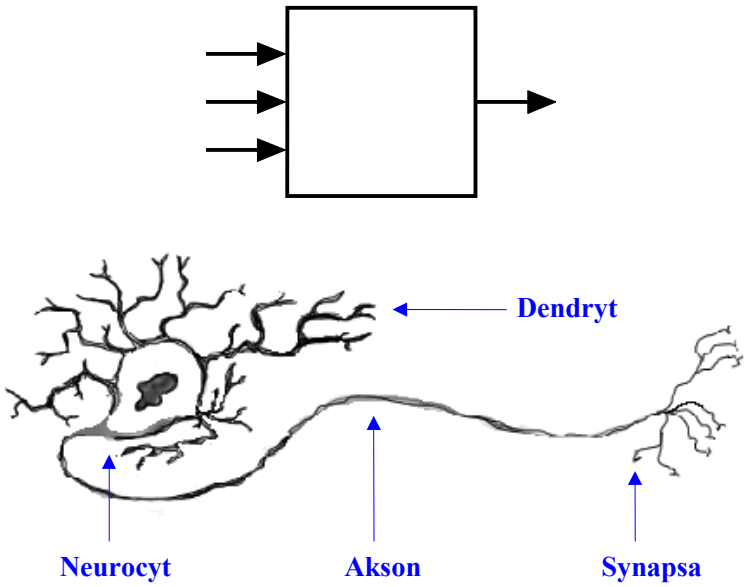
Spis treści

Rozdział 3. Sieci neuronowe

Sztuczne sieci neuronowe należą do metod analizy danych o bardzo dużych możliwościach aplikacyjnych. Inspiracją do ich skonstruowania była zdobyta na polu nauk biologicznych wiedza dotycząca budowy i sposobu funkcjonowania systemów nerwowych organizmów żywych, jednak sztuczne sieci neuronowe stanowią **bardzo** uproszczone naśladownictwo drobnych fragmentów rzeczywistego mózgu.

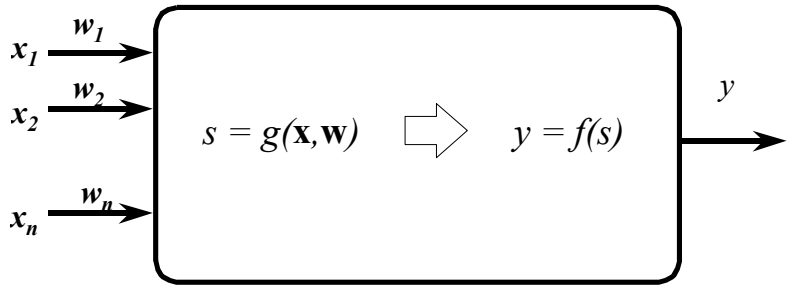
3.1. Sposób funkcjonowania sztucznego neuronu

Mówiąc o podobieństwie pomiędzy rzeczywistymi i sztucznymi sieciami neuronowymi **w zakresie ich budowy** należy zauważyć, że podobnie jak układ biologiczny nerwowy składa się z połączonych ze sobą komórek nerwowych, tak również sztuczna sieć neuronowa zbudowana jest z wzajemnie powiązanych ze sobą *elementów przetwarzających informacje* (zwanych, przez analogię do ich biologicznych odpowiedników, *sztucznymi neuronami* lub po prostu *neuronami*). Sztuczny neuron (rys. 16) może być rozpatrywany jako prosty system przetwarzający wartości sygnałów wprowadzanych na jego wejścia w pojedynczą wartość wyjściową, wysyłaną na jego jedynym wyjściu (dokładny sposób funkcjonowania określony jest przez przyjęty *model neuronu*).



Rys. 16. Schemat sztucznego neuronu (u góry) i jego biologiczny pirwowzór (u dołu).

Stosowane najczęściej w praktyce modele neuronów charakteryzują się następującymi cechami (rys. 17):



Rys. 17. Model neuronu

- sztuczne neurony są układami posiadającymi zależną od potrzeb liczbę n wejść i dokładnie jedno wyjście,
- informacje wprowadzane na wejścia mają zawsze postać numeryczną i tworzą wektor wartości wejściowych $\mathbf{x} = [x_1, x_2, ..., x_n]$,
- z każdym wejściem neuronu skojarzony jest współczynnik liczbowy zwany *współczynnikiem wagowym* lub krótko *wagą*; dla każdego wejścia wartość wagi (wyznaczana najczęściej automatycznie w procesie uczenia sieci) jest zwykle inna; współczynniki wagowe tworzą wektor wag neuronu $\mathbf{w} = [w_1, w_2, ..., w_n]$,
- sposób funkcjonowania neuronu składa się z dwóch faz, z których pierwszą można określić jako *wyznaczenie zagregowanej wartości wejściowej*, zaś drugą *wyznaczenie wartości wyjściowej neuronu*,
- w trakcie wyznaczania zagregowanej wartości wejściowej obliczana jest pomocnicza wartość s, określana na podstawie wszystkich wartości wejściowych **x** oraz wszystkich wartości składających się na wektor wag **w**. Sposób obliczania wartości s określony jest przez funkcję $g(\cdot)$ zwaną *funkcją agregującą*. Przeprowadzana w neuronie agregacja danych wejściowych opisana jest zwykle wzorem:

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

lub też formułą:

$$s = \sum_{i=1}^{n} (w_i - x_i)^2 \tag{2}$$

Warto zauważyć, że neuron funkcjonujący zgodnie ze wzorem (1) agregując dane wejściowe wyznacza sumę ważonych wejść, zaś neuron posiadający funkcję agregującą w postaci podanej wzorem (2) dokonuje obliczenia kwadratu odległości Euklidesa pomiędzy wektorem wartości wejściowych **x** a wektorem wag **w,**

- w trakcie wyznaczania *wartości wyjściowej* neuronu stosowana jest funkcja $f(\cdot)$, której argumentem jest wyznaczona wcześniej zagregowana wartość wejściowa s. Stosowana do wyznaczenia wartości wyjściowej neuronu funkcja $f(\cdot)$ nazywana jest *funkcją aktywacji* lub też *funkcją przejścia*. W najprostszym przypadku funkcja aktywacji przyjmuje postać funkcji identycznościowej:

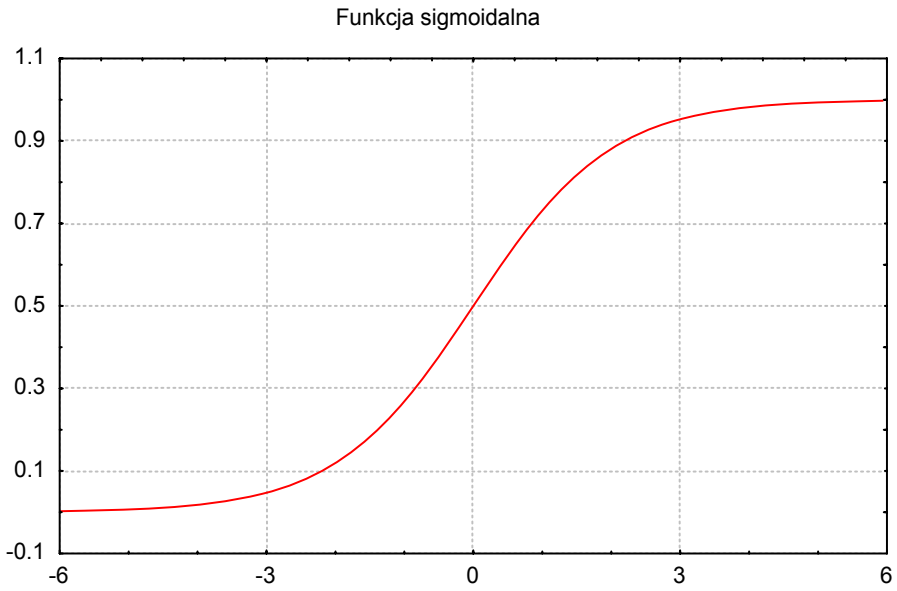
$$y = f(s) = s \tag{3}$$

Zastosowanie powyższej formuły powoduje, że na wyjściu neuronu pojawia się po prostu zagregowana wartość wejściowa. W połączeniu z funkcją agregacji daną wzorem (1) funkcja aktywacji (3) definiuje funkcjonowanie tzw. neuronu liniowego, pozwalającego na budowę bardzo użytecznych liniowych sieci neuronowych.

Bardzo popularną funkcją aktywacji jest *funkcja sigmoidalna* opisana wzorem:

$$f(s) = \frac{1}{1+e^{-as}}, \tag{4}$$

której wykres przypomina swoim wyglądem literę „s” (rys. 18). Występująca we wzorze wartość *a* jest stałym parametrem determinującym nachylenie wykresu funkcji (dla większych wartości *a* wykres funkcji przebiega bardziej stromo i przybliża się do wykresu funkcji progowej, zaś przy zmniejszaniu wartości tego parametru tempo zmian wartości funkcji jest coraz mniejsze i wykres funkcji coraz mocniej upodabnia się do wykresu funkcji liniowej). Jeśli zagregowana wartość wejściowa dąży do -∞, to wartość funkcji *f(·)* (a tym samym wartość wyjściowa neuronu) dąży do zera, zaś w przypadku wartość *s* dążących do +∞, na wyjściu neuronu pojawia się wartość dążąca do jedności.

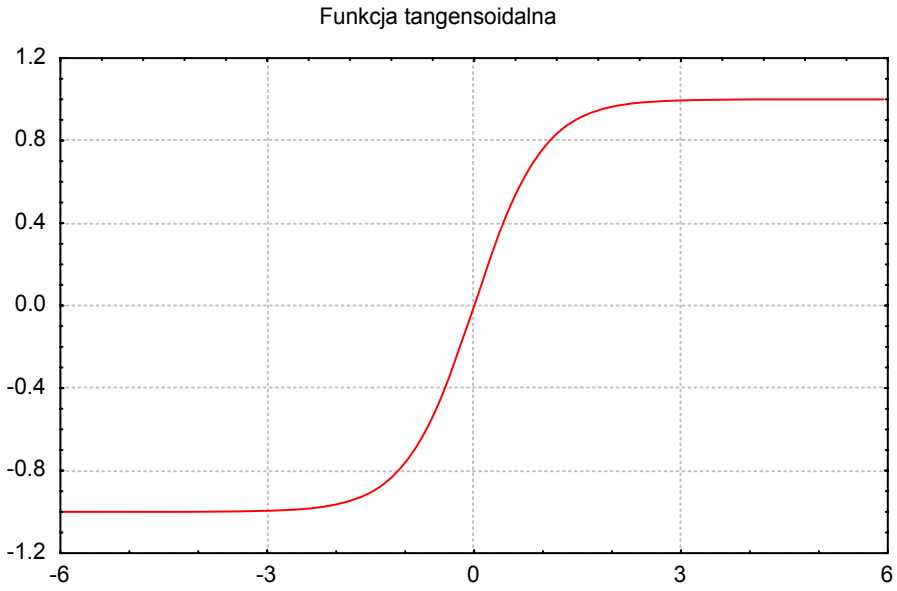


Rys. 18. Wykres funkcji sigmoidalnej

W charakterze funkcji aktywacji stosowana jest często również funkcja tangensoidalna:

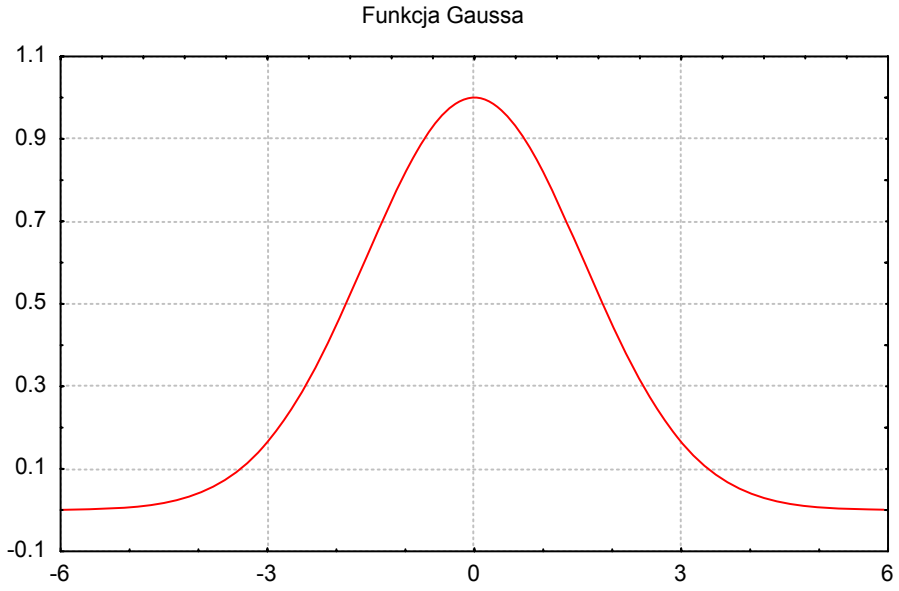
$$f(x) = tgh\left(\frac{ax}{2}\right) = \frac{1-e^{-ax}}{1+e^{-ax}} \tag{5}$$

która swoim wyglądem przypomina funkcję sigmoidalną, lecz jej wartości znajdują się w przedziale od minus do plus jedynki (rys. 19). Występujący we wzorze parametr *a* ma analogiczną interpretację jak w przypadku funkcji sigmoidalnej.



Rys. 19. Wykres funkcji tangensoidalnej

Dużą przydatnością charakteryzują się również tzw. radialne (gaussowskie) funkcje aktywacji (rys. 20).



Rys. 20. Wykres funkcji gaussowskiej

Mają one kształt dzwonowy, z maksimum przypadającym zwykle nad wartością zerową. Jedną z formuł opisujących takie funkcje dzwonowe jest wzór (6):

$$f(s) = e^{-as^2} \tag{6}$$

Występujący we wzorze współczynnik a odpowiada za szerokość wykresu funkcji - im większa jest jego wartość, tym wykres funkcji jest bardziej szpiczasty.

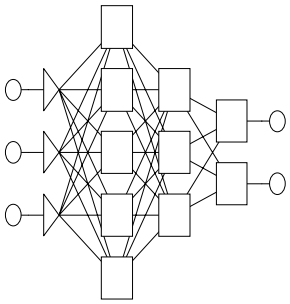
Zaprezentowany powyżej sposób funkcjonowania pojedynczego elementu przetwarzającego ukazuje jego prostotę. Warto podkreślić, że realizowany w neuronie sposób przetworzenia informacji wejściowych w informację wyjściową zawsze uzależniony jest od następujących elementów:

a) wartości wag neuronu,
b) zastosowanego sposobu agregacji danych wejściowych (czyli przyjętej funkcji agregującej),
c) sposobu generowania wartości wyjściowej (czyli od postaci funkcji aktywacji).

Elementy b) i c) są ustalane jednorazowo, w momencie tworzenia sieci i określania jej właściwości. Natomiast element a) jest przedmiotem procesu uczenia, co ma zasadnicze znaczenie w większości zastosowań sieci neuronowych – w tym także w obszarze *data mining*.

3.2. Budowa i zasada funkcjonowania sztucznych sieci neuronowych

Możliwości pojedynczego neuronu w zakresie przetwarzania informacji są stosunkowo niewielkie i z tego względu stosowane są (podobnie jak w przypadku układów nerwowych organizmów żywych) połączone ze sobą grupy sztucznych neuronów (czyli sieci neuronowe), pozwalające na przeprowadzenie znacznie bardziej złożonych obliczeń, koniecznych w przypadku korzystania z wielu bardziej ambitnych metod przetwarzania danych (rys. 21).



Rys. 21. Struktura przykładowej sztucznej sieci neuronowej

Neurony wchodzące w skład sztucznej sieci neuronowej ułożone są najczęściej w warstwach. Wartości wyjściowe wyznaczone dla neuronów jednej warstwy wprowadzane są na wejścia neuronów warstwy następnej. Wyjątkiem jest tu warstwa pierwsza (zwaną *warstwą wejściową*) składająca się z neuronów, do których wejść doprowadzane są wartości z zewnątrz sieci (dotyczące wartości zmiennych wejściowych, występujących w rozwiązywanym problemie jako *przesłanki* dla prowadzonego wnioskowania) oraz warstwa ostatnia (*wyjściowa*), składająca się z neuronów wyznaczających wartości wysyłane potem na zewnątrz jako wartości wyjściowe całej sieci (traktowane w rozwiązywanym problemie jako *wnioski* z przeprowadzonego rozumowania). Warstwy znajdujące się pomiędzy warstwą wejściową i wyjściową nazywane są *warstwami ukrytymi*.

Występujące w sieci połączenia komunikują zwykle wszystkie neurony jednej warstwy ze wszystkimi neuronami warstwy kolejnej. Taki typ połączeń znany jest pod nazwą *połączeń typu każdy-z-każdym*. W sieciach warstwowych zwykle nie występują połączenia pomiędzy neuronami znajdującymi się w nie sąsiadujących ze sobą bezpośrednio warstwach[1].

[1] Przedstawiony model sieci nazywany jest *jednokierunkowym*, gdyż informacja przepływa tylko w jednym kierunku - od warstwy wejściowej do warstwy wyjściowej. Wśród sieci warstwowych znane są również sieci *rekurencyjne*, w których występują połączenia przekazujące wartości wyjściowe neuronów jednej warstwy na wejścia warstw wcześniejszych.

Sieć neuronowa w trakcie swojego działania przetwarza wprowadzone na jej wejścia wartości zmiennych wejściowych w wyniku czego uzyskiwane są na jej wyjściach wartości zmiennych wyjściowych. Sposób pracy sieci uzależniony jest od wielu czynników, do których należy zaliczyć przede wszystkim:

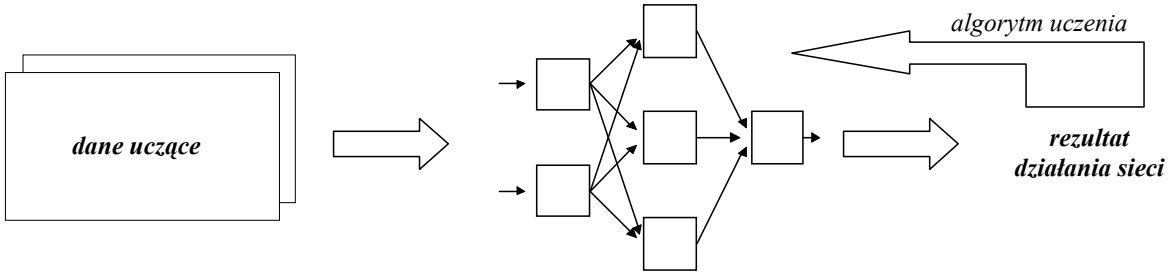
a) przyjęte modele neuronów - a więc wybrane dla neuronów sieci konkretne postacie funkcji agregującej i funkcji aktywacji; zwykle wszystkie neurony wchodzące w skład tej samej warstwy korzystają z takich samych formuł funkcji agregującej i funkcji aktywacji, natomiast w różnych warstwach stosowane są często neurony korzystające z różniących się formuł;
b) wartości współczynników wagowych neuronów - są one ustalane automatycznie w trakcie procesu uczenia i dlatego są różne dla poszczególnych wejść i dla poszczególnych neuronów;
c) liczba warstw sieci;
d) liczba neuronów w poszczególnych warstwach sieci;
e) przyjęty sposób połączeń neuronów.

Warunkiem koniecznym do właściwego funkcjonowania sieci jest poprawne określenie **wszystkich** wymienionych powyżej czynników.

3.3. Uczenie sieci neuronowych

Pewne podobieństwa pomiędzy rzeczywistymi i sztucznymi sieciami neuronowymi można dostrzec nie tylko w ich budowie, ale również w sposobie pozyskiwania wiedzy niezbędnej do ich **prawidłowego** funkcjonowania. Proces pozyskiwania wiedzy przez sieć neuronową musi gwarantować możliwość późniejszego jej uogólniania w celu rozwiązania całego postawionego przed nią zadania (a nie tylko odtworzenia wiedzy zgromadzonej w tzw. zbiorze uczącym).

