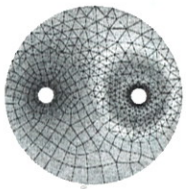
Wydawnictwo
Naukowe
AKAPIT

INFORMATYKA W TECHNOLOGII MATERIAŁÓW

Numer 1, Tom 1, Rok 2001



# PROBLEM WYBORU WŁAŚCIWEJ ARCHITEKTURY SIECI NEURONOWEJ

RYSZARD TADEUSIEWICZ

## *PROBLEM OF SELECTION OF PROPER ARCHITECTURE FOR NEURAL NETWORK*

*Abstract*

*Review of problems connected with applications of artificial neural networks in generally understood metallurgy is discussed in the paper. Conditions, which particular tasks impose on the selection of the network architecture, are pointed out. These suggestions, connected with earlier Author's research on training the networks, are a useful guide for further experiments aiming at an application of the artificial neural networks as a tool for solving of various problems in metallurgy. Scientific basis for practical metallurgical applications is given in the paper.*

## 1. Wstęp

W ostatnich latach bardzo gwałtownie wzrosło zainteresowanie sieciami neuronowymi. Są one z powodzeniem stosowane w niezwykle szerokim zakresie problemów, w tak różniących się od siebie dziedzinach jak finanse (Tadeusiewicz 1997), medycyna (Izworski i Tadeusiewicz 2000), diagnostyka techniczna (Tadeusiewicz i in. 2000) czy zajmująca nas tu najbardziej inżynieria materiałowa (Sayeh i in. 1990). W rzeczywistości sieci neuronowe mogą być zastosowane wszędzie tam, gdzie pojawiają się problemy związane z **predykcją** (Rojas i in. 2000, Rao i Mukherjee 1996, Kim i in. 1995a), **klasyfikacją** (Tadeusiewicz i in. 1999), **projektowaniem** (Johansen i in. 1995) czy **sterowaniem** (Demirci i in. 1996) i **kontrolą** (Kim i in. 1995b). Warto uświadomić sobie, że to powodzenie sieci neuronowych, datujące się przynajmniej od początku lat 90. (Tadeusiewicz 1993) ale dostrzegalne od kilku lat także w metalurgii[1] (Workman i Walker 1996), gdzie bywają one obecnie stosowane nawet do projektowania całych fabryk (Torres i in. 2000), jest związane z kilkoma konkretnymi czynnikami:

- Sieci neuronowe są bardzo wyrafinowaną techniką modelowania (Colla i in. 2000), zdolną do odwzorowywania nadzwyczaj złożonych funkcji, szczególnie często pojawiających się w zagadnieniach inżynierii materiałowej (Payne i in. 1993). W szczególności istotne jest to, że modele tworzone z pomocą sieci neuronowych mają charakter nieliniowy (Draeger and Gesthuisen 1996), co istotnie wzbogaca możliwości ich zastosowań (Lee i Twu 1995). Rozwiniemy nieco ten temat, żeby wskazać na wagę formułowanych w tym referacie rad i propozycji.

  Przez wiele lat powszechnie stosowaną techniką matematycznego opisywania różnych obiektów i

*R. Tadeusiewicz, Katedra Automatyki, Akademia Górniczo-Hutnicza*

[1] W przypadku zastosowań sieci neuronowych w metalurgii ogromne zasługi w ich wykorzystaniu i upowszechnianiu mają doroczne seminaria z serii ***NeuroMet,*** w ramach których prezentowany był także ten referat.



ISSN 1641-8581

procesów było modelowanie liniowe (Stone i Krishnamurthy 1996). Takie postępowanie jest z powodzeniem stosowane także obecnie, przynosząc korzystne rezultaty, głównie z uwagi na dobrze znaną strategię optymalizacji stosowaną przy konstrukcji modeli tego typu. Jednak wszędzie tam, gdzie nie ma podstaw do aproksymacji liniowej występujących zjawisk i procesów (a w inżynierii materiałowej przypadki takie są w praktyce dość częste (Cios i in. 1994)), modele liniowe nie sprawdzały się, prowadząc niekiedy do formułowania niesłusznych opinii o całkowitym braku możliwości matematycznego opisywania takich czy innych systemów (Cooper i in. 1996). W takich przypadkach, przy rozwiązywaniu tych trudnych i kłopotliwych zagadnień odwołanie się do modeli tworzonych z wykorzystaniem sieci neuronowych (a więc modeli, które bez trudu mogą odwzorować zależności nieliniowe) może być najszybszym i najwygodniejszym rozwiązaniem problemu (Gavard i in. 1996).