Proces przygotowania sieci do prawidłowego działania nazywany jest **uczeniem**. Sieć *uczy się* prawidłowo działać na podstawie prezentowanych jej *przykładów* realizacji badanych obiektów lub zjawisk. Bazując na przedstawionych rzeczywistych przypadkach (dla których odpowiedzi na stawiane pytania są znane) sieć stara się odkryć i zapamiętać ogólne prawidłowości charakteryzujące te obiekty lub kierujące przebiegiem badanych zjawisk. Rozpoznane reguły sztuczna sieć neuronowa przechowuje w postaci zakodowanej w wartościach współczynników wagowych neuronów. Można więc stwierdzić, że proces uczenia sieci polega na prawidłowym określeniu wartości współczynników wagowych neuronów na podstawie informacji zdobytych w trakcie prezentacji danych odzwierciedlających prawidłową kategoryzację rozważanych obiektów albo rzeczywisty przebieg badanych zjawisk. Zbiór danych wykorzystywany w trakcie uczenia nazywany jest *zbiorem uczącym*. Ogólny schemat uczenia przedstawia rys. 22.



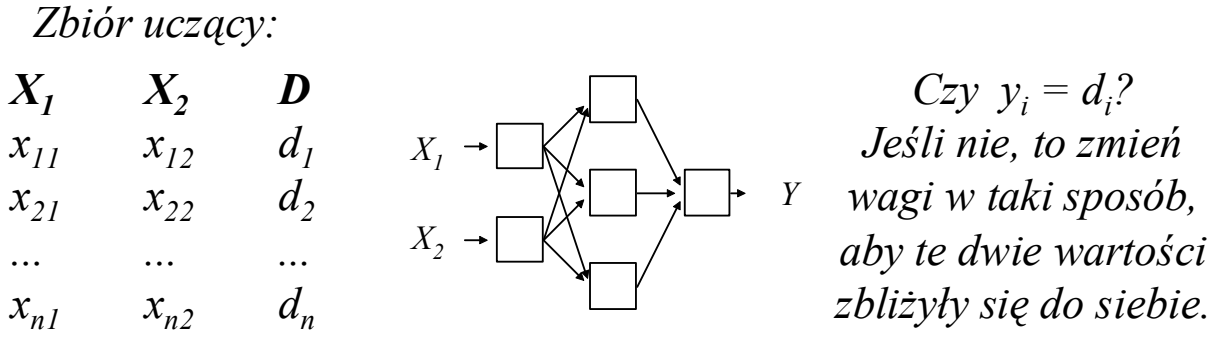
Rys. 22. Schemat uczenia sztucznej sieci neuronowej.

Uczenie sieci w większości przypadków rozpoczyna się od nadania wagom neuronów wartości losowych. Następnie, w czasie uczenia na wejściach sieci prezentowane są przechowywane w zbiorze uczącym wartości zmiennych wejściowych. Wprowadzone wartości wejściowe przepływają przez kolejne warstwy sieci i są przetwarzane przez wszystkie neurony sieci zgodnie z regułami, jakie są zawarte w ich strukturze, aby w końcu osiągnąć warstwę wyjściową jako sygnał będący odpowiedzią sieci na postawione jej zadanie. W trakcie **pierwszego** pokazu danych wejściowych wartości wyjściowe sieci są całkowicie przypadkowe - z uwagi na losowe wartości początkowe współczynników wagowych. Porównując te wartości wyjściowe, które sieć wypracowała, z tymi wartościami, które powinny być ustalone (a które znamy, bo w zbiorze uczącym są zawarte przykładowe zadania *wraz z rozwiązaniami*) można ustalić wartości błędów popełnianych przez sieć. Wyznaczone wartości błędów (bezpośrednio lub pośrednio przypisane do konkretnych neuronów) a także wartości wyjściowe określanie na wyjściach wszystkich neuronów sieci są podstawową informacją wykorzystywaną do uczenia (czyli do modyfikacji wartości współczynników wagowych). Zastosowany sposób zmiany wartości wag jest uzależniony od *typu* uczonej sieci i od zastosowanego *algorytmu uczenia*.

Proces uczenia sieci uwalnia nas od uciążliwego tworzenia i zapisywania (w określonym języku programowania) algorytmu wymaganego dla takiego przetwarzania wejściowych danych, by uzyskać pożądany wynik końcowy. Nie odbywa się to jednak za darmo, gdyż ceną, jaką trzeba zapłacić za wygodę nie myślenia o algorytmie rozwiązywania problemu, jest długotrwały i wymagający sporych mocy obliczeniowych proces uczenia. Zwykle jednokrotna prezentacja wszystkich przypadków wchodzących w skład zbioru uczącego (czyli tzw. jedna epoka uczenia) nie wystarcza do osiągnięcia prawidłowego działania sieci. Dlatego też dane uczące prezentowane są wielokrotnie – często kilkaset, kilka tysięcy, albo nawet milionów razy. Co gorsza, proces uczenia jest zawsze procesem indeterministycznym, co oznacza, że wynik uczenia nie jest nigdy całkowicie pewny. Czasami ta sama sieć w tym samym zadaniu może uzyskiwać znacząco różne (co do jakości) rozwiązania postawionego problemu. Fakt ten powoduje, że dla osiągnięcia najlepszego możliwego wyniku proces uczenia trzeba niekiedy powtarzać kilkakrotnie, za każdym razem startując od innych wartości początkowych przyjętych dla współczynników wag. Jest to bardzo kłopotliwe, ale czasami bywa jedynym sposobem pokonania wyłaniających się trudności. Na szczęście trudności te nie zawsze się pojawiają, co powoduje, że wśród użytkowników sieci neuronowe cieszą się (zasłużenie) opinią narzędzia sprawnego i wygodnego w użyciu.

Wyróżnia się dwa podstawowe sposoby realizacji procesu uczenia sieci - *uczenie z nauczycielem* oraz *uczenie bez nauczyciela*. Elementem różnicującym oba podejścia do procesu uczenia jest struktura zbioru uczącego. W przypadku **uczenia z nauczycielem** (rys. 23) w trakcie uczenia wykorzystywany jest zbiór uczący zawierający przykładowe zadania

wraz z ich poprawnymi (wzorcowymi) **rozwiązaniami.** Oznacza to, że w zbiorze uczącym zapisane są zarówno zaobserwowane wartości zmiennych wejściowych (które będą wprowadzane na wejścia sieci) oraz odpowiadające im wartości zmiennych wyjściowych (które mają charakter wzorców poprawnych odpowiedzi i które dzięki temu mogą być porównywane z tymi rozwiązaniami, które są generowane przez neurony wyjściowe sieci). Dzięki posiadaniu w zadaniu uczenia z nauczycielem pełnej informacji o celach uczenia sieci, proces uczenia może być precyzyjnie kontrolowany, a przez to może być maksymalnie skutecznie prowadzony.



Rys. 23. Schemat uczenia z nauczycielem

Sieć neuronowa uczona w trybie z nauczycielem służy zazwyczaj do zamodelowania nieznanej zależności pomiędzy zbiorem zmiennych wejściowych, a zbiorem zmiennych wyjściowych. Gdyby ta zależność była znana (w dowolnej formie), to właściwa procedura budowy modelu polegała by na stworzeniu algorytmu według znanych zasad generowania sygnałów wyjściowych na podstawie wartości sygnałów wyjściowych i przetworzeniu tego algorytmu w komputerowy program symulacyjny. W takim przypadku użycie sieci neuronowej nie było by konieczne – chociaż niekiedy także i w tych przypadkach sięgamy niekiedy do sieci neuronowych ze względu na ich zalety obliczeniowe (modele neuronowe pozwalają zwykle na szybsze symulowanie rzeczywistości, a to prowadzi do znaczących korzyści w momencie praktycznego korzystania z modelu – na przykład daje możliwości przebadania większej liczby wariantów przed podjęciem ostatecznej decyzji). Skoro jednak w większości zadań wchodzących w obszar *data mining* nie są nam znane zależności (przyczynowo-skutkowe albo dowolne inne) pozwalające na obliczanie interesujących (wyróżnionych) danych wyjściowych na podstawie znanych danych wejściowych – to sięgamy do modelu neuronowego. Przy budowie tego modelu odwołujemy się bowiem wyłącznie do eksplorowanych danych, licząc na to, że ich analiza pozwoli zbudować model, używany potem jako skuteczne narzędzie diagnostyczne, prognostyczne lub kontrolne. Proces uczenia sieci ma więc pozwolić na wydobycie wiedzy zawartej *implicite* w rozważanych (eksplorowanych) danych, przy czym sensowność tego procesu związana jest głównie z faktem, że zazwyczaj (ale nie zawsze!) wiedza taka może być potem uogólniona na przypadki, które nie wchodziły w skład eksplorowanej bazy danych.

Schemat procesu uczenia jest prosty: po wprowadzeniu na wejścia uczonej sieci wartości wejściowych pochodzących z eksplorowanej bazy danych (z pierwszego przypadku uczącego) wyznaczana jest – poprzez wykonanie wszystkich operacji wynikających ze struktury i aktualnych parametrów sieci wartość wyjściowa (wartości wyjściowe), będące odpowiedzią sieci na podane do niej dane. W opisywanym pierwszym kroku wszystkie parametry sieci przyjmują wartości przypadkowe (generowane automatycznie przez specjalny mechanizm losowy), co oczywiście powoduje, że obliczone wartości zmiennych wyjściowych

z sieci na ogół nie odpowiadają tym wartościom, jakich chcielibyśmy oczekiwać zgodnie z zadaniem stawianym modelowi. Dla polepszenia jakości działania sieci wyznaczane są więc wartości błędów, które ustala się w taki sposób, że wartości obliczone przez sieć są porównywane z rzeczywistymi (poprawnymi) wartościami zawartymi w zbiorze uczącym. Jeśli wartości rzeczywiste i teoretyczne (obliczona przez sieć) są różne, to wagi neuronów są modyfikowane w sposób zapewniający zmniejszenie błędu sieci.

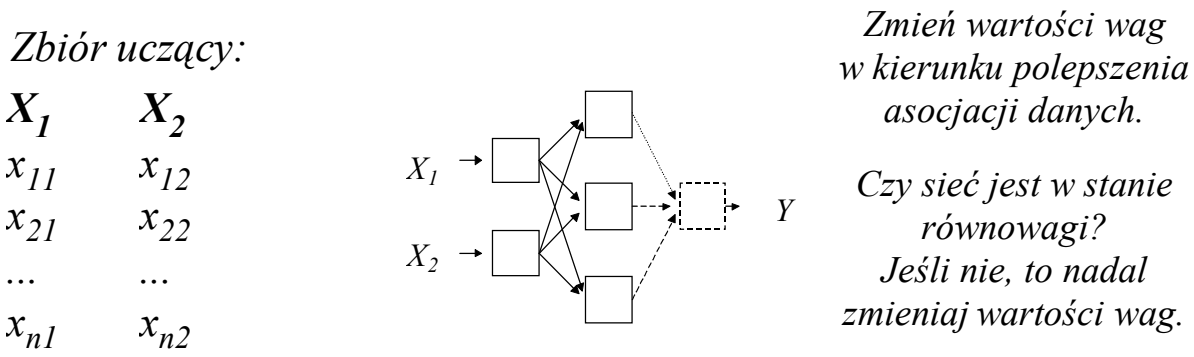
W podobny sposób postępuje się podczas prezentacji wszystkich pozostałych przypadków uczących, oczywiście wykorzystując w każdym kolejnym kroku wartości parametrów (wag) już wcześniej ulepszone w poprzednich krokach uczenia. Dzięki temu sieć stopniowo poprawia swoje działanie, coraz lepiej rozwiązując postawiony problem. Zdarza się przy tym oczywiście, że poprawki wprowadzone dla kolejnych przykładów uczących *pogarszają* wyniki, jakie można by było wcześniej uzyskać dla uprzednio pokazanych danych uczących, w związku z tym po jednorazowym przejrzeniu wszystkich rozważanych przykładów (co zwykle bywa nazywane pojedynczą *epoką* procesu uczenia), proces uczenia zakłada ponowne przeglądanie wszystkich danych uczących i ponowne korygowanie wag. Jak już wyżej wspominano, takich epok w typowym procesie uczenia zwykle bywa kilkaset, chociaż niekiedy dla uzyskania pożądanej jakości modelu trzeba zastosować kilka tysięcy epok. W wyniku procesu uczenia *wielokrotnie* angażującego wszystkie posiadane dane uczące dochodzi do wypracowania w sieci swoistego kompromisu: wartości parametrów nauczonej sieci są *wypadkową* wielu (często sprzecznych) dążeń. Dzięki temu proces uczenia sieci może prowadzić do wyeliminowania wpływu pojedynczych błędów lub zakłóceń występujących w danych użytych do uczenia – gdyż ta *wiedza,* jaką sieć wypracuje w następstwie uczenia trwającego wiele epok, będzie głównie wiedzą o cechach występujących w danych w sposób stabilny i powtarzalny. Efekty jednostkowe i zjawiska incydentalne nie mają wpływu na kształt uformowanego w sieci modelu rozważanego zjawiska lub procesu.

Ma to dwojakiego rodzaju konsekwencje. Po pierwsze nauczona sieć potrafi wykorzystywać zgromadzoną w niej wiedzę do rozwiązywania zagadnień, które bezpośrednio nie figurowały w zbiorze danych użytych do uczenia (jest to tak zwany efekt *generalizacji* wiedzy zdobytej w toku uczenia). Podczas generalizacji sieć opiera się na wiedzy o *najbardziej typowych* zjawiskach zachodzących w modelowanym systemie, podając odpowiedź będącą próbą przewidzenia najbardziej prawdopodobnej odpowiedzi. W większości zadań jest to właśnie dokładnie to, o co nam chodzi, gdyż zwykle próbujemy przewidzieć, jaka wartość zmiennych wyjściowych zaistnieje *najprawdopodobniej* w określonej sytuacji. Możliwe jest jednak także użycie rozważanego tu neuronowego modelu do **wykrywania sytuacji niezwykłych.** Taki neuronowy „detektor nowości" może działać w taki sposób, że po nauczeniu na podstawie danych zawartych z zbiorze uczącym w każdej nowej sytuacji, z którą się spotka, usiłuje przewidzieć najbardziej prawdopodobne wartości zmiennych wyjściowych na podstawie znanych wartościach zmiennych wejściowych. Jeśli przewidywanie jest w miarę trafne (obliczony błąd jest niewielki) – sytuacja jest ignorowana. Takich dobrze przewidywalnych (a więc nieistotnych) sytuacji jest oczywiście bardzo dużo – i przez cały czas ich obserwacji neuronowy „detektor nowości" milczy. Jeśli jednak pojawi się duża rozbieżność pomiędzy wartością przewidywaną przez neuronowy model a wartością naprawdę zaobserwowaną – wysyłany jest sygnał napotkania nowego zjawiska.

Warto zauważyć, że taka „detekcja niezwykłych zjawisk" może być przeprowadzona także w zbiorze uczącym i może być jedną z użytecznych form eksploracji danych. Mając „uśredniony" model neuronowy można sprawdzić (da się to także robić „w biegu", w trakcie procesu uczenia!), które z danych zebranych w bazie, wyraźnie odbiegają od modelu. Wykrycie i wydzielenie tych danych może prowadzić do różnych ciekawych wniosków

praktycznych, włącznie z tym, że można sobie wyobrazić proces eksploracji danych, w wyniku którego powstaną *dwa* neuronowe modele. Pierwszy model będzie obejmował obraz *dominującej* tendencji, uzyskany po usunięciu „odstających" danych, a drugi będzie zbudowany wyłącznie na podstawie danych „odstających". Jeśli oba modele będą dobrze objaśniały odpowiednie podzbiory eksplorowanych danych – może to być ważny argument przemawiający za przyjęciem hipotezy o heterogenicznym charakterze badanych danych.

Opisana wyżej (bardzo powierzchownie, celem uzupełnienia wiadomości patrz np. [Tadeusiewicz, 1993]) metoda uczenia sieci neuronowej nazywana bywa (jak wspomniano wyżej) „uczeniem z nauczycielem". Jest to pewien skrót myślowy, gdyż w istocie żadnego personalizowanego „nauczyciela" w całym tym procesie nie ma. Jednak rolę nauczyciela przyjmuje w rozważanej grupie metod zawartość eksplorowanej bazy danych, w której zawarte są przykłady zarówno danych podawanych na wejście sieci, jak i danych, z którymi można porównywać sygnały wyjściowe, produkowane przez sieć. W rezultacie możliwe jest ocenianie (w trakcie procesu uczenia) jakości odpowiedzi generowanych przez sieć, co przypomina w jakimś stopniu korygujące napomnienia nauczyciela, formującego wiedzę niesfornego ucznia – stąd nazwa tej metody uczenia. Inne podejście stosowane jest przy **uczeniu bez nauczyciela** (rys. 24).



Rys. 24. Schemat uczenia bez nauczyciela (Uwaga: sieci tego typu nie zawsze wypracowują ujednolicony sygnał wyjściowy – często odpowiedzią sieci jest rozkład pubudzeń w warstwie ukrytej)

W tym przypadku zbiór uczący zawiera tylko wartości zmiennych wejściowych, nie ma więc w nim żadnych informacji dotyczących wartości, które powinny pojawić się na wyjściu (wyjściach) sieci. Uczenie polega w tym przypadku na cyklicznym prezentowaniu danych uczących i na stopniowej, systematycznej modyfikacji wag, prowadzącej w efekcie do wytworzenia w sieci pewnej wiedzy o ogólnych cechach i właściwościach zbiorowości wejściowych sygnałów. Najczęściej celem samouczenia jest powiązanie struktury wiedzy, reprezentowanej wewnątrz sieci przez współczynniki wag synaptycznych, z ukrytą (nie znaną a priori) strukturą eksplorowanych danych.

Ponieważ cel samouczenia sieci jest określony bardzo ogólnie, więc niemożliwe jest wskazanie kryterium, którego spełnienie upoważniało by nas do zakończenia tego procesu. W przypadku uczenia z nauczycielem sprawa jest prostsza: proces uczenia przerywa się, gdy obserwowany błąd popełniany przez sieć staje się (ogólnie mówiąc) akceptowalny. Jest przy tym kilka subtelności, np. można dyskutować, czy chodzi o błąd popełniany przez sieć podczas przetwarzania danych uczących, czy też o błąd popełniany na innych danych (nie używanych bezpośrednio w procesie uczenia), tak zwanych danych walidacyjnych. Najczęściej to drugie rozwiązania jest uważane za właściwsze, chociaż nie uwalnia nas ono od wszelkich trudności, związanych z problemem zatrzymania uczenia. Przy samouczeniu problem jest poważniejszy, gdyż nie istnieje (w żadnej formie) odpowiednik wzorcowego

rozwiązania (prawdę mówiąc właśnie dlatego stosujemy samouczenie, że nie wiemy, co właściwie powinno zostać osiągnięte), zatem nie istnieje pojęcie błędu popełnianego przez sieć ani żadna inna miara określająca jakość rozwiązania, które sieć osiągnęła w wyniku samouczenia. Dlatego zwykle pozwalamy sieci uczyć się (poprzez przeglądanie zbioru wejściowych danych) tak długo, jak długo mamy ochotę, po czym proces ten arbitralnie przerywamy, uznając, że to co zostało osiągnięte, jest właśnie tym, co chcieliśmy mieć.

Czasami także stosuje się pewne formalne „kryterium stopu", wiążąc je najczęściej z faktem osiągnięcia przez sieć pewnego stanu równowagi. Czysto umownie możemy przy tym założyć, że stan równowagi przejawia się tym, że przy kolejnych prezentacjach tych samych przypadków uczących wyniki działania sieci są identyczne oraz, że zastosowany algorytm uczenia nie wskazuje na potrzebę modyfikacji wartości wag.

Sieci uczone w trybie bez nauczyciela stosowane są przy rozwiązywaniu zupełnie innych typów zadań niż sieci uczone z nauczycielem. Przede wszystkim są to (jak już wyżej pisano) zadania związane z klasyfikacją *bezwzorcową*, mającą na celu rozpoznanie struktury analizowanego zbioru obiektów czy też identyfikacji jednorodnych fragmentów szeregów czasowych. Samouczące się sieci Kohonena pozwalają także zorientować się w jakościowych relacjach i stosunkach charakteryzujących wielowymiarowy zbiór danych, przy czym jedną z interesujących cech tych sieci jest możliwość używania ich (po procesie samouczenia) do rzutowania (projekcji) wielowymiarowych zbiorów danych do przestrzeni o niskiej wymiarowości (zwykle na dwuwymiarową płaszczyznę ekranu komputera) w celu wygodnej do wzrokowych analiz wizualizacji tych zbiorów danych i ich wybranych cech.

3.4. Uczenie perceptronu wielowarstwowego jako klasyczny przykład uczenia sieci w trybie z nauczycielem

Perceptron wielowarstwowy (oznaczany skrótem MLP i nazywany również jednokierunkową siecią wielowarstwową) należy do najlepiej zbadanych i najczęściej wykorzystywanych w praktyce modeli neuronowych. Dokładny opis tego typu sieci można znaleźć między innymi w [Tadeusiewicz, 1993].

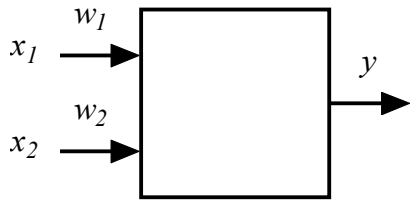
Perceptron jest siecią *wielowarstwową*, *jednokierunkową*, wykorzystującą neurony agregujące dane wejściowe poprzez *obliczenie sumy ważonych wejść* (wzór (1)) i wyznaczające swoje wartości wyjściowe za pomocą funkcji *liniowej*, *sigmoidalnej* lub *tangensoidalnej*. Połączenia występujące w tej sieci są zwykle typu *każdy z każdym*. Uczenie sieci przeprowadzane jest w *trybie z nauczycielem*.

Najważniejszym elementem procedury uczącej jest przyjęcie właściwego sposobu modyfikacji wag neuronów. Przyjęty w tym zakresie sposób postępowania powinien przede wszystkim prowadzić do zmniejszenia rozbieżności pomiędzy rzeczywistymi wartościami zmiennych wyjściowych i wartościami wyznaczanymi przez sieć. Klasyczną metodą wykorzystywaną do uczenia sieci perceptronowych jest *metoda wstecznej propagacji błędów*. Jej idea jest następująca: po wyznaczeniu błędów dla neuronów wyjściowych (na podstawie danych uczących w ich części dotyczącej pożądanych wartości wyjściowych) informacja o zaistniałym zróżnicowaniu pomiędzy wartościami rzeczywistymi i teoretycznymi przekazywana jest do wcześniejszych warstw sieci. W istocie zatem mamy do czynienia z przepływem informacji o błędach w kierunku przeciwnym (od warstwy wyjściowej do wejściowej) w stosunku do zasadniczego kierunku przepływu informacji o danych (od wejścia do wyjścia) - stąd w nazwie wspomnianej metody uczenia pojawia się określenie *„wsteczna" propagacja błędów*.

Wykorzystywana w omawianej metodzie technika modyfikacji wartości współczynników wagowych należy do grupy *gradientowych metod optymalizacji*. Jej idea zostanie zilustrowana na przykładzie uczenia pojedynczego neuronu o dwóch wejściach (rys. 25). Niech zastosowany przypadek uczący składa się z dwóch wartości zmiennych wejściowych: x_1, x_2 i odpowiadającej im wartości zmiennej wyjściowej d. Celem uczenia ma być dobór wag neuronu w_1 i w_2 zapewniający minimalizację błędu, czyli rozbieżności pomiędzy zaobserwowaną, rzeczywistą wartością d i wyznaczoną na wyjściu neuronu wartością y.

Wartość y wyznaczana na wyjściu neuronu jest zależna zarówno od sygnałów wejściowych x_1, x_2 jak i od wag neuronu w_1 i w_2:

$$y = \varphi\,(\,(x_1, x_2), (w_1, w_2)\,)$$



Rys. 25. Schemat neuronu poddawanego uczeniu w trybie z nauczycielem.

W charakterze miary błędu przyjmuje się zwykle kwadrat różnicy pomiędzy wartością teoretyczną i rzeczywistą (tzw. *błąd SE*), czyli:

$$SE = (y - d)^2 \tag{7}$$

a dla całego zbioru uczącego, złożonego z wielu przykładowych kompletów danych wejściowych $(x_1, x_2)_i$ oraz dopasowanych do nich wzorcowych wartości d_i – właściwą miarą błędu będzie **suma** kwadratów różnic pomiędzy wartościami obliczanymi

$$SSE = \sum_i (y_i - d_i)^2 \,. \tag{8}$$

Dla rozpatrywanego przypadku uczącego wartość błędu uzależniona jest od wartości wag, gdyż one wpływają na wartość wyznaczoną na wyjściu neuronu. Zbiór wartości błędów popełnianych przez neuron przy różnych wartościach jego parametrów może być rozpatrywany jako funkcja dwóch zmiennych:

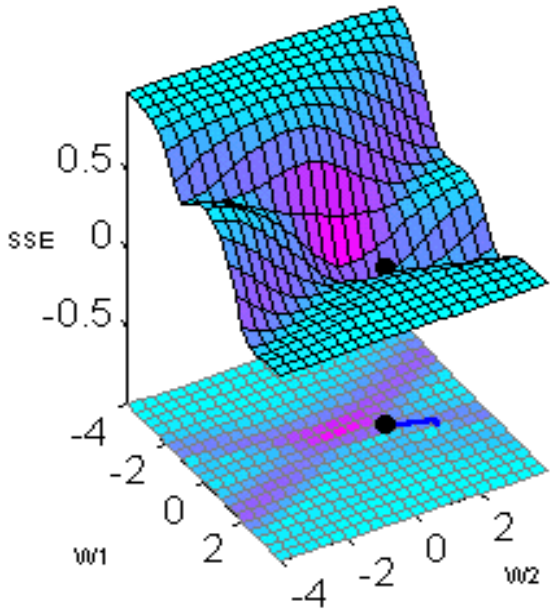
$$SSE = E(w_1, w_2), \tag{9}$$

bo każda odpowiedź neuronu zależy od obydwu sygnałów wejściowych i od wartości obydwu współczynników wagowych na jego dwóch wejściach:

$$y_i = \varphi\,(\,(x_1, x_2)_i, (w_1, w_2)\,)$$

ale po zsumowaniu wszystkich dostępnych sygnałach wejściowych $(x_1, x_2)_i$ pozostaje wyłącznie zależność od wag (w_1, w_2). Zależność ta może być graficznie przedstawiona jako pofałdowana powierzchnia rozpostarta nad przestrzenią wag. Przykładowy sposób kształtowania się tego typu zależności przedstawia rys. 26. Celem uczenia neuronu, przy przedstawionej tu interpretacji jego błędów, jest znalezienie takiej kombinacji jego parametrów (w_1, w_2), która gwarantuje osiągnięcie najniższego punktu rozważanej powierzchni – czyli najmniejszej wartości średniokwadratowego błędu popełnianego przez neuron.

Rozumowanie dla sieci zawierającej wiele neuronów jest w pełni analogiczne, tylko przestrzeń, nad którą rozpięta jest powierzchnia błędów złożona jest ze wszystkich wartości wszystkich współczynników wagowych we wszystkich neuronach. Jest to oczywiście przestrzeń o bardzo dużej liczbie wymiarów, praktycznie niemożliwa do wyobrażenia (ani do narysowania) – co nie zmienia faktu, że intuicje przedstawione wyżej funkcjonują w tej przestrzeni dokładnie tak samo, to znaczy istota procesu uczenia polega na znalezieniu takiej kombinacji wszystkich wag, która gwarantuje najmniejszą wartość błędu *SSE*.



Rys. 26. Zależność błędu neuronu od wartości współczynników wagowych

Uczenie (zarówno pojedynczego neuronu, jak i całego perceptronu wielowarstwowego) rozpoczyna się od losowo wybranych wartości wag, którym odpowiada pewien punkt położony na powierzchni reprezentującej błąd. Przeprowadzany proces uczenia może być rozpatrywany jako *wędrówka* od wyznaczone w ten sposób *punktu początkowego* położonego gdzieś przypadkowo na powierzchni błędu, do *punktu optymalnego* znajdującego się na niej najniżej. Wyznaczenie wag odpowiadających punktowi optymalnemu (położonemu najniżej na powierzchni błędu, a więc wiążącemu się z najdokładniejszym odwzorowywaniem przez sieć wymaganych cech szukanego rozwiązania) zapewni najlepsze, możliwe do osiągnięcia działanie neuronu – lub całej sieci.

Jak już wspomniano powyżej, przedstawiona metoda uczenia należy do grupy metod *gradientowych*, gdyż wykorzystuje informację o gradiencie funkcji błędu wyznaczonym w aktualnie rozpatrywanym punkcie ulokowanym na powierzchni błędu. Wyznaczony gradient opisanej wyżej funkcji E względem jej argumentów:

$$\nabla E = \left\langle \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2} \right\rangle \tag{10}$$

może być interpretowany jako wektor w przestrzeni wag, wskazujący kierunek, w którym wartość funkcji błędu najszybciej rośnie. Ponieważ celem uczenia jest *zmniejszanie* wartości błędu, więc w trakcie poruszania się po rozpatrywanej powierzchni wykonuje się krok w kierunku dokładnie przeciwny do kierunku wyznaczonego przez gradient funkcji błędu. W związku z tym zmodyfikowany w t-tym kroku uczenia wektor wag może być przedstawiony następująco:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathbf{E}(\mathbf{w}_t). \tag{11}$$

Interpretacja podanego wzoru jest następująca: wektor wag w kroku *t+1* jest równy wektorowi wag z kroku poprzedniego zmodyfikowanemu o poprawkę zmierzającą do zmniejszenia błędu. Poprawka ta jest liczona jako część (wyrażona przez dodatni, ale mniejszy od jedności *współczynnik uczenia* α) wektora *przeciwnego* do gradientu wyznaczonego w punkcie odpowiadającym wagom $\mathbf{w}_t$ (oznaczonego jako $\nabla E(\mathbf{w}_t)$). Na prawidłowy przebieg uczenia w dużym stopniu wpływa wartość współczynnika uczenia α, gdyż zastosowanie zbyt małej wartości tego parametru spowoduje, że kolejne kroki będą bardzo małe, co może bardzo spowolnić proces uczenia. Z kolei zastosowanie zbyt dużej wartości współczynnika uczenia może wpłynąć na wykonywanie za dużych kroków po powierzchni błędu, co może uniemożliwić precyzyjne osiągnięcie punktu położonego najniżej – zwłaszcza przy jej skomplikowanym (niekiedy) kształcie.

3.5. Ulepszenia metody uczenia perceptronu

Opisana wyżej, bardzo prosta koncepcyjnie, ale uciążliwa obliczeniowo (dla uzyskania poprawnego wyniku trzeba wykonać setki lub nawet miliony elementarnych poprawek) metoda wstecznej propagacji błędów poddawana była wielu modyfikacjom. Podstawowa idea pozostawała przy tych modyfikacjach niezmieniona – każdorazowo należy dotrzeć do punktu położonego najniżej na powierzchni błędu, rozpoczynając wędrówkę od wybranego (zwykle) w sposób losowy punktu startowego. W różnych modyfikacjach przedstawionego powyżej rozwiązania klasycznego stosowane są jednak zróżnicowane metody określania **kierunku**, w którym wykonywany będzie kolejny krok (nie zawsze jest to dokładnie kierunek przeciwny do gradientu funkcji błędu), bądź też podejmowane są próby optymalizowania **wielkości wykonywanego kroku** (który w rozwiązaniu klasycznym jest określany arbitralnie poprzez zdefiniowanie wartości współczynnika uczenia α). Wszystkie stosowane modyfikacje podejścia klasycznego maja na celu głównie zwiększenie efektywności obliczeniowej procedury uczącej - tak, aby można było w krótszym czasie dotrzeć do punktu odpowiadającego minimalnej wartości błędu.

Należy w tym miejscu wskazać jednak na podstawowy problem związany z uczeniem sieci perceptronowych - jest nim **możliwość zatrzymania procedury uczącej w punkcie stanowiącym minimum lokalne (a nie globalne) funkcji błędu**. Z uwagi na nieliniowy charakter modeli neuronowych powierzchnia błędu (prezentująca wartość błędu w zależności od parametrów modelu - czyli wag) jest często bardzo pofałdowana i posiada szereg minimów lokalnych. Taki sposób ukształtowania powierzchni błędu powoduje, że w zależności od wybranego punktu początkowego, poruszając się zawsze „w dół" można dotrzeć do różnych minimów **lokalnych**, w których proces uczenia się zatrzyma (każdy kierunek wychodzący z minimum prowadzi w górę, a więc powoduje pozorne pogorszenie jakości znalezionego rozwiązania). Taka sytuacja zdarza się podczas uczenia sieci neuronowych raczej często i jest groźna, gdyż utknąwszy w minimum (nie wiedząc i nie mogąc wiedzieć o tym, że jest ono minimum lokalnym) przerywamy proces uczenia i w związku z tym nigdy nie osiągniemy punktu, który rzeczywiście położony jest najniżej[2].

[2] Problem ten nie występuje w modelach liniowych, w których powierzchnia błędu ma zawsze postać paraboloidy. Taka gładka i regularna postać funkcji błędu gwarantuje, że poruszając się w kierunku przeciwnym do gradientu, zawsze dotrzemy do minimum globalnego funkcji błędu niezależnie od położenia punktu początkowego. Jest to jednak wyjątek potwierdzający regułę, a regułą tą jest stałe zagrożenie małą skutecznością uczenia, wywołane przez istnienie minimów lokalnych funkcji błędów.

Podejmowane są różnorodne próby przezwyciężania problemu minimów lokalnych. Ich stosowane prowadzi jednak tylko do zmniejszenia (często w bardzo dużym stopniu) możliwości zatrzymania algorytmu uczenia sieci w minimum lokalnym, ale w praktyce całkowite wyeliminowanie tego problemu jest niemożliwe. Walka z minimami lokalnymi wymaga zwykle znacznego wydłużenia czasu obliczeń. Do powszechnie stosowanych metod zwiększających prawdopodobieństwo wyznaczenia wag sieci, zapewniających osiągnięcie minimum **globalnego** funkcji błędu należy zaliczyć:

a) wielokrotną realizację procedury uczącej. Rozpoczynając każdorazowo wędrówkę od innego, losowo wybranego punktu, zwiększamy szanse na to, że przynajmniej jedna z trajektorii zmian wag ominie pułapki minimów lokalnych i dotrze do minimum globalnego. Jako wynik ostateczny takiej wielokrotnie ponawianej procedury uczącej, przyjmuje się najlepsze z uzyskanych rozwiązań. Zwiększając liczbę powtórzeń zwiększamy szanse na odnalezienie najlepszego zestawu wag, ale jednocześnie zwiększamy czas obliczeń;
b) stosowanie metod uczenia zawierających *człon momentum*. Przy stosowaniu tej grupy metod **kierunek**, w którym wykonywany jest kolejny krok po powierzchni błędu, jest wypadkową dwóch czynników: kierunku wyznaczonego na podstawie gradientu w aktualnie osiągniętym punkcie (wyznaczanie tego kierunku odbywa się w sposób opisany powyżej) oraz kierunku zmian wag zastosowanego w trakcie **poprzedniego** kroku. Siła wpływu na obecnie podejmowaną decyzję kierunku wykorzystywanego przy poprzedniej modyfikacji wag, określana jest przez współczynnik, który (ze względu na swój zachowawczy charakter) zyskał nazwę *momentum*[3]. Zastosowanie opisanej techniki zwiększa bezwładność metody poruszania się po powierzchni błędu – co z kolei powoduje, że pojawiające się niewielkie, lokalne pofałdowania powierzchni błędu nie są w stanie zakłócić zasadniczego kierunku wędrówki. Dodatkowym atutem metod wykorzystujących informacje o historii zmian jest przyspieszenie uczenia sieci poprzez zmniejszenie efektu „myszkowania" podczas zmierzania do minimum globalnego.
c) wprowadzanie szumu do danych uczących. W kolejnych epokach uczenia, przy prezentacji przypadków uczących nie wprowadza się zawsze takich samych wartości wejściowych, lecz za każdym razem oryginalne wartości zmiennych wejściowych zaburza się (w niewielkim stopniu) poprzez dodanie do nich losowych, niewielkich, skupionych wokół zera wartości. Ciągłe zaburzanie wartości wejściowych sieci powoduje, że uformowanie powierzchni błędu w trakcie uczenia nieznacznie się zmienia. Takie postępowanie powoduje, że występujące na powierzchni błędu minima lokalne mogą po pewnym czasie zniknąć i proces uczenia będzie mógł być w dalszym ciągu kontynuowany w kierunku minimum globalnego.
d) losowe zaburzanie osiągniętego zestawu wag. Po osiągnięciu minimum funkcji błędu (o którym nie wiadomo, czy ma charakter lokalny, czy też globalny) do uzyskanego wektora wag dodawane są niewielkie wartości losowe, a następnie kontynuowany jest proces uczenia. Przypomina to próbę wykopnięcia piłki z dołka. Jeśli osiągnięty wcześniej punkt stanowił minimum globalne, to wznowiony proces uczenia doprowadzi z powrotem do tego samego rozwiązania. Jeżeli jednak uzyskane rozwiązanie miało charakter lokalny, to wprowadzone zaburzenie może pozwolić na opuszczenie tego minimum lokalnego i uzyskanie lepszego rozwiązania. Niestety, przedstawiona metoda może również prowadzić do wielokrotnego powracania do tego samego rozwiązania lokalnego, a w skrajnie niekorzystnym przypadku może się zdarzyć, że zaburzenie wyrwie proces

[3] Termin *momentum* zaczerpnięty jest z fizyki – tym słowem w języku angielskim określa się wielkość fizyczną nazywaną po polsku *pędem*. Pęd powoduje, że ciało znajdujące się w ruchu w pewnym kierunku niechętnie zmienia kierunek ruchu, co stabilizuje (na przykład) narciarza zjeżdżającego w dół po nierównym stoku. Technika *momentum* w algorytmach uczenia sieci neuronowych odgrywa podobną stabilizującą rolę.

uczenia z dobrego minimum globalnego i uwięzi go w przypadkowym (blisko położonym) minimum lokalnym.

Przedstawione wyżej metody stanowią tylko przykłady różnych technik, zaprojektowanych z myślą o rozwiązywaniu problemu minimów lokalnych. Czasami dobre rezultaty można otrzymać stosując kilka podejść jednocześnie. Przy właściwym przeprowadzeniu uczenia pomocne jest więc doświadczenie badacza jak również właściwie dobrany program, który w zależności od sytuacji będzie w stanie samodzielnie zadecydować o najwłaściwszym sposobie przeprowadzenia uczenia sieci.

3.6. Problem optymalizacji struktury perceptronu

Omawiając problematykę związaną z perceptronami wielowarstwowymi nie można pominąć zagadnień związanych z doborem ich właściwej struktury. Liczba neuronów wchodzących w skład warstwy wejściowej i wyjściowej jest zwykle zdeterminowana przez typ rozwiązywanego problemu oraz liczbę i rodzaj wykorzystywanych zmiennych. Od decyzji konstruktora sieci zależy natomiast struktura części ukrytej - liczba warstw ukrytych i liczba neuronów w poszczególnych warstwach. Prawidłowe określenie tych parametrów ma decydujące znaczenie dla prawidłowego działania utworzonego modelu.