- Sieci neuronowe umożliwiają swobodne i łatwe (bez konieczności samodzielnego formułowania przez użytkownika skomplikowanych hipotez) tworzenie modeli takich systemów, dla których mechanizmy przyczynowe zachodzących w nich zjawisk są nie do końca poznane, lub są znane, ale zbyt skomplikowane, by je w całości modelować (Hill i in. 1994). Dzięki procesowi automatycznego uczenia, opartemu wyłącznie na seriach empirycznych obserwacji, można dla tych systemów uzyskać neuronowy model gwarantujący zadowalający stopień zgodności swego funkcjonowania z zachowaniem oryginalnego obiektu (Hill i in. 1996).
- Zazwyczaj modele neuronowe cechuje dodatkowo mała złożoność obliczeniowa, co powoduje, że uciekamy się do nich także w sytuacji posiadania pełnego modelu matematycznego badanego zjawiska, który jednak jest zbyt złożony (i czasochłonny obliczeniowo (Slattery 1995)), by z niego korzystać na przykład w systemach automatycznej regulacji (Gradisek i in. 1996). Dotyczy to w szczególności systemów dynamicznych opisywanych złożonymi układami równań różniczkowych zwyczajnych (Liao i Chen 1994) (np. zagadnienia sterowania mechanizmów o wielu stopniach swobody) lub cząstkowych (procesy cieplne (Yo i Okigata 1993), złożone formy drgań mechanicznych, przeróbka plastyczna metali (Di and Thomson 1997) itp.)
- Sieci umożliwiają także kontrolę nad złożonym problemem wielowymiarowości, który przy stosowaniu innych metod znacząco utrudnia próby modelowana funkcji nieliniowych z dużą liczbą zmiennych niezależnych (tzw. funkcji wektorowych) (Konishi 1993).
- Jak już wspomniano, sieci neuronowe w praktyce same konstruują potrzebne użytkownikowi modele, ponieważ automatycznie uczą się na podanych przez niego przykładach (Keshavaraj i in. 1995). Odbywa się to w taki sposób, że użytkownik sieci gromadzi reprezentatywne dane pokazujące, jak manifestuje się interesująca go zależność, a następnie uruchamia algorytm uczenia, który ma na celu automatyczne wytworzenie w pamięci sieci potrzebnej struktury danych. Opierając się na tej samodzielnie stworzonej strukturze danych sieć realizuje potem wszystkie funkcje związane z eksploatacją utworzonego modelu.
- Chociaż zatem użytkownik potrzebuje pewnej, w głównej mierze empirycznej wiedzy dotyczącej sposobu wyboru i przygotowania danych stanowiących przykłady, a także wyboru właściwego rodzaju sieci neuronowej oraz sposobu interpretacji rezultatów (Kim i in. 1995a), to jednak poziom wymaganej od użytkownika wiedzy teoretycznej, niezbędnej do skutecznego zbudowania modelu (Lee i Kim. 1995), przy stosowaniu sieci neuronowych jest znacznie niższy niż w przypadku stosowania tradycyjnych metod tworzenia modeli – na przykład metod statystycznych (Walker i Hill 1994).
- Pociągająca jest również taka własność sieci neuronowych, że stanowią one (w jakimś zakresie) naśladownictwo działania ludzkiego umysłu. Wprawdzie sieci oparte są na bardzo prostym modelu, przedstawiającym wyłącznie najbardziej podstawową istotę działania biologicznego systemu nerwowego, jednak ich działanie wzbudza ciekawość jako jedna z prób przeniknięcia istoty działania ludzkiego mózgu (Tadeusiewicz 1998). Niektórzy w związku z tym sądzą, że w przyszłości rozwój modelowania neuro-biologicznego może doprowadzić do powstania prawdziwych komputerów inteligentnych.

Tak będzie jednak dopiero w przyszłości - być może dość odległej. Tymczasem już dziś, „proste" modele sieci neuronowych, stanowią godne uwagi, bardzo użyteczne narzędzie. Są tacy, którzy przypuszczają, że będzie to prawdziwie „cudowny oręż" oddany do dyspozycji osób zajmujących się nauką stosowaną, w tym także technikami związanymi z metalurgią. Żeby jednak sensownie wykorzystać potencjalne możliwości tkwiące w sieciach neuronowych, trzeba umieć we właściwy sposób dobrać charakterystyki i cechy tych sieci do właściwości i cech zadania, które chcemy przy ich pomocy rozwiązywać. Jednym z kluczowych zagadnień w tym zakresie jest właściwy wybór optymalnej struktury sieci dla rozwiązania konkretnego zadania, jakie stoi przed użytkownikiem. Temu właśnie zagadnieniu poświęcona jest ta praca. Zanim jednak

podejmiemy problem wyboru struktury sieci, musimy kilka słów powiedzieć na temat generalnych zasad ich stosowania.