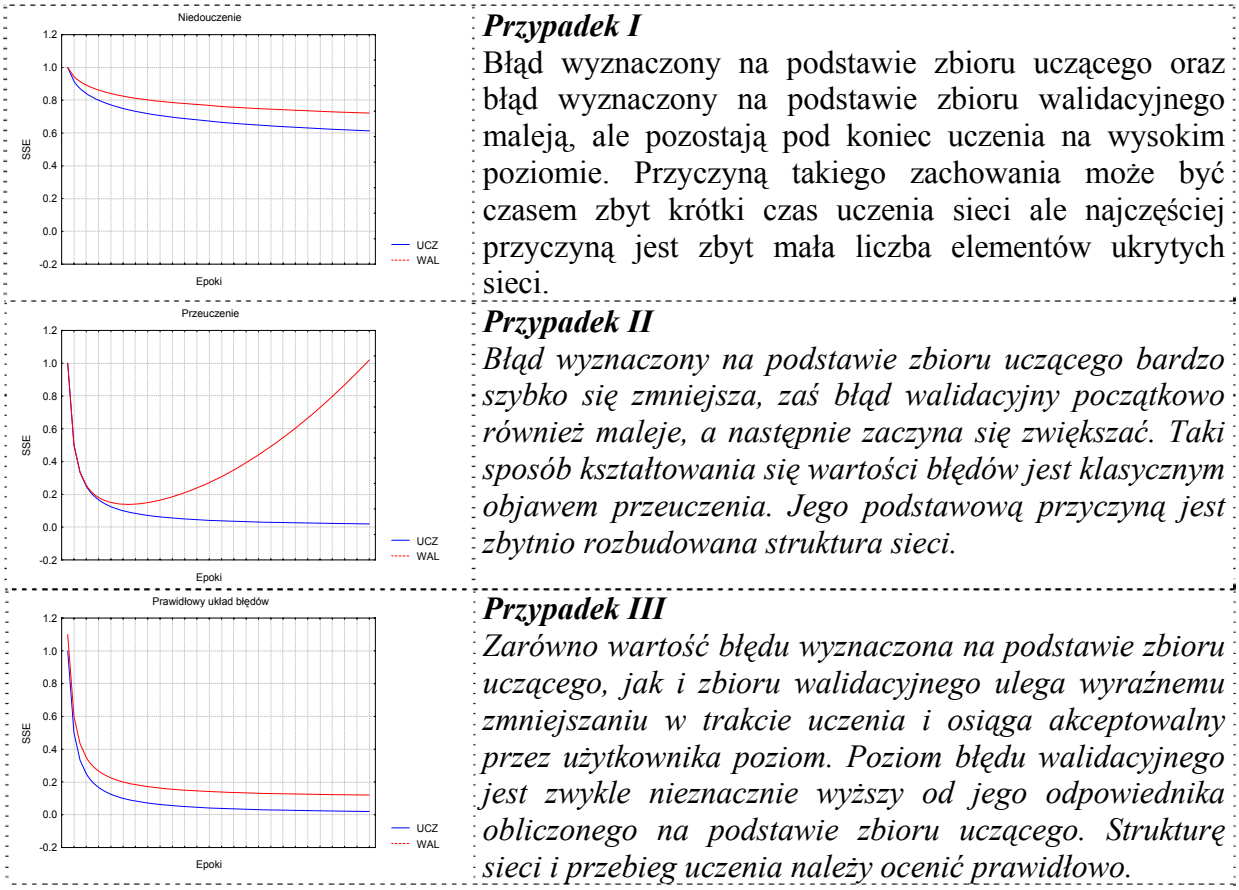
Pamiętając, że wynik działania sieci perceptronowej jest złożeniem rezultatów działania składających się na nią neuronów, można oczekiwać, że zwiększając liczbę neuronów wzbogacimy możliwości oferowane przez całą sieć. Efekt ten rzeczywiście występuje i jest najlepiej widoczny w trakcie uczenia sieci - im więcej elementów ukrytych wchodzić będzie w skład uczonej sieci, tym mniejsze będą wartości błędów dla przypadków wchodzących **w skład zbioru uczącego**. Tym samym można wykazać, że wraz ze wzrostem liczby elementów budujących sieć zwiększa się posiadana przez neuronowy model *zdolność do aproksymacji* (określana jako zdolność do prawidłowego działania sieci **dla danych prezentowanych w trakcie uczenia**).

Nieograniczone zwiększanie posiadanej przez model zdolności do aproksymacji nie zawsze jest jednak pożądane, gdyż przeprowadzana w tym celu rozbudowa sieci może doprowadzić do pogorszenia się sposobu działania sieci dla danych **różnych** od tych, które prezentowane były w czasie uczenia. Sytuacja taka określana jest jako *zanik zdolności do uogólniania* (*generalizacji*). Zjawisko to bywa następstwem nadmiernie rozbudowanej struktury sieci (zbyt dużej liczby neuronów zaangażowanych w jej warstwach ukrytych) i manifestuje się szczególnie w przypadku, kiedy proces uczenia prowadzony jest tak długo, aż błędy wykrywane podczas tego procesu znikną prawie do zera. Z tego powodu fenomen utraty zdolności do generalizacji, połączony z bardzo dobrymi wynikami oceny sieci w trakcie procesu uczenia, nazywany jest też niekiedy *przeuczeniem sieci*. Przyjrzymy się nieco dokładniej temu zjawisku, gdyż ma ono znaczący wpływ na praktyczną użyteczność sieci.

Zdolność do generalizacji (czyli do generowania prawidłowej odpowiedzi po wprowadzeniu danych, które nie były prezentowane w trakcie uczenia) jest podstawową i najważniejszą cechą poprawnie skonstruowanego modelu neuronowego, a warunkiem jej osiągnięcia jest (oprócz poprawnie przeprowadzonego uczenia) jest **prawidłowe określenie struktury części ukrytej sieci**. Obowiązująca w tym zakresie zasada jest prosta: liczba elementów ukrytych sieci musi być na tyle duża, aby sieć prawidłowo aproksymowała, ale jednocześnie nie może być zbyt duża, aby sieć nie utraciła zdolności do generalizacji. Niestety, proponowany kompromis jest często trudny do zlokalizowania, z reguły bowiem nie wiemy, ile elementów potrzeba aby sieć prawidłowo aproksymowała (czasem nie udaje się uzyskać poprawnej aproksymacji nawet za cenę bardzo dużej rozbudowy struktury sieci), niezwykle trudno jest też wychwycić moment, w którym liczba elementów sieci staje się za

duża, co grozi utratą zdolności do generalizacji. Zatem ideał w postaci sieci, której struktura jest dokładnie „na miarę" rozwiązywanego problemu jest bardzo trudny do osiągnięcia w praktyce badawczej i z reguły wymaga zastosowania wielu eksperymentów obliczeniowych, pozwalających na ocenę różnych wariantów sieci. Konieczne jest również zaprojektowanie metody oceny posiadanej przez aktualnie badaną sieć zdolności do aproksymacji i jej zdolności do generalizacji. Zwykle problem ten rozwiązywany jest poprzez podzielenie posiadanego zbioru danych na **zbiór uczący** oraz na dane weryfikacyjne. Z danych weryfikacyjnych wydziela się zwykle dodatkowy zestaw przypadków określany jako **zbiór walidacyjny**. Reszta (zwykle niewielka) danych weryfikacyjnych zatrzymywana jest w odwodzie jako tak zwany zbiór testowy.

Zbiór uczący prezentowany jest w trakcie uczenia i wyniki uzyskane na jego podstawie są podstawą do modyfikacji wag. Elementy zbioru walidacyjnego również są prezentowane w każdej epoce uczenia - jednakże nie służą one uczeniu, lecz jedynie ocenie sieci. Błąd reprezentujący zróżnicowanie pomiędzy wartościami rzeczywistymi i teoretycznymi wyznaczony na podstawie przypadków walidacyjnych pozwala na monitorowanie przebiegu uczenia i umożliwia wykrycie przeuczenia, gdyż zbiór walidacyjny nie stanowi podstawy do modyfikacji wag sieci i dlatego jest wrażliwy na wystąpienie symptomów przeuczenia. Błędy wyznaczone na podstawie danych wchodzących w skład zbioru uczącego i walidacyjnego stanowią podstawę do oceny sieci i wskazują na poprawność przyjętej jej struktury. Można wskazać tu na trzy typowe przypadki (rys. 27)



Przypadek I

Błąd wyznaczony na podstawie zbioru uczącego oraz błąd wyznaczony na podstawie zbioru walidacyjnego maleją, ale pozostają pod koniec uczenia na wysokim poziomie. Przyczyną takiego zachowania może być czasem zbyt krótki czas uczenia sieci ale najczęściej przyczyną jest zbyt mała liczba elementów ukrytych sieci.

Przypadek II

Błąd wyznaczony na podstawie zbioru uczącego bardzo szybko się zmniejsza, zaś błąd walidacyjny początkowo również maleje, a następnie zaczyna się zwiększać. Taki sposób kształtowania się wartości błędów jest klasycznym objawem przeuczenia. Jego podstawową przyczyną jest zbytnio rozbudowana struktura sieci.

Przypadek III

Zarówno wartość błędu wyznaczona na podstawie zbioru uczącego, jak i zbioru walidacyjnego ulega wyraźnemu zmniejszaniu w trakcie uczenia i osiąga akceptowalny przez użytkownika poziom. Poziom błędu walidacyjnego jest zwykle nieznacznie wyższy od jego odpowiednika obliczonego na podstawie zbioru uczącego. Strukturę sieci i przebieg uczenia należy ocenić prawidłowo.

Rys. 27. Sposoby kształtowania się błędu dla zbioru uczącego i walidacyjnego.

Przypisanie przypadku do zbioru uczącego lub walidacyjnego odbywa się zwykle w sposób losowy, przy czym pierwszy z nich zawiera zwykle więcej (zwykle w granicach 60 -

80%) przypadków. Opisana technika oceny sieci, wykorzystująca zbiór uczący oraz walidacyjny, jest – jak już sygnalizowano wyżej - często dodatkowo wzbogacana o jeszcze jeden element, którym jest *zbiór testowy*. Stosując takie podejście dostępny zbiór danych dzielony jest na trzy zbiory: uczący, walidacyjny i testowy (podział dokonywany jest zwykle w sposób losowy, ale również w tym przypadku zbiór uczący zawiera ponad połowę wszystkich dostępnych przypadków). Rola zbioru uczącego i walidacyjnego jest identyczna jak w poprzednim przypadku. Natomiast wydzielony **zbiór testowy** nie jest w żaden sposób prezentowany sieci w trakcie uczenia (ani do uczenia jako takiego, ani do kontroli, czy nie występuje efekt przeuczenie), lecz służy wyłącznie do jej ostatecznej oceny. Można przypuszczać, że na etapie eksploatacji sieci, kiedy trzeba jej będzie użyć jako narzędzia do rozwiązywania całkiem nieznanych problemów – jakość jej działania powinna być podobna do tej jakości, jaką udało się wymierzyć za pomocą zbioru testowego. Należy jednak pamiętać, że zbiór testowy służy tylko do oceny modelu, więc błąd wyznaczony na jego podstawie nie powinien stanowić kryterium wyboru modelu spośród kilku alternatywnych rozwiązań - gdyż w takim przypadku zbiór testowy spełniałby rolę zbioru walidacyjnego.

Właściwa struktura sieci - zapewniająca zdolność do aproksymacji i do generalizacji - jest podstawowym elementem wpływającym na poprawność funkcjonowania modelu neuronowego. Przedstawione powyżej rozważania pozwalają ją ocenić, czy też ją określić jej jakość na drodze eksperymentalnej. Jest rzeczą bezsporną, że topologia poprawnie działającej sieci powinna przede wszystkim odzwierciedlać rzeczywisty charakter zależności pomiędzy zmiennymi wejściowymi a wyjściowymi. W większości przypadków rodzaj tego związku nie jest jednak dokładnie sprecyzowany w chwili rozpoczęcia badań, co powoduje konieczność stosowania opisanego wyżej podejścia eksperymentalnego.

Kończąc uwagi dotyczące określenia struktury sieci należy zwrócić uwagę na jeszcze jeden aspekt tego problemu, jakim jest związek pomiędzy strukturą sieci a liczebnością zbioru uczącego. Próba uczenia bardzo rozbudowanej sieci za pomocą zbioru uczącego o małej liczbie elementów prowadzi zwykle do wystąpienia efektu przeuczenia nawet przy dobrze zdefiniowanej strukturze sieci. W literaturze można często spotkać reguły heurystyczne głoszące, że liczebność zbioru danych wykorzystywanego w procesie uczenia, powinna kilkakrotnie przewyższać liczbę wszystkich szacowanych parametrów modelu – czyli w naszym przypadku liczbę wag występujących we wszystkich neuronach sieci. Z tego powodu, niewielka liczebność dostępnego zbioru danych stanowi często element ograniczający możliwość zastosowania sieci o bardziej rozbudowanej strukturze[4].

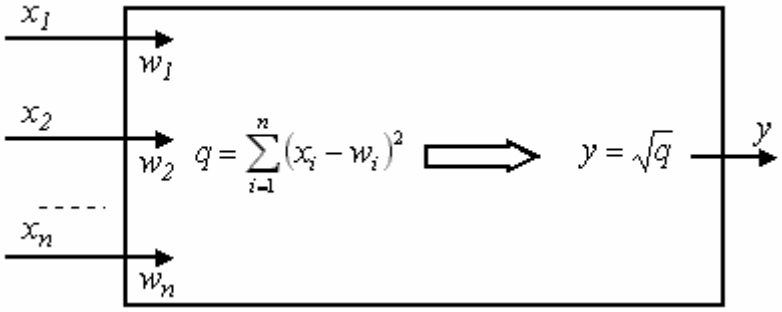
3.7. Sieć Kohonena - budowa, uczenie i zasada działania

Sieć Kohonena[5] jest jednym z najbardziej znanych typów sieci neuronowej uczonej w trybie bez nauczyciela. Jest siecią o bardzo prostej strukturze – tylko posiada dwie warstwy, a przepływ informacji w tej sieci jest ściśle jednokierunkowy. Mimo prostej budowy i nieskomplikowanych metod określających sposób jej funkcjonowania, możliwości aplikacyjne tego typu modelu są olbrzymie.

Przystępując do charakteryzowania sposobu funkcjonowania sieci Kohonena należy przede wszystkim zdefiniować sposób działania wchodzących w jej skład neuronów, gdyż jest on inny od wyżej omówionego. Neurony wchodzące w skład jej pierwszej warstwy sieci

[4] Brak możliwości zwiększenia liczebności dostępnych zbiorów danych często występuje w przypadku badania zjawisk ekonomicznych lub społecznych.
[5] Nazwa sieci wywodzi się od nazwiska jej twórcy, T. Kohonena.

Kohonena praktycznie nie dokonują przetwarzania danych - posiadają one jedynie pojedyncze wejście i pojedyncze wyjście i jedynym ich celem jest rozesłanie (poprzez ich wyjścia) do wszystkich neuronów drugiej warstwy wszystkich wartości wprowadzonych na wejścia sieci (wykorzystuje się tu znowu regułę połączeń typu „każdy z każdym" nagminnie stosowaną przy tworzeniu sieci wielowarstwowych). Funkcjonalność sieci Kohonena zapewniają więc wyłącznie neurony warstwy drugiej (wyjściowej). Najczęściej działają one w sposób przedstawiony na rys. 28.

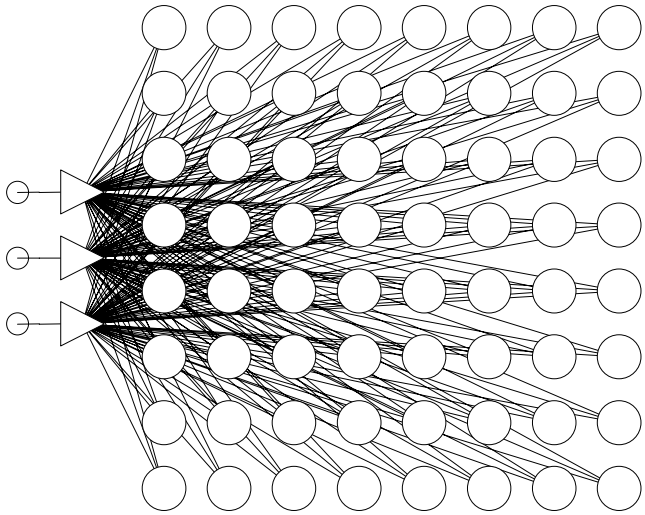


Rys. 28. Schemat neuronu wykorzystywanego w warstwie wyjściowej sieci Kohonena.

Agregacja danych wejściowych odbywa się w nich zgodnie z formułą przedstawioną we wzorze 2 - co oznacza, że zagregowana wartość wejściowa obliczana jest jako kwadrat odległości Euklidesa pomiędzy wektorem wartości wejściowych (oznaczanym symbolem **x**) a wektorem współczynników wagowych (wektor **w**)- czyli jest sumą kwadratów różnic pomiędzy wagami skojarzonymi z poszczególnymi wejściami a wartościami wprowadzanymi na te właśnie wejścia[6] (na powyższym rysunku zagregowana wartość wejściowa oznaczona jest symbolem *q*). W neuronach wykorzystywanych w sieci Kohonena wykorzystywana jest również specyficzna funkcja aktywacji - jest nią funkcja wyznaczająca wartość pierwiastka kwadratowego. Pamiętając, że w trakcie agregacji danych wejściowych wyznaczony został **kwadrat odległości Euklidesa** pomiędzy wektorami **w** oraz **x**, łatwo możemy zauważyć, że wartość uzyskiwana na wyjściu neuronu jest po prostu **odległością Euklidesa** pomiędzy wspomnianymi wektorami.

Neurony o przedstawionej charakterystyce wchodzą w skład warstwy wyjściowej sieci. Bardzo istotnym elementem architektury sieci Kohonena jest odpowiednie rozmieszczenie neuronów warstwy wyjściowej, gdyż zwykle nie są one umieszczane wzdłuż jednej prostej (jak w perceptronie wielowarstwowym), lecz są rozmieszczone na płaszczyźnie (rys. 29)

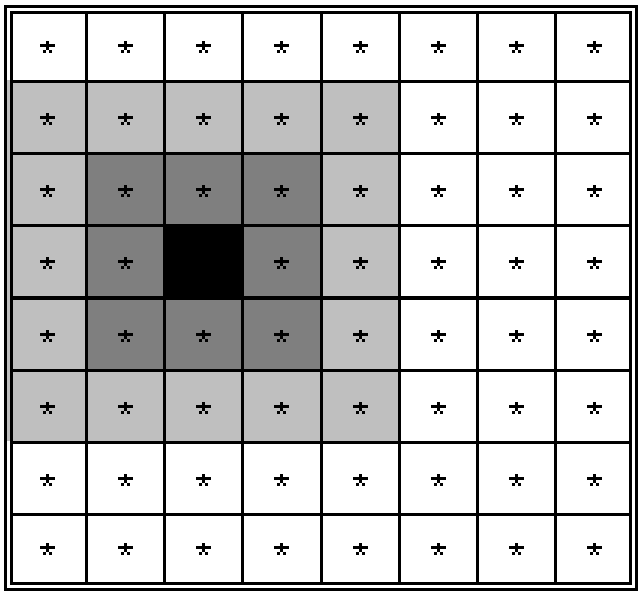
[6] Neurony o tak skonstruowanej funkcji agregującej nazywane są często *neuronami radialnymi*.



Rys. 29. Schemat przykładowej sieci Kohonena. Trójkąty symbolizują neurony warstwy wejściowej, a kółka – neurony warstwy wyjściowej.

Sposób rozmieszczenia neuronów może być różny, na przykład spore zalety ma topologia sieci przypominająca strukturę plastra pszczelego (tzw. raster heksagonalny), ale do najpopularniejszych należy rozmieszczenie neuronów w strukturze siatki prostokątnej. Przy takiej topologii neurony umieszczamy na przecięciu pewnej liczby wyznaczonych na płaszczyźnie wierszy i kolumn (właśnie taki przypadek ilustruje rys. 29).

Jednym z podstawowych pojęć związanych z siecią Kohonena jest pojęcie *sąsiedztwa*. Dotyczy ono wzajemnych relacji rozmieszczonych w przyjęty sposób neuronów, wchodzących w skład warstwy wyjściowej sieci. Przyjmowane dla omawianego typu sieci pojęcie *sąsiedztwa* nie odbiega w swoim znaczeniu od intuicyjnego znaczenia przypisywanego temu słowu - za *sąsiadów* pewnego wybranego neuronu uznaje się inne neurony znajdujące się w jego bliskości. Zasięg sąsiedztwa charakteryzowany jest przez tzw. *promień sąsiedztwa*. Rysunek 30 przedstawia przykładowy układ neuronów wyjściowych sieci. Rozpatrywać będziemy sąsiedztwo neuronu znajdującego się w trzeciej kolumnie i w czwartym wierszu (neuron ten zaznaczony został ciemnym kolorem). Sąsiedztwo o promieniu zerowym obejmuje tylko ten jeden neuron. Sąsiedztwo o promieniu jednostkowym obejmuje dany neuron oraz jego bezpośrednich sąsiadów, to znaczy neurony umieszczone w kolumnach i wierszach o numerach różniących się co najwyżej o jedną jednostkę (w sumie jest to dziewięć neuronów). Sąsiedztwo o promieniu równym dwa zawiera wszystkie te neurony, które wchodziły w skład sąsiedztwa o promieniu jeden oraz ich bezpośrednich sąsiadów (w sumie jest to 25 neuronów). W podobny sposób definiuje się (w razie potrzeby) sąsiedztwa o większej wartości promienia.