## 2. KORZYSTANIE Z SIECI NEURONOWYCH W WYBRANYCH ZAGADNIENIACH METALURGICZNYCH

Zakres problemów rozwiązywanych przez sieci neuronowe jest wyznaczony przez sposób ich działania oraz przez zastosowaną metodę uczenia (Cerqueira i in. 2000). Kluczowym elementem przy stosowaniu sieci jest fakt, że mogą one (całkiem same!) stworzyć złożony, nieliniowy model badanego zjawiska (patrz także rozdział 3), co ma generalnie liczne zastosowania. Przykładowo w obszarze metalurgii sieci w sposób udany zastosowano między innymi w następujących zadaniach:

- przewidywania właściwości mechanicznych stopów (Xia i in. 1996)
- projektowanie nadstopów (Cios i in. 1996)
- modelowania właściwości metali poddanych określonej obróbce cieplnej (Chidambaram i in. 1995) w zależności od ich składu
- modelowanie hydrometalurgicznych procesów ekstrakcji metali ziem rzadkich (Giles i in. 1996)
- przewidywanie właściwości mechanicznych płaskich wyrobów stalowych po procesie walcowania (Myllykoski i in. 1996)
- przewidywanie sił w procesie walcowania na zimno (Larkiola i in. 1996)
- prognozowania struktury powierzchni przekroju poprzecznego wyrobów walcowanych na zimno (Jung i in. 1996)
- analizy zjawiska pękania zmęczeniowego nadstopów na bazie niklu (Fujii i in. 1996)
- badania związku zjawisk obserwowanych w skali mikro i w skali makro w materiałach z pamięcią kształtu (Isalgue i in. 1994)
- sterowania procesem spiekania rudy (Matsuda i in. 1994)
- przewidywanie procesu korozji metali w glebie (Cuo i in. 1995)

W literaturze jest także pełno doniesień o skutecznym zastosowaniu sieci neuronowych w licznych zadaniach związanych jedynie pośrednio (chociaż blisko) z kwestiami metaloznawczymi. Dotyczy to między innymi takich zadań jak:

- modelowanie rozkładu temperatur i sterowanie technologią prowadzenia narzędzie podczas spawania gazowego (Middle i Khalaf 1994), elektrycznego (Jin i in. 1996) i laserowego (Sergi 1996)
- przewidywanie metalograficznej mikrostruktury warstw metali nakładanych techniką galwaniczną (Samways 1996)
- korygowanie błędów stykowych metod pomiaru rozmiaru i położenia obiektów poddawanych zautomatyzowanej obróbce plastycznej (Shen i Moon 1996)
- polepszanie jakości sygnałów w ultradźwiękowych defektoskopach wykrywających wady odlewów (Thavasimuthu i in. 1996)
- prognozowanie żywotności narzędzi tnących wykonanych ze stali stopowych (chromowo-molibdenowych) (Teshima i in. 1993)
- prognozowanie deformacji cieplnych części maszyn w zależności od intensywności ich pracy oraz struktury metalograficznej materiału z którego są wykonywane (Hattori i in. 1996)

Warto odnotować fakt, że wykorzystaniem sieci neuronowych do optymalizacji określonych procesów metalurgicznych interesują się nie tylko zespoły i ośrodki naukowe. Na przykład w pracy (Aistleitner i in. 1996) opisano bardzo korzystne praktyczne efekty, jakie uzyskała znana firma ***Voest-Alpine*** podczas stosowania sieci neuronowych do oceny stopnia ekscentryczności walców w kontekście osiągalnych tolerancji grubości wyrobów walcowanych. Fakt ten powinien zainteresować tych badaczy, którzy szczególnie cenią bliski związek prowadzonych prac naukowych z potrzebami praktyki.

Technika sieci neuronowych może być użyteczna także i z tego powodu, że stosunkowo łatwo jest ją komponować z innymi technikami rozwiązywania określonych problemów. W metalurgii i obróbce metali interesujące wyniki uzyskano na przykład łącząc technikę tradycyjną z modelem neuronowym. Przykładowo takim sposobem posłużono się w pracy (Zgonc i Achenbach 1996), gdzie wykazano efektywność hybrydowego podejścia (kombinującego sieci neuronowe i metody elementów skończonych) w zadaniu modelowania linii przełomu wyrobu aluminiowego.

W większości zadań opisanych wyżej zastosowanie sieci neuronowej ukierunkowane było na określony cel praktyczny. Model realizowany przez sieć zastępował w tych pracach na ogół brakującą wiedzę o badanych zjawiskach i pozwalał na skuteczne działanie (np. sterowanie) bez konieczności odwoływania się do teorii. Bywają jednak prace, w których sieć używana jest wręcz do tego, by z jej pomocą **stworzyć** brakującą wiedzę na temat jakiegoś mało znanego zjawiska. Przykładem takiej pracy może być publikacja (Rademan i in. 1996) w której pokazano, jak z pomocą sieci neuronowej można przetworzyć surowe dane empiryczne na wiedzę pozwalającą lepiej zrozumieć

skomplikowane zjawisko (w pracy (Rademan i in. 1996) zjawiskiem tym było ługowanie wykorzystywane w wybranych procesach hydrometalurgicznych).

W przywołanych tu przykładach (będących jedynie małą próbką dostępnej obecnie literatury) uzyskano bardzo dobrą zgodność działania modeli neuronowych z funkcjonowaniem rzeczywistego zjawiska lub procesu, w związku z czym model neuronowy można było użyć jako wygodne narzędzie obliczeniowe, szczególnie użyteczne dla prognozowania zjawisk zachodzących w warunkach, które nie mogły by być bezpośrednio wytworzone i obserwowane doświadczalnie. Szczególnie interesujące okazują się przy tym badania i analizy prowadzone z użyciem sieci neuronowych, które pozwalają ustalić wrażliwość rozważanego zjawiska na te czynniki fizyczne, chemiczne i technologiczne, które w normalnym doświadczeniu nie mogą być w prosty sposób uzmiennione niezależnie od pozostałych. Na przykład w cytowanej wyżej pracy (Fujii i in. 1996) zbudowano model neuronowy zjawiska pękania nadstopów, przy czym model ten uzależniał badane zjawisko aż od 51 parametrów chemicznych i fizycznych. Jednak po zbudowaniu i pozytywnym zweryfikowaniu modelu można było przy jego pomocy zbadać wiele trudnych do empirycznego zbadania zależności – na przykład wpływu mikrostruktury stopu na jego właściwości zmęczeniowe. Podobnie w pracy (Cuo i in. 1995) udało się w neuronowym modelu przebadać oddzielnie różne uwarunkowania fizyczne i chemiczne decydujące o szybkości i formach korozji przedmiotów metalowych pozostawionych w glebie, co ma duże znaczenie dla budowy maszyn rolniczych, ale także dla kryminalistyki i archeologii.

## 3. OGRANICZENIA W STOSOWALNOŚCI SIECI

Wszystkie wyżej wymienione sukcesy w stosowaniu sieci neuronowych uzyskano w bardzo prostym schemacie działania sieci, który określany jest w literaturze jako zasada *feedforward*. Jak wiadomo polega ona na tym, że do wybranych kanałów wejściowych sieci neuronowej wprowadza się pewne zmienne wejściowe (czyli dane). Dane te rozpływają się w sieci, przekazywane od neuronu do neuronu za pośrednictwem złącz wyposażonych w możliwość uczenia się, w wyniku czego neurony warstw ukrytych sieci wypracowują w niej pewne charakterystyki danych wejściowych, pomocne przy rozwiązywaniu postawionego zadania, zaś jej wyjścia definiują zmienne wyjściowe (czyli rozwiązania). Nic więcej sieci nie potrafią.

Fakt, że ten prosty schemat działania prowadzi do pożądanego wyniku końcowego osiąga się głównie dzięki odpowiedniemu zastosowaniu procesu uczenia sieci, podczas którego w sieci wydobywana jest i podlega zapamiętaniu (w formie danych rozproszonych po wszystkich dostępnych złączach) wiedza zawarta w konkretnych przykładach, reprezentujących rozważany problem. Wiedza ta może następnie posłużyć (w rezultacie generalizacji zebranych doświadczeń) do tego, żeby na bieżąco rozwiązywać także te zadania, które bezpośrednio nie były prezentowane w czasie uczenia sieci, ale w określonym sensie są podobne do tych zagadnień, które wykorzystywano w trakcie uczenia sieci. Dlatego też mogą one być stosowane wszędzie tam ale też i wyłącznie tam, gdzie na podstawie pewnych znanych informacji szczegółowych wnioskuje się o pewnych ogólnych (nieznanych) prawidłowościach. Zadania dające się „wpisać" w podany wyżej schemat rozwiązuje się z wykorzystaniem sieci neuronowych stosunkowo łatwo i wygodnie, przy czym bardzo często okazuje się, że rozwiązania uzyskane z pomocą sieci są znacząco lepsze, niż rozwiązania uzyskiwane z pomocą innych technik i innych metod.