Rys. 30. Sąsiedztwo w sieci Kohonena

Warto zauważyć, że niezależnie od wyboru neuronu, dla którego definiowane jest sąsiedztwo, w przypadku sieci przedstawionej na rys. 30 sąsiedztwo o promieniu równym osiem zawsze obejmuje wszystkie neurony wchodzące w skład warstwy wyjściowej sieci.

Przystępując do budowy sieci Kohonena badacz ustala strukturę warstwy wyjściowej sieci, określa w jaki sposób rozłożone są wchodzące w jej skład neurony, wskazuje w jaki sposób one ze sobą sąsiadują. To, że jeden neuron jest sąsiadem innego wynika z ustalonego (arbitralnie!) sposobu rozmieszczenia neuronów na płaszczyźnie i **nie jest** zależne od wartości wag neuronów[7]. W czasie uczenia zmieniają się wartości wag neuronów wyjściowych, nie zmieniają się natomiast zależności sąsiedzkie.

Sieć Kohonena jest uczona w trybie *bez nauczyciela*, co oznacza, że wykorzystywany zbiór uczący obejmuje wyłącznie wartości zmiennych wejściowych i nie zawiera **żadnych** wiadomości dotyczących wartości oczekiwanych na wyjściach sieci. Uczenie ma charakter *iteracyjny* - zbiór uczący jest wielokrotnie prezentowany sieci. Uczenie rozpoczyna się od losowych, zwykle skupionych wokół zera, wartości współczynników wagowych; przy czym losowe generowanie wartości współczynników wagowych przeprowadza się po ustaleniu struktury sieci, co powoduje, że początkowe wartości wag neuronów sąsiadujących ze sobą nie pozostają w żadnym związku.

W trakcie **uczenia sieci** prezentowane są kolejne wzorce uczące. Po wprowadzeniu pierwszego z nich wartości wszystkich zmiennych wejściowych wchodzących w skład tego wzorca są przekazywane do wszystkich neuronów warstwy drugiej. Neurony wchodzące w jej skład (działając zgodnie z zasadami opisanymi powyżej) obliczają swoje sygnały wyjściowe. Każdy neuron drugiej warstwy określa swój sygnał wyjściowy niezależnie od innych neuronów tej warstwy, przy czym wartość sygnału wyjściowego określonego neuronu jest równa odległości Euklidesa pomiędzy wprowadzonym wektorem wartości wejściowych a wektorem wag rozważanego neuronu. Po przeprowadzeniu tych obliczeń nadrzędny algorytm uczący porównuje wartości wyznaczone na wyjściach poszczególnych neuronów i wybiera ten neuron, dla którego wyznaczona wartość jest **najmniejsza** (czyli ten, którego wektor wag

[7] Stosowany algorytm uczenia (opisany w dalszej części tekstu) prowadzi zwykle do takiego ukształtowania wartości współczynników wagowych, że neurony sąsiadujące ze sobą posiadają zbliżone do siebie wartości współczynników wagowych, ale to jest **skutek** a nie przyczyna zdefiniowanego sąsiedztwa.

znajduje się w najmniejszej odległości od wektora wejściowego **x**). Wyznaczony w ten sposób neuron zostaje nazwany *neuronem zwycięskim*, gdyż on właśnie wygrał współzawodnictwo pomiędzy neuronami warstwy wyjściowej. Jak z tego wynika rywalizacja pomiędzy neuronami sieci Kohonena polega na wyborze takiego neuronu, którego wektor wag jest w największym stopniu zbliżony do wprowadzonego wektora wartości wejściowych.

Po zidentyfikowaniu neuronu zwycięskiego modyfikowane są jego wagi (co jest zasadniczym etapem każdego procesu uczenia). Modyfikacja wag przeprowadzana jest tak, aby neuron zwycięski jeszcze bardziej przybliżył się do wprowadzonego aktualnie wektora wejściowego. Stosowana formuła jest bardzo prosta. Jeśli przyjmiemy, że wektor wag neuronu zwycięskiego jest równy $\mathbf{w}_{zw}$, zaś wektor wejściowy oznaczymy przez **x**, to wyznaczyć można wektor **r** stanowiący różnicę pomiędzy interesującymi nas wektorami:

$$\mathbf{r} = \mathbf{x} - \mathbf{w}_{zw}. \tag{12}$$

Jeśli obliczony w ten sposób wektor **r** zostałby *w całości* dodany do aktualnego wektora wag neuronu zwycięskiego ($\mathbf{w}_{zw}$), to zostałaby uzyskana równość wektora wejściowego **x** i nowego wektora wag ($\mathbf{w}_{zw} + \mathbf{r}$). Algorytm Kohonena zakłada jednak mniej drastyczny sposób modyfikacji wag. Jest ona realizowana według formuły:

$$\mathbf{w}_{zw}^{(t+1)} = \mathbf{w}_{zw}^{(t)} + \beta \mathbf{r}. \tag{13}$$

gdzie β jest *współczynnikiem uczenia* przyjmującym wartości z przedziału (0; 1)

Zgodnie z przedstawioną formułą w kroku *t* algorytmu uczenia wagi neuronu zwycięskiego modyfikowane są nie o cały wektor różnic (co maiłoby miejsce, gdyby współczynnik β przyjmował wartość równą 1), lecz o pewną część wektora **r.** Wielkość poprawki wag wyznacza iloczyn współczynnika uczenia β i wektora **r**. Im większa wartość współczynnika uczenia β zostanie zastosowana, tym w większym stopniu nowe wagi neuronu zwycięskiego przybliżą się do prezentowanego przypadku uczącego.

Przeprowadzając uczenie sieci Kohonena stosuje się najczęściej *zmienną* wartość współczynnika uczenia β. W początkowych opokach jego wartość ustala się na stosunkowo wysokim poziomie (np. β = 0,8). Powoduje to, że zmiany wag są duże i poszczególne neurony warstwy wyjściowej dość szybko upodabniają się do kolejnych przypadków uczących. W czasie dalszego treningu wartość współczynnika uczenia jest jednak stopniowo (na przykład w liniowy) sposób zmniejszana do bardzo małej wartości (np. β = 0,001). Zmiany w wartościach wag mają w związku z tym coraz mniejszym stopniu skokowy charakter i sieć w dalszych etapach procesu uczenia może w płynny sposób, precyzyjnie dostosować się do prezentowanych przypadków.

Istotną cechą algorytmu uczenia stosowanych w sieci Kohonena jest przeprowadzanie modyfikacji wag nie tylko dla neuronu zwycięskiego, lecz również dla jego sąsiadów. Liczba neuronów modyfikowanych w czasie uczenia jest określona przez obowiązujący **w danej chwili** promień sąsiedztwa. W początkowej fazie uczenia promień sąsiedztwa jest tak ustalany, aby niezależnie od lokalizacji neuronu zwycięskiego, jego sąsiedztwo obejmowało wszystkie neurony warstwy wyjściowej sieci (co powoduje, że modyfikowane są zawsze wszystkie neurony). W czasie kontynuacji procesu uczenia promień sąsiedztwa stopniowo się zmniejsza – aż do wartości zerowej (wówczas, jak pamiętamy, modyfikacji podlega tylko sam neuron zwycięzca).

Omówienia wymaga jeszcze sposób uczenia neuronów - sąsiadów. Stosowane są w tym zakresie dwa podejścia:

a) neurony należące do aktualnie obowiązującego sąsiedztwa uczone są w ten sam sposób, co neuron zwycięski. Wówczas do bieżących wartości wag każdego z nich dodawany jest wektor poprawek wyznaczony za pomocą formuły 13.
b) przy uczeniu sąsiadów zachowany jest **kierunek zmian** wyznaczony dla neuronu zwycięskiego, natomiast - wraz z oddalaniem się sąsiada od neuronu zwycięskiego - maleje **siła** wprowadzonych zmian. Stosując tę zasadę sprawiamy, że uczenie neuronu zwycięskiego realizowane jest zgodnie ze wzorem 13, zaś wraz z oddalaniem się od zwycięzcy wektor poprawek przemnażany jest przez **zmniejszający** się stopniowo *współczynnik sąsiedztwa*. Współczynnik ten ustawia się w taki sposób, że dla neuronu zwycięskiego przyjmuje on wartość 1, a dla sąsiadów ma odpowiednio mniejsze wartości i maleje do zera wraz z oddalaniem sąsiadów od neuronu zwycięskiego.

Niezależnie od zastosowanych szczegółów implementacyjnych algorytmu Kohonena prowadzi on **dopasowania** się siatki utworzonej przez warstwę wyjściową sieci do obiektów uczących. Oznacza to, że:

- poszczególne neurony dopasowują się do wzorców uczących - prezentacji każdego obiektu towarzyszy uaktywnienie **jednego** neuronu, posiadającego wektor wag najbardziej zbliżony do wartości zmiennych charakteryzujących obiekt. Często się zdarza, że ten sam neuron jest zwycięzcą dla wielu różnych wektorów wejściowych. Oznacza to, że charakteryzowane przez nie obiekty są podobne, a wektor wag odpowiadającego im neuronu reprezentuje całą tę grupę. Kolejne pokazy w trakcie uczenia i kolejne korekty prowadzą do tego, że wagi tego neurony stają się uśrednionymi wartościami cech charakteryzujących wszystkie obiekty, reprezentowane przez dany neuron). Sieć Kohonena dzieli więc obiekty zbioru uczącego na klasy (grupy, skupienia, klastry), które reprezentowane są przez poszczególne neurony wyjściowe.

  W sposób bezpośredni badacz nie może zadać liczby klas, na które podzielone zostaną obiekty zawarte w eksplorowanych danych. Może to jednak zrobić w sposób pośredni - określając strukturę warstwy wyjściowej sieci. Im więcej neuronów będzie wchodzić w jej skład, tym większe będzie rozdrobnienie uzyskanych grup. Należy jednak pamiętać, że nie zawsze liczba grup będzie równa liczbie neuronów, gdyż w następstwie samouczenia sieci nie dostajemy nigdy tak dobrych wyników grupowania, jak w przypadku zastosowania uczenia nadzorowanego (uczenia z nauczycielem). Dlatego w sieci Kohonena często pewna liczba neuronów pozostaje niewykorzystana - nie stają się one neuronami zwycięskimi dla żadnego przypadku uczącego – chociaż mogą się one uaktywnić dla niektórych obiektów wchodzących w skład zbioru testowego;
- sieć Kohonena zachowuje informację o *zależnościach* pomiędzy zidentyfikowanymi grupami obiektów. Zależności te głównie odwołują się do pojęcia *bliskości* (w sensie metryki Euklidesa) odpowiednich wektorów danych, a tę bliskość z kolei możemy kojarzyć z pojęciem podobieństwa opisywanych przez te dane obiektów. Oznacza to, że wraz ze wzrostem podobieństwa pomiędzy wyznaczonymi grupami obiektów, przybliżają się do siebie również (zgodnie z zasadą sąsiedztwa) reprezentujące je neurony. W efekcie sąsiadujące ze sobą neurony odpowiadają podobnym do siebie grupom danych, zaś neurony oddalone od siebie odpowiadają zróżnicowanym grupom.

  Z uwagi na opisaną zdolność sieci do reprezentowania informacji o strukturze zbioru obiektów jej warstwa wyjściowa nazywana jest czasem **mapą topograficzną** lub **mapą topologiczną**. Ta, charakterystyczna dla sieci Kohonena, własność nabywana jest całkowicie automatycznie dzięki wykorzystaniu mechanizmu sąsiedztwa w trakcie treningu sieci. Dla zaakcentowania unikatowości i znaczenia tej cechy, sieci Kohonena bywają nazywane niekiedy nazywane *samoorganizującymi się odwzorowaniami* (w skrócie SOM od *self-organizing maps*), przy czym zwolennicy tej nazwy często

podkreślają, że samoorganizacja jest znacząco bogatszą formą eksploracji danych, niż zwykłe grupowanie.

Utworzona w trakcie samouczenia sieć Kohonena może posłużyć także do zbadania prezentowanego w trakcie uczenia zbioru danych i do ujawnienia takich jego cech, o których istnieniu twórca sieci mógł wcale nie wiedzieć. Nauczone sieć stanowi również pewien model przechowujący wiedzę o strukturze zbiorowości. Może on być wykorzystany dla klasyfikacji i porządkowania także innych danych niż te, które prezentowane były w czasie uczenia. Nauczona sieć Kohonena ma bowiem tę właściwość, że po wprowadzeniu na jej wejścia dowolnych danych (byle tylko tego samego typu, jak wartości podawane na wejścia sieci podczas uczenia), zawsze uaktywni się jakiś neuron wskazujący na klasę, do której *najprawdopodobniej* powinien zostać przypisany nowy obiekt.

Uogólniając powyższe rozważania należy stwierdzić, że sieć Kohonena dokonuje grupowania obiektów, opisuje zależności pomiędzy tymi grupami oraz pozwala na zaklasyfikowanie nowych obiektów do wcześniej rozpoznanych klas.

3.8. Zastosowania sieci neuronowych

Możliwości zastosowań sieci neuronowych w dziedzinie ekonomii są bardzo szerokie. O podejmowanych próbach i uzyskanych rezultatach bardzo dokładnie informują źródła literaturowe wskazane w bibliografii zestawionej na końcu skryptu. Zrelacjonowanie wszystkich dokonań związanych z zastosowaniem sieci neuronowych w eksploracyjnej analizie danych ekonomicznych nie jest możliwe, więc na kilku najbliższych stronach przedstawimy tylko najważniejsze informacje dotyczące typowych sposobów ich wykorzystania. Wskażemy także na najistotniejsze pozycje literaturowe pozwalające na dalsze, samodzielne studiowanie tego zagadnienia.

Obszar zastosowań sieci neuronowych rozciąga się na różnego typu problemy – obejmując między innymi opis zależności (modelowanie), klasyfikację wzorcową i bezwzorcową czy też analizę szeregów czasowych. W każdym wymienionym przypadku inny będzie sposób budowy i oceny modelu neuronowego, a także nieco odmienny będzie sposób przygotowania danych i interpretacji wyników, co postaramy się przedstawić w kolejnych podpunktach.

3.8.1. Sieci neuronowe jako narzędzie opisu zależności

Stosując sieci neuronowe w charakterze narzędzia opisu zależności między danymi chcemy, aby po wprowadzeniu na wejścia sieci wartości zmiennych objaśniających, na wyjściu sieci pojawiła się odpowiadająca im wartość zmiennej objaśnianej.

Stosowanie sieci neuronowych do opisu zależności znajduje uzasadnienie na gruncie teorii i praktyki. Teoria mówi nam, że odpowiednio skonstruowana sieć neuronowa może aproksymować dowolną zależność nieliniową (por.: [Kołmogorow, 1957], [Hecht - Nielsen, 1987], [Cybenko, 1989]). Praktyka potwierdza przydatność modeli neuronowych do opisu zależności istniejących w rzeczywistych danych gospodarczych i wskazuje na ich przydatność na etapie prognozowania czy też wspomagania procesu decyzyjnego.

Proces budowy modelu neuronowego to nie tylko określenie jego struktury i przeprowadzenie uczenia sieci neuronowej. Stosowana procedura składa się zwykle z większej liczby etapów, do których zalicza się:

- wstępną analizę danych,

- zaprojektowanie struktury i uczenie sieci neuronowej,
- ocena modelu,
- zastosowanie modelu w praktyce.

Punktem wyjścia do **wstępnej analizy danych** (określanej również w literaturze jako *preprocessing*) jest odpowiednie przygotowanie ciągów danych rzeczywistych, będących podstawą do szacowania parametrów modelu sieciowego. Wstępna analiza danych obejmuje wszystkie te operacje, które wykonywane są na posiadanym zbiorze danych przed ich wprowadzeniem na wejścia sieci. Zakładamy, że zbiór posiadanych danych ma postać macierzy, w której liczba wierszy jest równa liczbie obserwacji, zaś liczba kolumn jest sumą liczby zmiennych objaśnianych i zmiennych objaśniających. Podstawowe etapy wstępnej analizy danych to:

a) *sprawdzanie poprawności danych pierwotnych.* – Etap ten obejmuje między innymi odpowiednie potraktowanie informacji brakujących i nietypowych (danych błędnych i danych prawidłowych, ale stanowiących zapis zaobserwowanych anomalii) (por.: [Pawełek i in., 1996], [Jajuga, 1993]). W przypadku wystąpienia braków lub nietypowych wartości należy podjąć decyzję dotyczącą sposobu ich potraktowania (pominięcia ich w trakcie dalszych prac lub sposobu ich wykorzystania);

b) *wybór zmiennych wejściowych (objaśniających).* – Przy wyborze tych danych (traktowanych potem jako sygnały wejściowe do sieci) należy podejmować decyzje z dużą uwagą i z poczuciem odpowiedzialności. Jest sprawą oczywistą, że pominięcie istotnej informacji wejściowej może powodować, że wartości generowane na wyjściu modelu będą błędne. Należy jednak również pamiętać, że dostarczenie na wejścia modelu zbyt dużej liczby informacji może także negatywnie wpływać na prawidłowość uzyskiwanych wyników, gdyż wprowadzanie wartości większej liczby zmiennych wymaga zastosowania większej liczby neuronów wejściowych, co z kolei przyczynia się do zwiększenia liczby połączeń i odpowiadających im współczynników wagowych. Nieostrożne zwiększenie liczby wag ustawialnych w sieci (których wartości określane są w toku uczenia) zwiększa wymagania pamięciowe, podnosi złożoność obliczeniową zarówno procesu uczenia, jak i procesu eksploatacji sieci, a także (co jest już najgorsze) może być bezpośrednią przyczyną zaniku w sieci zdolności do generalizacji wyników uczenia.

Prowadząc badania dysponujemy zwykle stosunkowo dużą liczbą potencjalnych zmiennych wejściowych, których zbiór należy we właściwy sposób zredukować, żeby uniknąć wskazanych wyżej kłopotów. Metody służące redukcji zbioru zmiennych wejściowych podzielić można na dwie zasadnicze grupy: pierwsza z nich obejmuje te metody, które polegają na eliminacji niepotrzebnych zmiennych, zaś druga grupa skupia algorytmy pozwalające na zastąpienie pierwotnego zbioru zmiennych nowym zbiorem zmiennych „sztucznych", składającym się z mniejszej liczby elementów. Czytelnikom zainteresowanym odpowiednimi algorytmami redukcji liczby danych wejściowych polecamy pracę [Grabiński, 1992].

W procesie redukcji zbioru zmiennych wejściowych wykorzystywane są niekiedy algorytmy genetyczne, służące w tym przypadku wyłącznie poszukiwaniu optymalnego zestawu zmiennych objaśniających. Informacja o uwzględnionych oraz nie uwzględnionych zmiennych wejściowych kodowana jest wtedy w postaci chromosomów, składających się na ewoluującą populację. Różne chromosomy mają w sobie zakodowane różne zestawy uwzględnianych i nie uwzględnianych danych wejściowych. W praktyce realizuje się to w taki sposób, że poszczególne składniki łańcucha binarnego, jakim jest w istocie chromosom, związane są z poszczególnymi sygnałami wejściowymi, czyli danymi objaśniającymi, które potencjalnie mogą być podane do sieci. Konkretny chromosom jest

tak zbudowany, że w różnych miejscach ma wartości **1**, oznaczające, że określona zmienna ma być wprowadzona do sieci, a w innych miejscach ma wartości **0**, co jest wskaźnikiem, że z odpowiednich danych rezygnujemy przy budowie sieci. Każdy chromosom opisuje więc inną strukturę sieci, a kryterium oceny poszczególnych „osobników" w populacji uzależnione są wprost od jakości (dokładności)opisywanych przez nie modeli neuronowych. Ewolucja populacji, wyłaniająca chromosomy cechujące się najlepszą funkcją przystosowania prowadzi więc prostą drogą do budowy sieci neuronowej o takim zestawie sygnałów wejściowych, który gwarantuje najlepszą jakość budowanego modelu neuronowego.