W tradycyjnym podejściu do modelowania matematycznego różnych złożonych systemów (np. w modelowaniu liniowym wykorzystującym podejście statystyczne) stosowane są algorytmy określające taką konfigurację modelu, która zapewnia osiągnięcie minimum globalnego funkcji błędu. Oznacza to, że stworzony (na przykład z pomocą metody regresji) model matematyczny liniowy jest najlepszym możliwym odwzorowaniem posiadanego zbioru danych - ale najlepszym w klasie modeli liniowych. Nie da się zbudować modelu liniowego, który by lepiej pasował do posiadanych danych niż model uzyskany za pomocą metod regresji – ale nie jest wykluczone, że istnieje model nieliniowy, który będzie opisywał dane nieporównanie lepiej. Prawdę powiedziawszy prawie wszystkie złożone systemy, których modele chcemy budować, są w rzeczywistości w mniejszym albo większym stopniu nieliniowe. Decydujemy się jednak często na opisywanie ich przy pomocy formuł (modeli) liniowych, ponieważ zarówno opisane wyżej wyznaczenie modelu liniowego, jak i jego wykorzystywanie jest szczególnie łatwe, a przez to szczególnie wygodne. Czasem jednak niedokładność wnoszona przez model liniowy, nawet taki optymalnie wyznaczony, jest niedopuszczalnie duża. W takim przypadku musimy próbować stworzyć dla modelowanego obiektu lub rozważanego procesu model nieliniowy – i tu się zaczynają problemy. Pojęcie „model liniowy" jest pojęciem jednoznacznym. Dla danego zbioru danych istnieje tylko jeden optymalny model liniowy i zadanie jego wyznaczenia (poprzez minimalizację błędu) jest zadaniem dobrze określonym, stąd można je łatwo rozwiązać.

Natomiast pojęcie modelu nieliniowego jest pojęciem wysoce wieloznacznym. „Nieliniowy" to w istocie oznacza każdy, który nie jest liniowy – a takich jest nieskończenie dużo. W związku z tym znalezienie modelu nieliniowego oznacza w istocie wykonanie **dwóch** czynności: wyboru kształtu modelu (jego ogólnego wzoru matematycznego) oraz doboru takich parametrów tego modelu, żeby odwzorowywał posiadane dane w optymalny sposób. Druga część zadania daje się zautomatyzować, to znaczny można zastosować odpowiednie metody statystyczne (technikę regresji nieliniowej) do wyznaczenia optymalnych parametrów modelu nieliniowego **o uprzednio wybranym kształcie.** Natomiast trudniejsze jest zadanie wyboru (wymyślenia) kształtu formuły matematycznej, która będzie mogła pełnić rolę modelu określonego zjawiska (lub określonego procesu). Tu nie ma żadnych ułatwień ani żadnych ogólnych reguł. Twórca modelu musi po prostu **odgadnąć** jego kształt. Jeśli odgadnie dobrze – uzyska dobre działanie modelu. Jeśli odgadnie źle, to nie pomoże żadna, nawet najbardziej wyrafinowana optymalizacja parametrów modelu – jego działanie będzie stale nie zadowalające.

Sieci neuronowe wnoszą w modelowaniu nową jakość. Sieć w trakcie procesu uczenia może całkiem sama znaleźć nieliniowy model rozważanego systemu, przy czym twórca sieci nie musi jej podawać żadnych założeń dotyczących kształtu modelu. Stosowanie modeli opartych na sieciach neuronowych a zwłaszcza opartych na tej technice modeli nieliniowych zwiększa zatem istotnie możliwości modelowania. Jednak ceną płaconą za tę wygodę jest brak pewności, że osiągnięty w trakcie uczenia sieci poziom błędu osiągnął swój poziom minimalny, co oznacza, że uzyskany model może być odległy od modelu optymalnego i użytkownik nigdy nie wie, jak bardzo uzyskany model „odstaje" od modelu optymalnego, jaki mógłby być uzyskany w tych samych warunkach. Dodatkowa wada modelu neuronowego polega na tym, że uzyskany w trakcie uczenia sieci model nie podlega dalszemu doskonaleniu, wiec błąd modelowania nie może być w dalszym zakresie zmniejszany – na przykład przez dodawanie dodatkowych informacji.

Jest jednak sprawą oczywistą, że **nie każdy problem może być rozwiązany przy pomocy sieci neuronowych.** Na przykład, jeśli ktoś chciałby znać wyniki losowania Totolotka, które odbędzie się w przyszłym tygodniu, a znałby jako jedyną daną wejściowa tylko numer własnego buta, to ani sieć neuronowa, ani żadna inna technika nie jest w stanie mu dostarczyć wymaganej procedury obliczeniowej. Wynika to z faktu, że po prostu nie istnieje reguła (ani neuronowa, ani żadna inna), która na podstawie dostępnej danej (numer buta) pozwoliłaby wyznaczyć wartość potrzebnej zmiennej wyjściowej (numery „szczęśliwych" liczb podczas losowania loterii).

W rozważanym tu przypadku sprawa jest w miarę prosta, ponieważ jest rzeczą z góry wiadomą, że nie istnieje żaden związek pomiędzy rozważanymi wartościami. W rzeczywistości, jeśli losowanie Totolotka przeprowadzane jest prawidłowo, to **w ogóle** nie istnieją fakty, których znajomość pozwoliłaby wnioskować o przyszłotygodniowych wynikach, zatem nikt rozsądny nie będzie wymagał, żeby sieć neuronowa przeprowadziła w tej sprawie skuteczną prognozę. Nie zawsze jednak tak bywa i czasem badacze wytrwale „męczą" sieci neuronowe w nadziei na uzyskanie jakiegoś wyniku (na przykład modelu jakiegoś zjawiska), który po prostu nie istnieje.

Dlatego też rekomendując stale (w ramach kolejnych seminariów serii NeuroMet) korzystanie z sieci neuronowych przy rozwiązywaniu wielu zagadnień badawczych i praktycznych, związanych z szeroko rozumianą problematyką metalurgii, sformułować powinniśmy ważny wymóg związany z ich użyciem. Badacz chcący użyć sieci neuronowych musi wiedzieć (lub przynajmniej musi mieć mocne podejrzenia), że rzeczywiście **istnieje** zależność pomiędzy proponowanymi, znanymi sygnałami (traktowanymi jako wejścia sieci) a nieznanymi wartościami wynikowymi, które chciałby związać z jej wyjściami. Zależność ta może być nieznana co do swojego charakteru i dokładnego kształtu (to ustali sama sieć w trakcie procesu ucznia) a także wiedza o niej, zawarta w konkretnych empirycznych przykładach, zgromadzonych w formie zbioru uczącego, może być mało czytelna (bo z tym sieć sobie także poradzi) – jednak musi naprawdę **istnieć.** Warto wiedzieć, że przy posiadaniu dużego zbioru uczącego, poszczególne budujące go obserwacje mogą być nawet mało precyzyjne, ponieważ sieć podczas nauki poradzi sobie z problemem „ukrycia" użytecznej wiedzy na skutek faktu, że w każdym konkretnym zaobserwowanym przypadku ogólna (nieznana, ale istniejąca) prawidłowość może być zniekształcona przez szumy. Jednak dla sensownego zastosowania sieci konieczne jest, żeby ta konkretna zależność **istniała** i miała charakter regularny i powtarzalny.

Ostatnią kwestię można nawet wzmocnić, odnotowując, że problem szumu w danych wejściowych generalnie w technice sieci neuronowych nie jest problemem krytycznym. Odpowiednio nauczona sieć neuronowa potrafi bowiem dość skutecznie „odfiltrować" wszelkie szumy i umie prowadzić skuteczne obliczenia w oparciu o wypadkowe (najczęściej uśrednione) tendencje i trendy. Trzeba jednak pamiętać, że ta niewrażliwość sieci na szumy manifestuje się również na etapie eksploatacji sieci, więc podczas jej eksploatacji drobne subtelności w danych wejściowych nie będą miały wpływu na uzyskiwane odpowiedzi sieci. Fakt ten z pewnością obniża trafność i precy-

zję neuronowych wyników i prognoz.

Opisany wyżej stan niepewności co do jakości i trafności wyniku, jaki sieć wyprodukuje, jest dosyć częstą cechą obliczeń neuronowych. Wynika on z faktu, że osoba korzystająca z sieci neuronowych z reguły nie zna dokładnej natury związku pomiędzy wejściami a wyjściami, więc buduje całość swego wnioskowania wyłącznie w oparciu o dane empiryczne i próbę ekstrapolacji wykrytych (w trakcie uczeni sieci) zależności i trendów. Jeśli znałaby tę zależność, to mogłaby zdefiniować bezpośrednio jej model bez odwoływania się do sieci i ryzykownego mechanizmu uczenia.

## 4. DOBÓR STRUKTURY SIECI

Jak już wspomniano wyżej, jakość wyników uzyskiwanych za pomocą sieci neuronowych jest bardzo mocno uzależniona od właściwego doboru najlepszej struktury sieci do właściwości zadania podlegającego rozwiązywaniu. Przedstawimy tu kilka najbardziej popularnych struktur i przy każdej z nich naświetlimy możliwie dokładnie problem kryteriów doboru właściwej struktury tego typu sieci.