Przydatność takiej metody poszukiwania optymalnego zbioru zmiennych wejściowych potwierdzona została w licznych pracach badawczych. O genetycznej optymalizacji zbioru zmiennych wejściowych piszą między innymi: [Hertz i in., 1993], [Braun i in., 1995], [Rymarczyk, 1997];

c) *określenie sposobu prezentacji informacji o charakterze jakościowym.* Ten etap budowy modelu neuronowego wystąpi wtedy, gdy wśród zmiennych objaśniających występują zmienne o charakterze jakościowym. Ich wartości przed wprowadzeniem na wejścia sieci neuronowej muszą zostać przekształcone do postaci numerycznej, gdyż tylko takie sygnały mogą być wprowadzane na wejścia neuronów. Najprostszy sposób przekształcenia wartości jakościowych do postaci numerycznej akceptowanej na wejściach sieci polega na przyporządkowaniu występującym w zbiorze danych rozróżnialnym wartościom jakościowym – ustalonych w pewien sposób wartości numerycznych (np. kolejnych liczb całkowitych), które wprowadzane będą na wejścia sieci w miejsce wartości pierwotnych.

Taka prosta operacja posiada jednak zasadniczą wadę: jeśli pierwotne wartości zmiennej wyrażone zostały na skali **nominalnej**, to ich proste zastąpienie wartościami liczbowymi wprowadza w sposób sztuczny uporządkowanie tych wartości – które nie jest uzasadnione ich naturą. Chcąc uniknąć opisanej niedogodności stosuje się często inny sposób reprezentacji wartości jakościowych. Do wprowadzenia wartości **jednej** zmiennej wejściowej stosuje się wtedy tyle neuronów wejściowych, ile różnych wartości ta zmienna może przyjąć (jeden neuron wejściowy odpowiada jednej ustalonej **wartości** zmiennej wejściowej). Wprowadzenie konkretnej wartości jakościowej na wejście takiej sieci realizowane jest poprzez wprowadzenie wartości równej jedności na neuron odpowiadający tej wartości, jaką aktualnie ma zmienna jakościowa i wprowadzenie zer na wszystkie pozostałe neurony, odpowiadające innym wartościom rozpatrywanej zmiennej jakościowej.

Ten sposób prezentacji wartości jakościowych jest wygodny, gdyż nie stwarza problemów związanych ze sztucznym wprowadzaniem uporządkowania wartości nominalnych, ale również nie jest wolny od wad. W tym przypadku najistotniejszym problemem może być zbytnia rozbudowa warstwy wejściowej sieci, co było już wyżej wskazywane jako wada sieci (taka nadmierna liczba neuronów wejściowych może być przyczyną zmniejszenia zdolności sieci do generalizacji). Problemy związane z zapewnieniem właściwej reprezentacji danych jakościowych omówione zostały dokładnie między innymi w godnej polecenia pracy [Masters, 1996]. Autor rozpatruje w niej oddzielnie problematykę reprezentacji wartości wyrażonych na skali nominalnej i oddzielnie dyskutuje kwestie wartości wyrażonych na skali porządkowej. Do tej publikacji odsyłamy zainteresowanych Czytelników;

d) *operacjonalizacja danych.* – Czynność ta polega na przekształceniu danych za pomocą formuł algebraicznych, dokonywanym w celu uzyskania wartości eksponujących ważne, z punktu widzenia przeprowadzanych badań, cechy badanego fragmentu rzeczywistości. Problematyka operacjonalizacji danych została omówiona między innymi w pracach

[Grabiński i in., 1990] oraz [Masters, 1996], do tych pozycji powinien więc sięgnąć Czytelnik potrzebujący bardziej szczegółowych informacji, niż te, które podano w niniejszym skrypcie. Potrzeba przekształcenia pierwotnych wartości zmiennych może również wynikać z konieczności dostosowania danych do specyfiki stosowanych modeli neuronowych - wartości zmiennych objaśnianych muszą zostać przekształcone tak, aby zakres uzyskanych wartości był zgodny z zakresem wartości uzyskiwanych na wyjściu sieci (zakres tych ostatnich jest uzależniony od rodzaju funkcji aktywacji neuronu), natomiast wartości zmiennych wejściowych przekształcane są do przedziału, któremu odpowiada stosunkowo duża zmienność w wartościach wyjściowych neuronu. Przedstawione problemy rozwiązywane są zwykle poprzez skalowanie, normalizację lub standaryzację danych (por.: [Pawełek i in., 1995], [Masters, 1996], [Lula, 1999]).

Po przeprowadzeniu wstępnej analizy danych rozpocząć można drugi etap prac związanych z budową modelu neuronowego. Etapem tym jest **zaprojektowanie struktury i uczenie sieci neuronowej**.

Struktura sieci określa jej możliwości - wraz z jej rozbudową rośnie zwykle posiadana przez model zdolność do aproksymacji (popularnie mówi się, że bardziej rozbudowane sieci neuronowe są „inteligentniejsze"), jednakże zastosowanie sieci o zbyt mocno rozbudowanej strukturze wpływa na wydłużenie procesu uczenia oraz (w przypadku posiadania ograniczonego zasobu danych uczących) może prowadzić do zmniejszenie zdolności sieci do uogólniania wyników procesu uczenia.

Określenie struktury warstwy wejściowej oraz warstwy wyjściowej sieci jest stosunkowo proste, gdyż liczba występujących w nich neuronów jest wprost uzależniona od liczby zmiennych i od przyjętego sposobu reprezentacji zmiennych. Problemem może być natomiast określenie liczby warstw i neuronów ukrytych. Stosowane w tym zakresie techniki uzależnione są od rodzaju wybranej sieci neuronowej. My (w tym podrozdziale) naszą uwagę skupimy na scharakteryzowanych w poprzednim rozdziale sieciach perceptronowych. Stosując ten rodzaj sieci wykorzystywać można różne techniki określania struktury i wielkości jej części ukrytej, do najważniejszych z nich należy zaliczyć:

- *metody heurystyczne* - mające zwykle postać prostych reguł „zdroworozsądkowych", pozwalających na określenie struktury sieci w zależności od cech charakteryzujących rozpatrywany problem. Najczęściej spotykaną zasadą tego typu jest reguła głosząca, że w sieciach stosowanych do opisu zależności należy stosować (jeśli to tylko możliwe) jedną warstwę ukrytą z liczbą neuronów równą średniej arytmetycznej z wartości określających liczbę neuronów wejściowych i wyjściowych. Tak sformułowana prawidłowość niestety nie uwzględnia rzeczywistego charakteru zależności występujących pomiędzy zbiorem zmiennych wejściowych, a zmienną wyjściową, zatem dla jednych (łatwych) problemów proponowana liczba neuronów może być za duża, podczas gdy w przypadku naprawdę trudnego zadania liczba neuronów „ukrytych" może być za mała.

  Inną dość często stosowaną regułą jest zasada głosząca, że struktura sieci powinna zostać dobrana w taki sposób, aby liczba jej parametrów (wag i wartości progowych wszystkich neuronów tworzących sieć) była przynajmniej kilka razy mniejsza od liczby danych uczących. Stosowanie przedstawionej zasady gwarantuje, że ilość informacji, jakie sieć musi zgromadzić (w postaci wartości swoich parametrów), nie będzie przekraczała ilości informacji dostarczanych w danych uczących, dzięki czemu zmniejsza się prawdopodobieństwo zaniku zdolności do generalizacji (przeuczenia sieci). Przedstawione powyżej reguły są bardzo pomocne, ale nie zawsze wystarczająco skutecznie wskazują na najlepszą strukturę sieci, dlatego stosuje się także dalej omówione techniki iteracyjnej optymalizacji struktury sieci;

- *techniki polegające na redukcji początkowej struktury sieci* – są jednymi z wielu technik iteracyjnej optymalizacji struktury sieci. Punktem wyjścia jest w nich sieć o początkowo bardzo rozbudowanej strukturze (zwykle stosuje się jako punkt wyjścia sieć o jednej warstwie ukrytej posiadającej dużą liczbę neuronów); w trakcie uczenia zbędne połączenia i/lub neurony są usuwane z sieci. Do podstawowych metod redukcji należy zaliczyć *metody wykorzystujące tzw. człon kary* (opisane w: [Hertz i in., 1993], [Osowski, 1996]) oraz *metody wrażliwościowe* (np.: metody *Optimal Brain Damage* lub *Optimal Brain Surgeon* opisane w [Hassibi i in., 1993], [Osowski, 1996]). Wspomniany powyżej „człon kary" to wyrażenie, którego wartość dodawana jest do minimalizowanej w trakcie uczenia funkcji błędu. Wartość członu kary jest proporcjonalna do liczby elementów składowych sieci, a więc sieć „karana" jest za zbyt rozbudowaną strukturę i może polepszyć swoje wskaźniki jakości działania zarówno zmniejszając błąd rozwiązywania zadań uczących, jak i redukując swoją strukturę. Idea metod wrażliwościowych jest inna: w trakcie uczenia wykrywane i usuwane są te elementy sieci (pojedyncze połączenia lub całe neurony), które w najmniejszym stopniu wpływają na wartość funkcji błędu;

- *techniki polegające na rozbudowie początkowej struktury sieci* - są innymi technikami iteracyjnej optymalizacji struktury sieci. Punktem wyjścia jest w nich sieć o początkowo bardzo prostej strukturze (bez warstw ukrytych) - natomiast w trakcie działania algorytmu uczącego dodawane są niezbędne neurony i/lub połączenia. Propozycje tego typu algorytmów przedstawione zostały w pracy [Refenes, 1995]. Prawdopodobnie najbardziej znanym algorytmem rozbudowującym jest algorytm Fahlmana ([Fahlman i in., 1990], [Osowski, 1996]), którego stosowanie nie prowadzi jednak do utworzenia sieci o strukturze charakterystycznej dla sieci perceptronowych, lecz o strukturze kaskadowej;

- *techniki przeszukujące* – są to techniki bazujące na zestawie kilku struktur wejściowych, w oparciu o które przy pomocy algorytmu genetycznego tworzone są i testowane różne warianty sieci w celu określenia struktury najlepiej przystosowanej do realizacji stawianych wymagań. Najważniejszym etapem tej grupy metod jest zaprojektowanie sposobu opisu struktury sieci w postaci chromosomu ([Braun i in., 1995], [Balakrishan i in., 1995]);

Uczenie sieci neuronowej polega na określeniu prawidłowych wartości jej parametrów. Najlepiej znana metoda uczenia, jaka jest *metoda wstecznej propagacji błędów* została scharakteryzowana w początkowej części bieżącego rozdziału. W praktyce jest ona najczęściej stosowana, gdyż wymaga najmniejszej liczby założeń aby mogła prawidłowo działać, chociaż stale poszukuje się lepszych metod, z uwagi na długi czas uczenia sieci przy stosowaniu metody *wstecznej propagacji błędów*. W większości przypadków uczenie sieci można przeprowadzić w znacznie krótszym czasie stosując o wiele bardziej zaawansowane, ale mniej popularne, algorytmy. Do najbardziej znanych takich „alternatywnych" algorytmów należą: metoda gradientów sprzężonych ([Billings, 1995]), metoda Marquardta-Levenberga ([Hagan i in., 1994]) oraz metody zmiennej metryki ([Setiono i in., 1995]). We wskazanych źródłach literaturowych oraz w podręcznikach: [Kręglewski i in., 1984] oraz [Osowski, 1996] należy szukać szczegółów dotyczących sposobu działania wymienionych (oraz innych) algorytmów uczenia, najczęściej jednak użytkownik sieci nie musi mieć wiedzy teoretycznej o zasadach działania używanych przez siebie algorytmów ucznia, gdyż nagminnie stosowane programy wspomagające budowę i eksploatację sieci neuronowych mają odpowiednie algorytmy uczenia wbudowane w swoją strukturę, zatem wybór metody uczenia jest w istocie wyborem jednej z pozycji menu – a do tego na ogół nie potrzeba wiadomości teoretycznych.

Podejmując trud budowy modelu neuronowego służącego do opisu zależności między określonymi danymi gospodarczymi należy również dokładnie przemyśleć zagadnienie jego **oceny**. Problem właściwej oceny sieci pojawia się już na etapie jej uczenia, w trakcie którego cyklicznie bada się jakość sieci, aby w porę wykryć przeuczenie. Na tym etapie budowy modelu sytuacja jest jednak w miarę prosta, ponieważ za podstawowy miernik jakości sieci przyjmuje się zwykle wartość funkcji błędu, zdefiniowanej najczęściej w postaci sumy kwadratów różnic pomiędzy wartościami wyznaczonymi przez sieć i wartościami poprawnymi (wzorcowymi), pobieranymi ze zbioru uczącego. W ten sposób jakość sieci daje się w każdej chwili procesu uczenia wyrazić liczbą określającą sumaryczną wartość popełnianego przez sieć błędu. Wartość ta może być jedna, jeśli skupimy uwagę tylko na zbiorze uczącym, albo można rozważać dwie oddzielne miary błędu – jedną dla danych uczących, a drugą dla danych walidacyjnych. Przy ocenie i interpretacji uzyskanych wartości miar błędu pomocna jest bardzo prezentacja graficzna (zwykle w postaci zależności wartości błędu od liczby pokazów wykorzystanych do tej pory w procesie uczenia sieci). Na podstawie uzyskanych wartości błędów oraz na podstawie ich zmienności (ocenianej wzrokowo lub w sposób formalny) można precyzyjnie dobrać moment, w którym proces uczenia trzeba przerwać, gdyż dalsze jego kontynuowanie grozi efektem przeuczenia.

O wiele bogatszy zestaw mierników stosowany jest do finalnej oceny skonstruowanego modelu. Na tym etapie stosowane są różnorodne mierniki, do których zalicza się między innymi:

- *sumę kwadratów reszt* – jest wspomniany już w poprzednim akapicie miernik wykorzystywany także do monitorowania procesu uczenia wraz z ustaleniem kryterium jego przerwania. Miernik ten jest chyba najpopularniejszym narzędziem oceny jakości sieci – także na etapie, kiedy jest ona całkowicie nauczona i gotowa do roboczej eksploatacji. Podstawową wadą tego miernika jest uzależnienie jego wartości od liczby elementów w zbiorze danych;

- *błąd średniokwadratowy* - liczony jako iloraz sumy kwadratów reszt i liczby elementów w uwzględnionym zbiorze danych (jest wolny od wskazanej wyżej wady);

- *pierwiastek błędu średniokwadratowego* – pozwala na łatwą interpretację jakości sieci, ponieważ wartość błędu wyrażana jest w tej samej skali, w jakiej podawane są rozwiązania (sygnały wyjściowe z sieci);

- *miary oparte na współczynniku korelacji liniowej* pomiędzy wartościami rzeczywistymi i wartościami wyznaczonymi za pomocą modelu.

Zarówno ocena sieci dokonywana w trakcie jej uczenia, jak i przeprowadzana po zakończeniu prac związanych z określaniem wartości jego parametrów przeprowadzana jest niezależnie dla zbioru *uczącego*, *walidacyjnego* i *testowego*. Poprawność interpretacji uzyskanych wartości uzależniona jest w dużym stopniu od poprawności podziału dostępnego zbioru danych na te trzy zbiory. W większości przypadków podział ten dokonywany jest w sposób losowy.

Osoby zainteresowane teoretycznymi aspektami modelowania neuronowego znajdą wiele cennych informacji w pracach: [White, 1989] i [Zapranis i in., 1999]. Dla osób zainteresowanych tworzeniem prognoz przedziałowych za pomocą modeli neuronowych niezbędna będzie pozycja [Husmeier, 1999].

3.8.2. Sieci neuronowe jako narzędzie klasyfikacji wzorcowej

Zasadniczym celem *klasyfikacji wzorcowej* (dyskryminacji, rozpoznawania obrazów) jest przypisanie badanego obiektu do jednej ze znanych klas. Zaklasyfikowanie obiektu dokonywane jest na podstawie wartości opisujących go zmiennych. Jednym z narzędzi pozwalających na przeprowadzenie klasyfikacji wzorcowej obiektów są sieci neuronowe. Skonstruowanie właściwie działającego modelu neuronowego, służącego jako klasyfikator, składa się z podobnych etapów jak w przypadku modeli wykorzystywanych do opisu zależności, jednakże w sposobie realizacji kolejnych etapów budowy modelu można dojrzeć pewne różnice.

Pierwsza, zasadnicza różnica związana jest ze sposobem interpretacji wartości wyjściowej modelu. Modele klasyfikacyjne powinny dostarczać informacji o klasie, do której należy zakwalifikować obiekt charakteryzowany przez wartości zmiennych wejściowych. Klasy te identyfikowane są najczęściej przez przypisane im nazwy (etykiety), które **nie** mogą być bezpośrednio generowane przez sieć. Z tego względu opisane w poprzednim punkcie elementy realizowane w ramach wstępnego przetwarzania danych, uzupełnić należy w przypadku zadań klasyfikacji wzorcowej o dodatkowy element: określenie sposobu **reprezentacji** informacji o przynależności obiektu do określonej klasy przez wartości, jakie mogą być generowane przez neuron (neurony) wyjściowe sieci. Inaczej mówiąc, należy określić, w jaki sposób wartości numeryczne uzyskane na wyjściu sieci przekształcane będą w informację identyfikującą wskazywaną (jako odpowiedź) klasę, do której sieć zalicza rozważany obiekt.

Przedstawiony problem można w stosunkowo prosty sposób rozwiązać w przypadku występowania tylko **dwóch** klas. Każdej z nich można wtedy przyporządkować jedną ze skrajnych wartości mogących pojawić się na wyjściu neuronu (zależnie od przyjętej charakterystyki neuronu mogą to być wartości **0** i **1** albo **–1** i **1**). Tak określone wartości kodujące odpowiednie klasy wykorzystywane będą w czasie treningu sieci i one także powinny pojawiać się na jej wyjściu w trakcie eksploatacji już wytrenowanej sieci. Na przykład, jeżeli wykorzystywana jest sieć posiadająca neuron wyjściowy wyposażony w sigmoidalną funkcję aktywacji, to uzyskanie wartości równej **jeden** może być interpretowane jako decyzja o zaliczeniu obiektu do pierwszej klasy, zaś uzyskanie wartości **zerowej** może świadczyć o przynależności obiektu do drugiej klasy.

Niestety, przedstawiony sposób interpretacji wartości wyjściowych nie jest do końca przydatny w praktyce, gdyż dla nieznanych obiektów (w trakcie eksploatacji sieci) bardzo rzadko uda się nam uzyskać na wyjściu sieci takie skrajne wartości funkcji aktywacji. Zwykle będziemy uzyskiwać wartości pośrednie, zawarte w przedziale określonym przez te dwie skrajne wartości. Interpretacja takich „nieostrych" wartości wyjściowych wymaga określenia wartości dwóch dodatkowych parametrów, którymi są dwie wartości progowe (rys. 31).

Rys. 31. Interpretacja wartości wyjściowej neuronu dla modeli realizujących zadania klasyfikacji wzorcowej w przypadku dwóch klas

Po zdefiniowaniu *górnej* i *dolnej wartości progowej* sposób interpretacji wartości wyjściowej neuronu jest następujący: uzyskanie wartości większej od górnej wartości progowej informuje o zaliczeniu obiektu do pierwszej klasy, uzyskanie wartości mniejszej od dolnej wartości progowej świadczy o przynależności obiektu do drugiej klasy, zaś uzyskanie wartości zawartej w przedziale określonym przez dolną i górną wartość progową można interpretować jako brak możliwości podjęcia decyzji o przynależności rozpatrywanego obiektu.

Wartość wyjściowa pojedynczego neuronu wyjściowego w sieci klasyfikującej dychotomicznie (to znaczy zaliczającej obiekt do jednej z **dwóch** tylko możliwych klas) może być również interpretowana jako *prawdopodobieństwo przynależności obiektu* do tej klasy, która reprezentowana jest na wyjściu przez wartość **jeden**. W takim przypadku prawdopodobieństwo przynależności obiektu do klasy alternatywnej (kodowanej na wyjściu sieci sygnałem wyjściowym neuronu mającym znamionową wartość **zero**) może być obliczane poprzez odejmowanie aktualnej wartości sygnału wyjściowego od wartości **jeden**.