### 4.1. Sieci liniowe

Ogólna zasada stosowana w nauce głosi, że w przypadku gdy istnieje możliwość wyboru pomiędzy modelem prostym i bardziej złożonym, należy zawsze preferować model prostszy – o ile oczywiście ten drugi nie dopasowuje się znacząco lepiej do posiadanych danych. Stosując terminy charakterystyczne dla aproksymacji funkcji, można stwierdzić, że najprostszym modelem aproksymującym matematycznie pewną zaobserwowaną praktyczną zależność jest model liniowy. W modelu takim funkcją dopasowywaną do posiadanych danych jest zawsze hiperpłaszczyzna, a zadanie polega na znalezieniu właściwego położenia i właściwego nachylenia tej hiperpłaszczyzny. W problemach klasyfikacyjnych wspomniana hiperpłaszczyzna jest umieszczana w takiej pozycji, aby **oddzielać** od siebie dwie rozważane klasy. Działa ona wtedy jako liniowa funkcja dyskryminująca. Z kolei w zagadnieniach regresyjnych umieszczana jest ona w taki sposób, aby dobrze reprezentowała generalną tendencję reprezentowaną przez dane, chociaż precyzyjne dopasowanie hiperpłaszczyzny do wszystkich subtelności zawartych w zgromadzonych danych jest zwykle raczej trudne ze względu na sztywność jej formy - zarówno jako formuły matematycznej jak i jako formy geometrycznej. Model liniowy jest zwykle reprezentowany przy użyciu macierzy współczynników kierunkowych o wymiarach $N \times N$ (ustalających położenie hiperpłaszczyzn tworzących model) i wektora wyrazów wolnych o wymiarach $N \times 1$ definiującego przesunięcia hiperpłaszczyzn w stosunku do początku układu współrzędnych.

Stosując terminologię związaną z sieciami neuronowymi można stwierdzić, że model liniowy jest reprezentowany przez sieć nie posiadającą warstw ukrytych, zaś znajdujące się w warstwie wyjściowej neurony są w pełni liniowe (tzn. są to neurony, w których łączne pobudzenie wyznaczane jest jako liniowa kombinacja wartości wejściowych i które posiadają liniową funkcję aktywacji). Wagi neuronów odpowiadają wzmiankowanej wyżej macierzy, zaś wartości progowe neuronów – wektorowi wyrazów wolnych. W trakcie działania sieci wykonywana jest operacja mnożenia wejść przez macierz wag, a następnie do uzyskanego wyniku dodawany jest wektor wyrazów wolnych, co łącznie tworzy wektor sygnałów wyjściowych.

Sieci liniowe stanowią dobry punkt odniesienia, z którym porównuje się jakość innych sieci neuronowych. Jest bardzo możliwe, że problem, który uważany był za bardzo złożony, może zostać w rzeczywistości rozwiązany równie dobrze przez sieć liniową jak i przez inną sieć neuronową, o złożonej nieliniowej strukturze. Zgodnie z uwagami, które przytoczono na początku tej sekcji – bezwarunkowo należy w takim przypadku wykorzystywać model liniowy. W szczególności jeśli zbiór uczący składa się tylko z kilku przypadków, to stosowanie bardziej złożonych modeli nie jest uprawomocnione i korzystanie z modelu liniowego (i odpowiednio z liniowej sieci neuronowej) jest najbardziej właściwą metodą rozwiązania problemu.

### 4.2. Wielowarstwowe perceptrony klasy MLP

Istnieje wiele typów i rodzajów sieci neuronowych, różniących się między sobą strukturą i zasadami działania, ale chyba najpopularniejsza obecnie architektura sieciowa związana jest z koncepcją **wielowarstwowego perceptronu** zwanego w skrócie MLP. Koncepcja ta została po raz pierwszy opisana przez Rumelharta oraz McClellanda (w 1986), a jej dokładne omówienia znajdują się większości książek z dziedziny sieci neuronowych (np.: Tadeusiewicz 1993).

Przypomnijmy: Sieć taka składa się z wielu neuronów ułożonych w warstwy. Każdy z neuronów oblicza ważoną sumę swoich wejść, a wyznaczony w ten sposób poziom pobudzenia staje się argumentem funkcji przejścia, która oblicza wartość wyjściową neuronu. Neurony tworzą strukturę jednokierunkową, to znaczy przesyłanie sygnałów odbywa się w kierunku od wejścia do wyjścia – bez sprzężeń zwrotnych. Sieć można interpretować jako model typu wejście-wyjście, którego parametrami są wagi oraz wartości progowe. Sieć taka może modelować – przy odpowiedniej licz-

bie warstw i neuronów – funkcję o prawie dowolnej złożoności.

Określenie prawidłowej liczby warstw i neuronów w kolejnych warstwach jest bardzo ważnym etapem procesu konstrukcji każdego perceptronu wielowarstwowego. Liczba neuronów wejściowych i wyjściowych jest zdeterminowana przez rozwiązywany problem. Mogą pojawić się pewne wątpliwości związane dokładnym określeniem wejść, które wnoszą istotne informacje i jako takie powinny być użyte w sieci, ale problem ten był już omawiany w innych publikacjach.