W inny sposób należy postępować, budując neuronowy model realizujący zadanie klasyfikacji wzorcowej w przypadku, gdy liczba klas jest większa od dwóch. W takiej sytuacji stosuje się zwykle sieci, w których liczba neuronów w warstwie wyjściowej jest równa liczbie rozpoznawanych klas. W trakcie uczenia dąży się wtedy do określenia wartości parametrów sieci na poziomie zapewniającym uzyskanie jedynki na neuronie odpowiadającym klasie, jaka powinna być przypisana do obiektu opisanego aktualnym zestawem danych wejściowych i gwarantującym pojawienie się zer na wszystkich pozostałych neuronach. Podobnie jak w przypadku dwóch klas, wartości uzyskiwane w trakcie uruchomienia sieci dla danych roboczych (odmiennych od danych uczących) odbiegają od podanych wyżej wartości skrajnych i do ich interpretacji wykorzystuje się (zdefiniowane wtedy dla każdego neuronu wyjściowego) wartości progowe. O przynależności neuronu do danej klasy świadczy pojawienie się wartości większej od górnej wartości progowej na neuronie odpowiadającym danej klasie **i równocześnie** uzyskanie wartości mniejszych od dolnej wartości progowej na wszystkich pozostałych neuronach wyjściowych. Jeśli oba wymienione warunki nie są spełnione jednocześnie, to sieć nie jest w stanie podjąć decyzji o przynależności obiektu. Możliwe jest wtedy podanie rozwiązania cząstkowego, polegającego na wskazaniu kilku hipotetycznych klas („to jest kot albo pies, ale na pewno nie koń”), ale to już zależy od tego, do czego i w jaki sposób zamierzamy wykorzystać wyniki produkowane przez sieć.

W przypadku stosowania sigmoidalnej funkcji aktywacji we wszystkich neuronach wyjściowej warstwy sieci klasyfikacyjnej, wartości uzyskiwane na wyjściach sieci nie sumują się do jedności, co powoduje, że nie należy ich interpretować jako prawdopodobieństw przynależności do poszczególnych klas. Chcąc uzyskać taką możliwość należy otrzymane wartości poddać odpowiedniej procedurze normalizacyjnej (na przykład dzieląc każdą wartość wyjściową przez sumę wszystkich wartości uzyskanych na wyjściach). Formalna poprawność takiej interpretacji może jednak pozostawiać wiele do życzenia.

Stosowanie praktyczne neuronowych modeli klasyfikujących wymaga użycia właściwych metod ich oceny. Za najprostszy miernik jakości działania sieci klasyfikacyjnej należy uznać odsetek przypadków, w którym sieć dokonała poprawnej klasyfikacji. Możliwe są jednak bardziej wyrafinowane oceny, na przykład odwołujące się do faktu, że różne błędy klasyfikacji mogą być w różnym stopniu niebezpieczne. Tworząc odpowiednie macierze „cen błędów" można niekiedy uzyskać bardziej subtelne oceny jakości działania neuronowego klasyfikatora, daleko lepiej charakteryzujące jego użyteczność, niż same tylko „surowe" odsetki poprawnych i błędnych decyzji.

Modele realizujące zagadnienia klasyfikacji wzorcowej mogą być na szeroką skalę stosowane w dziedzinie ekonomii. W celu zilustrowania ich możliwości aplikacyjnych przytoczone zostaną ich przykładowe zastosowania (wraz z odnośnikami do dokładnych opisów, w których można dokładniej poznać i dyskutowany problem i zastosowaną metodę):

- tworzenie modeli służących ocenie zdolności kredytowej potencjalnych kredytobiorców lub osób ubiegających się o przydział karty kredytowej - na wejścia modelu wprowadzane są dane charakteryzujące klienta, zaś na wyjściu pojawia się informacja o sposobie jego zaklasyfikowania ([Rymarczyk, 1997]);
- podejmowanie decyzji inwestycyjnych ([Rymarczyk, 1997], [Zieliński, 2000]);
- określanie profilu klienta ([Zieliński, 2000]);
- prognozowanie bankructwa firmy ([Trippi i in., 1993], [Refenes, 1995]).

Analiza wskazanych przypadków będzie pomocna przy projektowania własnych systemów wspomagających procesy decyzyjne.

3.8.3. Sieci neuronowe jako narzędzie klasyfikacji bezwzorcowej

Metody klasyfikacji bezwzorcowej w zastosowaniach ekonomicznych służą zwykle badaniom struktury zbioru obiektów. Ich zastosowanie pozwala na identyfikację skupień (o ile one występują), przypisanie obiektów do wyodrębnionych grup oraz określenie zależności pomiędzy zidentyfikowanymi skupieniami. Do narzędzi pozwalających na rozwiązywanie tego typu problemów należy również zaliczyć sieci neuronowe samouczące się.

W trakcie rozwiązywania zadań z dziedziny klasyfikacji bezwzorcowej stosowane są różne sieci uczone w trybie bez nauczyciela. Sieciom takim w trakcie treningu dostarczane są wyłącznie informacje charakteryzujące badane obiekty, zaś nie jest udostępniana wiedza dotycząca klasyfikacji tych obiektów, ani żadne inne informacje dotyczące rzeczywistej struktury zbiorowości (jej odkrycie jest podstawowym celem przeprowadzanego procesu samouczenia). Zadania klasyfikacji bezwzorcowej mogą być realizowane przez różne rodzaje sieci, my jednak ograniczymy się tu wyłącznie do scharakteryzowanej w tym rozdziale sieci Kohonena.

Pierwszym etapem budowy modelu wykorzystującego sieć Kohonena jest (podobnie jak w przypadku innych typów sieci) wstępna analiza danych, obejmująca wszystkie operacje wykonywane na pierwotnym zbiorze danych przed ich przetworzeniem przez sieć. Ten etap

prac wymaga w szczególności sprawdzenia poprawności danych oraz podjęcia decyzji dotyczących sposobów ich reprezentacji czy ewentualnie zastosowanej do nich operacjonalizacji. Przygotowane w ten sposób dane wejściowe będą stanowić podstawę do procesu samouczenia sieci, która musi zbadać i wykryć struktury skupień występujących (ewentualnie) w analizowanych danych.

Posiadając wstępnie przetworzony zbiór danych należy podjąć decyzję dotyczącą struktury sieci neuronowej. Sieć Kohonena nie stwarza żadnych problemów w zakresie doboru liczby warstw, gdyż posiada zawsze jedynie dwie warstwy: wejściową i wyjściową. Ustalenie liczby neuronów w warstwie wejściowej nie jest trudne, gdyż jest ona ściśle określona przez liczbę zmiennych wejściowych i przyjęty sposób ich reprezentacji. Od decyzji badacza uzależniona jest natomiast liczba neuronów wyjściowych. Pamiętając, że znajdujące się w niej neurony będą służyły do reprezentacji skupień występujących w analizowanym zbiorze danych, należy ich liczbę ustalić na poziomie równym akceptowalnej przez nas maksymalnej liczbie wyróżnionych grup.

Ważny jest również sposób rozmieszczenia neuronów w warstwie wyjściowej. Chcąc przeprowadzić badanie **struktury** zbioru obiektów decydujemy się na rozmieszczenie w określony sposób neuronów - najczęściej na dwuwymiarowej płaszczyźnie (czasami stosowane są też interpretacje trójwymiarowe). Budując model służący **porządkowaniu** obiektów wybierzemy z kolei liniowy układ neuronów wyjściowych.

Procedura samouczenia sieci Kohonena ma na celu utworzenie tak zwanej mapy topologicznej, której analiza może być cennym źródłem informacji o strukturze badanego zbioru obiektów.

Przydatność sieci Kohonena w zastosowaniach ekonomicznych potwierdzają rezultaty licznych prac badawczych. Warto prześledzić publikowane doniesienia, by w ten sposób wzbogacić swoją wiedzę o możliwościach tego typu modeli i prawidłowym sposobie ich konstrukcji. Szereg interesujących przykładów zastosowań ekonomicznych sieci Kohonena przedstawiono w pracy [Zieliński, 2000]. Warto je w tym miejscu wymienić i zachęcić zainteresowanych Czytelników do sięgnięcia do wymienionego podręcznika (gdzie oprócz opisów można znaleźć wiele cennych informacji o źródłach literaturowych):

- *badanie struktury zbioru przedsiębiorstw*. Przeprowadzone badania dotyczyły grupy ponad 10000 przedsiębiorstw belgijskich, z których każde charakteryzowane było przez 13 wskaźników finansowych. W trakcie analizy zastosowano sieć posiadającą w warstwie wyjściowej 36 neuronów umieszczonych w węzłach prostokątnej siatki o wymiarach 6 na 6. Po zaklasyfikowaniu przedsiębiorstw do grup opracowano - na podstawie wartości cech obiektów należącego do każdego skupienia - precyzyjny ich profil. Uzyskane wyniki posłużyły ich autorom do stworzenia systemu wspomagającego podejmowanie decyzji leasingowych;

- *ocena sytuacji finansowych banków i predykcja możliwości ich bankructwa*. Zrealizowane i opisane badania dotyczyły banków hiszpańskich. Sieci neuronowej prezentowano wartości wskaźników finansowych dotyczących 66 banków. Utworzona w ten sposób mapa pozwoliła ma określenie profili poszczególnych grup banków, pozwoliła zidentyfikować zależności pomiędzy charakteryzującymi je wartościami wskaźników oraz umożliwiła identyfikację zależności pomiędzy sytuacją banków a faktem ich bankructwa;

- *analiza wzorców występujących w szeregach czasowych*. Zagadnienia analizy szeregów czasowych za pomocą narzędzi neuronowych będą omawiane w następnym punkcie, ale warto już w tym miejscu odnotować możliwości wykorzystania w tym celu sieci

Kohonena. Relacjonowane w pracy [Zieliński, 2000] badania polegały na utworzeniu sieci, na wejścia której wprowadzane były pochodzące z kolejnych okresów wartości analizowanego szeregu czasowego. Po zakończeniu treningu neurony znajdujące się w warstwie wyjściowej reprezentowały typowe formacje występujące w szeregu czasowym. Zastosowana w ten sposób sieć Kohonena może służyć obiektywizacji decyzji dotyczących rozpoznaniu charakteru rozpatrywanego fragmentu szeregu czasowego.

Dla osób chcących wzbogacić swoją wiedzę na temat sieci Kohonena, a przede wszystkim możliwości ich wykorzystania w ekonomii szczególnie cenną pozycją będzie z pewnością praca [Deboeck i in., 1998], której współautorem jest twórca rozważanej tu klasy sieci T. Kohonen. Autorzy przedstawili przykłady zastosowań sieci do rozwiązywania różnorodnych problemów mikro- i makroekonomicznych. W pracy tej można także znaleźć szereg informacji dotyczących stosowanych technik analizy, oprogramowania oraz praktycznych porad dotyczących sposobów przeprowadzania analiz.

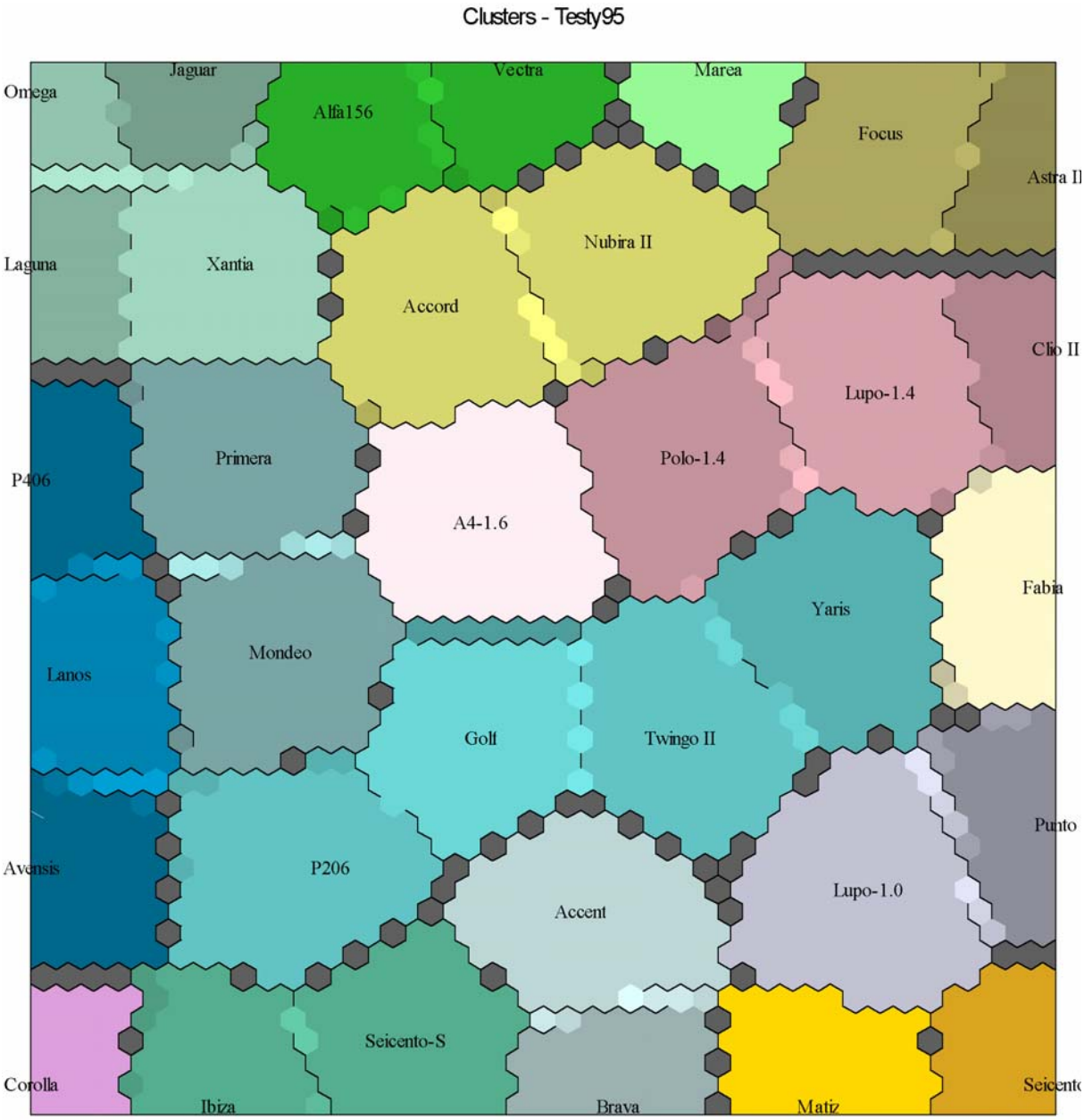
W celu zilustrowania procedury tworzenia sieci Kohonena i dla poglądowego pokazania możliwości zastosowań sieci Kohonena, przedstawimy przykład dotyczący analizy struktury zbioru wybranych samochodów osobowych. Dane wykorzystane w trakcie analizy przedstawia zamieszczona poniżej tabela (dane zaczerpnięto z publikacji: *Auto Świat - Katalog. Testy 2001*). W kolejnych kolumnach znajdują się następujące informacje:

- model pojazdu,
- pojemność silnika (dcm^3),
- moc (KM),
- przyspieszenie od 0 do 100 km/godz. (s),
- elastyczność od 60 do 100 km/godz. (s),
- prędkość maksymalną (km/godz.),
- droga hamowania od 100 do 0 km/godz. (m),
- poziom hałasu przy prędkości 50 km/godz. (dB),
- zużycie paliwa (l/100 km),
- etykieta będąca skróconą nazwą samochodu wyświetlaną na uzyskanej mapie.

Model	Pojemn. (dcm^3)	Moc (KM)	Przysp. 0-100 (s)	Elastycz 60-100 (s)	Pręd. max. (Km/h)	Droga hamow. (m)	Poziom hałasu (dB)	Zużycie paliwa (l/100 Km)	Etykieta
Alfa Romeo 156 2.5 24V	2.5	190	7.6	9	230	39.4	60	11.2	Alfa156
Audi A4 1.6	1.6	101	12.9	14.6	191	39.9	60	8.4	A4-1.6
Citroen Xantia 3.0 V6 Activa	3	190	8	8.4	230	37.7	59	11.8	Xantia
Daewoo Lanos 1.6 SX	1.6	106	10.9	10.3	180	42.9	60	9.3	Lanos
Daewoo Matiz 0.8	0.8	51	18.9	18.2	144	43.6	62	6.7	Matiz
Daweoo Nubira II 2.0 16V CDX	2	133	10.5	10	195	40	62	9.3	Nubira II
Fiat Brava 1.4 12V SX	1.4	80	14.9	15.1	170	44.2	64	8.5	Brava
Fiat Marea Weekend JTD 2.4	2.4	130	11.2	6.9	195	38.5	61	6.9	Marea
Fiat Punto 1.2 8V ELX	1.2	60	16.2	16.1	155	41.7	66	6	Punto
Fiat Seicento 900 SX	0.9	40	22.6	21.8	140	44	66	7.3	Seicento
Fiat Seicento Sporting 1.2	1.2	55	15.6	12.9	144	43.6	62	6.7	Seicento -S
Ford Focus 1.6 16V	1.6	100	11.3	12.5	180	37.8	60	8.1	Focus

Ambiente									
Ford Mondeo Turnier 1.6 16V	1.6	95	12.7	12.5	180	40.1	58	8.9	Mondeo
Honda Accord 1.8 iLS	1.8	136	10.6	11.1	205	38.7	61	8.3	Accord
Hyundai Accent GL 1.3	1.3	75	13.2	16.4	163	43.3	62	8.1	Accent
Jaguar S-Type 3.0 V6	3	238	8.4	9.7	235	39.4	60	12	Jaguar
Nissan Primera 2.0 CVT	2	140	11.2	10.2	202	40.2	58	8.7	Primera
Opel Astra II 1.6 16V	1.6	100	11.5	12.6	188	37.1	62	8.5	Astra II
Opel Omega 3.0 V6	3	211	8.2	9.7	240	38.3	60	12.4	Omega
Opel Vectra 2.5 V6 CDX	2.5	170	9.2	8.4	230	38.6	60	10.2	Vectra
Peugeot 206 1.4 XR Presence	1.4	75	13.2	13.1	170	41.7	60	7.3	P206
Peugeot 406 2.0 16V	2	135	10.1	9.6	208	42.4	59	10	P406
Renault Clio II 1.4 RN	1.4	75	12.8	11.9	170	39.8	62	6.4	Clio II
Renault Laguna 3.0 V6 24V Gr.	3	190	7.8	8.6	225	37.8	58	13.2	Laguna
Renault Twingo II 1.2	1.2	60	15.3	15.3	151	41.3	60	7.2	Twingo II
Seat Ibiza 1.4 Stella	1.4	60	15.9	11.7	157	44.5	61	6.9	Ibiza
Skoda Fabia 1.4 Comfort	1.4	68	15.3	14.7	160	39.9	63	7.7	Fabia
Toyota Avensis 1.6	1.6	110	12.1	14.2	190	43.4	58	8.9	Avensis
Toyota Corolla 1.6 Linea Terra	1.6	110	11.3	12.1	195	47.2	60	7.8	Corolla
Toyota Yaris 1.0	1	68	14.2	15.1	155	40.2	61	6.5	Yaris
Volkswagen Golf 1.4 16V	1.4	75	14.8	14.4	171	41.1	60	7.6	Golf
Volkswagen Lupo 1.0	1	50	17.9	14.3	151	42.8	64	5.7	Lupo-1.0
Volkswagen Lupo 1.4	1.4	75	11.9	11.9	172	39.8	63	7.7	Lupo-1.4
Volkswagen Polo 1.4 16 V	1.4	101	10.7	11.8	188	40.4	62	8	Polo-1.4

Do utworzenia mapy Kohonena wykorzystano program *Viscovery SOMine* firmy *Eudaptics Software* z Austrii (testową wersję oprogramowania, przykładowe zbiory danych oraz elektroniczną wersję podręcznika znaleźć można pod adresem http://www.eudaptics.co.at). Po przeprowadzeniu uczenia sieci uzyskano mapę przedstawioną na rys. 32.



Rys. 32. Mapa topologiczna reprezentująca analizowany zbiór samochodów.

Etykiety tekstowe wskazują na położenie poszczególnych modeli samochodów na mapie obrazującej ich właściwości. Zastosowany program pozwala na wyodrębnienie skupień znajdujących się na mapie (neurony zaliczane są do tego samego skupienia, jeśli odległość pomiędzy ich wektorami wagowymi jest mniejsza od ustalonej przez użytkownika wartości progowej; modyfikując poziom tego progu można uzyskać mniejszą bądź też większą liczbę skupień).

Korzystający ze wzmiankowanego programu użytkownik może użyć wielu narzędzi służących do analizy uzyskanej mapy. Jedną z możliwości jest analiza statystyczna ukształtowanych skupień. Na przykład można zażądać statystyki charakteryzującej skupienie obejmujące samochody: Peugeot 206 1.4 XR Presence, Volkswagen Golf 1.4 16V, Renault Twingo II 1.2 oraz Toyota Yaris. Wynik takiej analizy przedstawiony został na rys. 33.

Component	Mean	Standard de...	Minimum	Maximum	Sum
Pojemn.	1.253	0.168	1.000	1.502	5.000
Moc (KM)	69.9	6.6	59.7	88.1	278.0
Przys. 0 - 100 (s)	14.32	0.79	13.11	15.38	57.50
Elastycz 60-100	14.45	0.87	12.99	15.30	57.90
Prędmax	162.1	9.1	151.0	181.0	647.0
Drog. ham.	41.07	0.56	40.18	41.83	164.30
Poz. hałasu 50	60.25	0.43	59.92	61.13	241.00
Zużycie paliwa	7.16	0.41	6.47	8.02	28.60

Rys. 33. Statystyki charakteryzujące skupienie obejmujące wybrane samochody

Nie sposób przedstawić w tym miejscu wszystkich możliwości analizy danych za pomocą sieci Kohonena i programów pozwalających na budowę tego typu modeli. Ale zachęcamy Czytelników do studiowania literatury i zasobów internetowych związanych z hasłem sieci Kohonena, gdyż tą drogą można uzyskać bardzo dużo cennych i użytecznych informacji.

3.8.4. Sieci neuronowe w analizie szeregów czasowych

Do najbardziej znanych i najczęściej prezentowanych w literaturze ekonomicznych zastosowań sieci neuronowych należą problemy związane z analizą i prognozowaniem szeregów czasowych. Zasadniczym czynnikiem wpływającym na rozwój tego typu metod jest wzrastająca ciągle potrzeba doskonalenia metod analizy danych ekonomicznych oraz duże zapotrzebowanie w zakresie prognoz związanych z funkcjonowaniem rynków kapitałowych.

W podstawowych założeniach budowa modeli neuronowych służących analizie i prognozowaniu szeregów czasowych, jest zbliżona do procesu tworzenia neuronowych modeli regresyjnych. W analizie szeregów czasowych wykorzystywane są sieci uczone w trybie z nauczycielem, na wejścia których wprowadzane są dane charakteryzujące **bieżącą i przeszłe** wartości szeregu, zaś zadanie sieci polega na tym, że na jej wyjściu generowana jest przyszła wartość szeregu. Ten schemat funkcjonowania modelu jest często wzbogacany o możliwość wprowadzenia na wejścia sieci bieżącej i przeszłych wartości także innych zmiennych, o których mamy prawo sądzić, że mogą wpływać na przyszłe (prognozowane) wartości rozważanego szeregu. Należy jednak pamiętać, że mimo licznych podobieństw, pomiędzy procesem analizy danych o charakterze przekrojowym i danych mających postać szeregów czasowych - istnieją dość istotne różnice. Najważniejsza z nich dotyczą:

- etapu wstępnego przygotowania danych,
- realizacji uczenia oraz
- metod oceny neuronowych modeli szeregów czasowych.

Rozważając problematykę **wstępnego przygotowania danych,** mających postać szeregów czasowych należy zwrócić uwagę na następujące zagadnienia:

- *zapewnienie właściwego układu danych* - w przypadku budowy modeli regresyjnych wykorzystujących dane przekrojowe, na wejścia sieci wprowadzane były jednorazowo zmierzone *w tym samym czasie* wartości *różnych zmiennych* charakteryzujących ten sam obiekt. W trakcie analizy szeregów czasowych sytuacja jest odmienna: na kolejne wejścia sieci wprowadzane są wartości *tej samej zmiennej* (pojedynczej w przypadku szeregów jednowymiarowych, albo wektorowej w przypadkach wielowymiarowych), pochodzące z *różnych okresów czasu*. Chcąc wykorzystywać sieć do analizy szeregu czasowego należy więc z niego wydzielić (stosując technikę przesuwanego wzdłuż szeregu spójnego „okna”

danych) kolejne przypadki wykorzystywane do uczenia czy też do oceny sieci. Każdy z rozważanych przypadków składa się w tym zadaniu z zestawu wartości wejściowych (pochodzących z różnych wcześniejszych okresów wartości szeregu) oraz z wartości wyjściowej (to znaczy tej wartości szeregu, która następuje bezpośrednio po nich i która powinna zostać oszacowana na podstawie uwzględnionych wartości wejściowych). Większość programów umożliwiających konstruowanie modeli neuronowych dla analizy szeregów czasowych tego typu operację „okienkowania" danych wykonuje automatycznie, ale w niektórych przypadkach czynność ta musi zostać wykonana bezpośrednio przez użytkownika i dlatego trzeba znać jej sens i jej cel;

- *dekompozycja szeregu* - obserwowane w rzeczywistości szeregi czasowe, opisujące czasowe kształtowanie się zjawisk ekonomicznych, mają często bardzo skomplikowaną strukturę, powstałą poprzez nałożenie się procesów o różnym charakterze. Na przykład niektóre z tych składowych procesów mają postać trwałej tendencji (np. generalny wzrost na giełdzie), inne występując cyklicznie (w cyklach tygodniowych, miesięcznych, rocznych itp.,), a jeszcze inne pojawiają się nieregularnie lub nawet jednorazowo). Wyniki wielu badań wskazują na potrzebę wyodrębnienia z szeregu poszczególnych składowych i na celowość budowy dla każdej składowej jej niezależnego opisu (modelu cząstkowego).

  Procedura dekompozycji szeregu na modele częściowe może zostać zrealizowana na wiele sposobów, do których zaliczyć należy miedzy innymi:

  a) dopasowanie do danych funkcji regresji opisującej zasadniczy kierunek zmian i zastosowanie sieci neuronowej do opisu prawidłowości nie uwzględnionych /reprezentowanych przez reszty/,

  b) filtrację danych, na przykład poprzez różnicowanie /w celu wyeliminowania trendu/ lub uśrednianie średnią ruchomą /w celu usunięcia zmian o krótkim okresie trwania/,

  c) analizę spektralną przeprowadzoną poprzez zastosowanie transformacji Fouriera lub falkowej;

- *dobór informacji wejściowych modelu* - podobnie jak przy konstruowaniu modeli regresyjnych, również w przypadku budowy neuronowych modeli szeregów czasowych należy bardzo wnikliwie rozpatrzyć problem właściwego doboru informacji wprowadzanych na wejścia sieci. Jest oczywiste, że dla osiągnięcia sukcesu (w postaci udanej prognozy) należy dostarczyć sieci wszystkie te informacje, które są niezbędne do oszacowania przyszłej wartości szeregu. Jednocześnie, podobnie jak to dyskutowaliśmy w kontekście modeli regresyjnych, należy zadbać o to, aby nie wprowadzać do sieci informacji w niewielkim stopniu przydatnych, gdyż związane jest to ze zwiększeniem liczby neuronów w sieci, co zwiększa liczbę jej parametrów nastawialnych w czasie uczenia i grozi wystąpieniem zjawiska przeuczenia (zaniku zdolności do generalizacji).

  Budując modele dla jednowymiarowych szeregów czasowych możemy przyjmować, że problem doboru właściwych informacji wejściowych dla sieci sprowadza się do sformułowania odpowiedzi na trzy pytania:

  a) jak długi musi być okres czasu, z którego pochodzić będą informacje wprowadzane na wejścia sieci?

  b) czy z ustalonego okresu należy na wejścia sieci wprowadzać wszystkie wartości, czy tylko wybrane; a jeśli wybrane, to które.

  c) czy obok samych tylko wcześniejszych wartości zmiennej będącej przedmiotem szeregu czasowego, potrzeba na wejście sieci wprowadzać wartości innych zmiennych sterujących „z zewnątrz" zmiennością czasową szeregu?

W większości przypadków udzielenie odpowiedzi na te pytania jest trudne i wymaga przeprowadzenia wielu eksperymentów. Narzędziem wspomagającym podjęcie tego typu decyzji może być opisany wyżej algorytm genetyczny, który w tym przypadku będzie decydować głównie o tym, które z opóźnionych wartości szeregu należy wprowadzać na wejście sieci (kolejne geny chromosomu będą odpowiadały kolejnym opóźnieniom więc pojawienie się jedynki na określonej pozycji świadczyć będzie o potrzebie uwzględnienia wartości opóźnionej o dany odcinek czasu, zaś zero wskazywać będzie na możliwość jej pominięcia; przykład zastosowania algorytmu genetycznego do rozwiązania tego problemu znaleźć można w pracy: [Lula, 1999]);

- *wydzielenie jednorodnych fragmentów szeregu* - celem tego etapu wstępnej analizy danych jest wyodrębnienie z skomplikowanego szeregu takich fragmentów, które cechują się podobnymi właściwościami (na przykład można zgrupować oddzielnie wszystkie fragmenty szeregu odpowiadające hossie albo bessie giełdowej od fragmentów odpowiadających długotrwałym stanom równowagi). Zakłada się przy tym, że w toku dalszych prac skonstruowany zostanie oddzielny model dla każdego wydzielonego fragmentu szeregu. Ten etap analizy ma charakter opcjonalny i może zostać zrealizowany tylko wtedy, gdy dysponujemy odpowiednio dużą liczbą obserwacji i mamy powody sądzić, że w rozważanym szeregu występują takie fragmenty o wyraźnie zróżnicowanej dynamice;

- *operacjonalizacja danych* – pojęcie operacjonalizacji było już wyżej objaśniane w kontekście sieci neuronowych rozwiązujących typowe problemy regresyjne. W przypadku szeregów czasowych operacjonalizacja oznacza dokładnie to samo, tylko sposób jej wykonania musi być ściśle powiązany z celem prowadzonych prac badawczych. Jeśli w centrum uwagi badacza znajduje się fakt występowania i wartość **zmian** w szeregu (na przykład wykrywanie symptomów zwyżki lub obniżki kursu określonych akcji wraz z oszacowaniem, czy będzie to duża, czy mała zmiana), a nie bezwzględny poziom wartości zmiennej tworzącej szereg, to przed utworzeniem modelu warto szereg pierwotny zastąpić szeregiem różnic pomiędzy kolejnymi wartościami. Okazuje się, że sieć ma wtedy znacznie łatwiejsze zadanie, więc prawdopodobieństwo sukcesu znacząco wzrasta. Można pójść jeszcze dalej, na przykład jeśli badacza interesuje wyłącznie kierunek zmian, a nie ich wartości, to w wyznaczonym szeregu różnic należy wartości dodatnie zastąpić wartością „1”, wartości ujemne wartością „-1” i starać się utworzyć sieć prognozującą właśnie te „sygnalizacyjne” wartości na podstawie wcześniejszych elementów pierwotnego bądź przetworzonego szeregu;

- *skalowanie* - w przypadku stosowania wielu typów sieci neuronowych występuje potrzeba wcześniejszego przeskalowania danych w celu uzyskania wartości pochodzących z określonego przedziału, dozwolonego z punktu widzenia właściwości sztucznych neuronów wchodzących w skład sieci. Zachodzi jednak pewna znamienna różnica pomiędzy skalowaniem danych dla modeli regresyjnych oraz skalowaniem szeregu czasowego W przypadku tworzenia przez sieć neuronową modeli regresyjnych, dopuszczalny był każdy rodzaj skalowania, przekształcający oryginalne dane do właściwego przedziału wartości. Natomiast w przypadku analizy szeregów czasowych wielu autorów (np. [Azoff, 1994]) wskazuje na potrzebę stosowania tylko takich metod przekształcania szeregu, które nie tylko pozwalają na uzyskanie wartości z określonego przez użytkownika zakresu, ale które jednocześnie zachowują informację o znaku poszczególnych (oryginalnych) składników szeregu.

Właściwe zaprojektowanie i przeprowadzenie wszystkich czynności wchodzących w skład wstępnego przetworzenia szeregu czasowego jest w wielu przypadkach czynnikiem decydującym o sukcesie lub porażce prac, mających na celu stworzenie neuronowego modelu szeregu czasowego. Realizacja wstępnej analizy danych wymaga przy tym często zastosowania wyrafinowanych metod obliczeniowych, ale jest to mało uciążliwe dla użytkownika, ponieważ jest z reguły realizowana przez specjalizowane w tym kierunku narzędzia programowe, dołączane do pakietu oprogramowania używanego do budowy i wykorzystania sieci neuronowych, albo istniejące samodzielnie. Zdaniem autorów na szczególną uwagę zasługuje w tym zakresie dostępny nieodpłatnie w sieci Internet program *TimeStat* (przeznaczony do pracy pod kontrolą systemu MS Windows).

Określając strukturę sieci neuronowej wykorzystywanej do opisu szeregu czasowego (lub jednej z wydzielonych z niego składowych) należy brać pod uwagę:

a) liczbę danych wejściowych (gdyż one determinują liczbę neuronów w warstwie wejściowej),

b) liczbę prognozowanych wartości, określających liczbę neuronów wyjściowych (zwykle tworzone są modele pozwalające na prognozowanie jednej wartości, ale bywają wyjątki od tej reguły i sieci neuronowe także mogą je obsługiwać),

c) złożoność prawidłowości opisujących zachowanie szeregu (mającą swoje odzwierciedlenie w strukturze części ukrytej sieci – im bardziej skomplikowane właściwości szeregu chcemy wykrywać i opisywać, tym więcej elementów musimy wbudować w warstwę ukrytą budowanej sieci).

Spośród wymienionych elementów najwięcej trudności sprawia poprawne określenie struktury części ukrytej sieci. Podejmowanie decyzji w tym zakresie może być wspomagane przez stosowanie algorytmów służących do projektowania struktury sieci neuronowej, w tym także przez wzmiankowane tu wielokrotnie algorytmy genetyczne, jednak w ogólnym przypadku znalezienie optymalnego rozwiązania wymaga sporej wiedzy, doświadczenia oraz… odrobiny szczęścia. Można się jednak pocieszać faktem, że również **suboptymalne** struktury sieci neuronowych mogą dostarczać bardzo dobrych, w pełni użytecznych praktycznie rozwiązań, więc problem doboru optymalnej struktury sieci jest bardziej problemem naukowym, niż wynikającym z wymagań praktyki.

Uczenie sieci jest zawsze bardzo ważnym etapem budowy modelu neuronowego. Ucząc sieci wykorzystywane do opisu sposobu zachowania się szeregów czasowych stosuje się algorytmy pozwalającego na przeprowadzenie treningu w trybie z nauczycielem. Przy wyborze metody uczenia należy uwzględnić podobne przesłanki, jak w przypadku modeli regresyjnych. W przypadku analizy danych uporządkowanych w czasie w sposób szczególny należy jednak potraktować problem wyodrębniania zbioru uczącego, walidacyjnego i testowego. Do rozwiązania tego problemu stosowane są dwa podejścia:

- stosując technikę „przesuwanego okna” wydziela się z szeregu wszystkie możliwe pary, w których pierwszym elementem jest zestaw informacji wejściowych (zbiór przesłanek dla prognozy), zaś drugim wartość wyjściowa szeregu (rzeczywista wartość, odpowiadająca pierwszemu elementowi pary, którą sieć powinna próbować „odgadnąć” dokonując prognozy). Następnie w sposób losowy przydziela się poszczególne pary (stanowiące wyodrębnione wzorce) odpowiednio do zbioru uczącego i zbiorów (lub zbioru) pozwalających na ocenę jakości sieci.

  Warto zauważyć, że stosowanie przedstawionego podejścia zwykle powoduje, że ten sam pojedynczy element szeregu czasowego (albo pewien fragment ich szeregu) może być wykorzystywany zarówno we wzorcach wykorzystywanych do uczenia, jak i we wzorcach służących ocenie sieci. W ten sposób **niespełniony** pozostaje warunek mówiący

o całkowitej niezależności omawianych zbiorów danych. Metodą pozwalającą na wyeliminowanie przedstawionego tu problemu jest modyfikacja sposobu wyodrębniania wzorców z szeregu czasowego, polegająca na tym, że przy tworzeniu poszczególnych par należy za każdym razem „przesuwać" okno nie o jeden okres, ale o taką liczbę okresów, która będzie gwarantować, że kolejne wzorce nie będą zachodziły na siebie. Niestety, taki sposób postępowania ma również wadę, gdyż znacznie ogranicza liczbę możliwych do wyodrębnienia z szeregu wzorców;

- dzieląc w sposób arbitralny szereg na podokresy - trzy (jeśli stosuje się zbiór uczący, walidacyjny i testowy) lub na dwa (gdy nie tworzy się zbioru testowego). Następnie w każdym wydzielonym fragmencie szeregu wydziela się wzorce przeznaczone do używania (odpowiednio jako uczące lub sprawdzające sieć) stosując opisaną powyżej technikę „przesuwanego okna". Wzorce wydzielone z pierwszego fragmentu szeregu są przy tym zwykle wykorzystywane na etapie uczenia, z drugiego służą jako elementy walidacyjne, zaś z trzeciego (o ile taki utworzono) wchodzą w skład zbioru testowego. Jeżeli dodatkowo wydzielone podokresy zostaną rozdzielone od siebie odpowiednią liczbą obserwacji, to z całą pewnością będziemy mogli stwierdzić, że żaden element szeregu nie należy jednocześnie do więcej niż jednego zbioru. Dodatkowym atutem przedstawionej metod podziału jest to, że nie ogranicza ona w istotny sposób liczby wzorców, które można wydzielić z szeregu. Natomiast wadą przedstawionego sposobu podziału danych jest to, że dane wykorzystywane do uczenia oraz do oceny sieci mogą nie być jednorodne, gdyż dotyczą wyraźnie różnych okresów zjawiska zarejestrowanego w postaci szeregu czasowego. Na przykład może się zdarzyć, że zbiór uczący utworzony zostanie z fragmentu szeregu odpowiadającego jego fazie wzrostowej, zaś do testowania wykorzystane zostaną dane odpowiadające fazie spadkowej; w takiej sytuacji trudno jest oczekiwać poprawnego działania modelu.

Budując model neuronowy należy zawsze poddać go gruntownej ocenie przed jego użyciem do rozwiązywania praktycznych problemów. Przeprowadzony sposób oceny musi przede wszystkim uwzględniać **cel**, któremu ma służyć konstruowany model.

Jakość modelu można ocenić poprzez wyznaczenie wartości różnorodnych mierników, a także poprzez porównanie sposobu jego funkcjonowania do rezultatów uzyskiwanych za pomocą innych technik, czy też poprzez oszacowanie skutków wykorzystania modelu w procesie decyzyjnym. Dokładne omówienie wskazanych technik, jak również wskazówki dotyczące literatury pozwalającej na zgłębienie problematyki oceny jakości neuronowych modeli szeregów czasowych można znaleźć między innymi w pracy: [Tadeusiewicz i in., 2000].

Kończąc prezentację uwag dotyczących konstrukcji neuronowych modeli szeregów czasowych warto wskazać na kilka prac dotyczących tej dziedziny, które można zalecić Czytelnikowi w celu kontynuowania studiów nad tym zagadnieniem. Jedną z pierwszych prac z tego zakresu była praca [White, 1988], zaś za klasyczny podręcznik uznaje się zwykle pracę E. M. Azoffa ([Azoff, 1994]), poświęconą problematyce budowy neuronowych modeli finansowych szeregów czasowych. Szereg interesujących informacji z tego zakresu znaleźć również można w pracach [Trippi i in., 1993] oraz [Refenes, 1995]. Ciekawą pozycją jest również wydana w języku polskim praca [Gately, 1999] przeznaczona przede wszystkim dla osób zainteresowanych praktycznymi aspektami budowy modeli neuronowych służących do opisu prawidłowości występujących na rynkach finansowych. Zbliżonym zagadnieniom poświęcona jest również część pracy [Dunis, 2001]. Natomiast w pracy [Zieliński, 2000] przedstawiono oparty na sieci perceptronowej system wspomagający podejmowanie decyzji inwestycyjnych.