Obecnie przyjmiemy, że zmienne wejściowe zostały już wybrane (na podstawie istotnych przesłanek merytorycznych albo w sposób intuicyjny) i założymy, że nie zachodzi potrzeba ich selekcjonowania, czyli przyjmujemy, że wszystkie dostępne zmienne wejściowe są istotne. Określenie liczby warstw ukrytych i liczby neuronów znajdujących się w tych warstwach nie jest już tak proste. Jako punkt wyjścia (który jest równie dobry, jak każde inne rozwiązanie), można przyjąć sieć z jedną warstwą ukrytą, zawierającą taką liczbę neuronów, która jest równa połowie sumy liczby neuronów wejściowych i liczby neuronów wyjściowych. Problem mniej arbitralnego (ale również nie zawsze pewnego) doboru właściwych liczebności liczby warstw i liczby neuronów w warstwach został dodatkowo omówiony w pracy (Tadeusiewicz 1997), trzeba jednak wyraźnie i jednoznacznie stwierdzić, że na ogół najlepsze wyniki otrzymuje się wybierając te liczebności w sposób empiryczny. Najistotniejszym problemem, jaki tu trzeba pokonać, jest sprzeczność pomiędzy dążeniem do posiadania możliwie najbardziej „inteligentnej" sieci (co skłania do użycia jak największej liczby neuronów ukrytych) a skłonnością takich właśnie przewymiarowanych sieci do przeuczania się w wyniku czego pojawia się problem nadmiernego dopasowania sieci do danych uczących.

Najprościej można pokazać istotę tego zagadnienia opisując dopasowanie modelu do danych przy pomocy sieci neuronowej w taki sposób, jakbyśmy poszukiwali modelu dopasowując wielomiany. Pojęciowo jest to w pełni równoważne, gdyż w obu przypadkach zasadniczy problem jest taki sam – jak dopasować postać modelu do zasadniczego kształtu aproksymowanej zależności, ale nie do drobnych szczegółów reprezentowanych przez pojedyncze dane.

Wielomian jest tu wygodnym przykładem, ponieważ zawiera człony wyłącznie w postaci stałych współczynników i potęg danej zmiennej. Na przykład:

$$y = 2x + 3$$

$$y = 3x^2 + 4x + 1$$

Różne wielomiany mają różne kształty (chodzi oczywiście o kształt wykresu jaki powstaje w momencie gdy wykreślimy zależność $y$ jako funkcji $x$). Wraz ze stosowaniem coraz wyższych potęg (a tym samym większej liczby członów) wielomiany przyjmują coraz bardziej złożone kształty.

Mając pewien konkretny zbiór danych można dopasować krzywą wielomianową (stanowiącą model) tak, aby opisywała ona te właśnie konkretne dane. Dane te są jednak najprawdopodobniej zniekształcone przez szumy, dlatego też nie należy oczekiwać, aby najlepszy model przechodził **dokładnie** przez każdy z punktów. Przeciwnie, dobry model to taki, który odwzorowuje ogólną postać poszukiwanej zależności, ale abstrahuje od drobnych, zwykle nieistotnych „zafalowań" wejściowych danych. Wiąże się z tym kwestia stopnia używanego wielomianu: wielomiany niskiego stopnia mogą być zbyt mało elastyczne do tego, aby przejść w pobliżu wszystkich punktów wyznaczanych przez posiadane dane, podczas gdy wielomiany wysokiego stopnia mogą być **zbyt** elastyczne, co zamanifestują dopasowując się zbyt dokładnie do konkretnych (być może trochę zakłóconych) danych. Skutek takich nadmiernych dopasowań łatwo zauważyć na wykresie wielomianu: „ściga" on konkretne dane poprzez przyjmowanie bardzo osobliwych kształtów, które nie są w rzeczywistości w żaden sposób powiązane z modelowaną, rzeczywistą funkcją.

Opisany wyżej przykład opisany został w kontekście aproksymacji funkcji za pomocą wielomianu, jednak w przypadku sieci neuronowych pojawia się dokładnie ten sam problem. Sieć z większą liczbą wag może modelować bardziej złożone funkcje i z tego powodu ma większą skłonność do **zbytniego** dopasowywania się do danych. Sieć z mniejszą liczbą wag może z kolei nie być dostatecznie mocnym narzędziem do opisu występującej w rzeczywistości zależności. Na przykład, sieć nie posiadająca warstw ukrytych może w rzeczywistości modelować wyłącznie proste zależności liniowe. Natomiast sieć o zbyt dużej liczbie warstw i zbyt dużej liczbie neuronów w warstwach ukrytych będzie miała skłonność do uczenia się „na pamięć" całego zbioru uczącego.

W związku z tym pojawia się pytanie dotyczące sposobu wyboru sieci o „właściwej" złożoności. Większe sieci prawie zawsze osiągają ostatecznie mniejszą wartość błędu, ale to może raczej wskazywać na ich przeuczenie niż na dobrą jakość modelu. Odpowiedzi na powyższe pytanie nie można poszukiwać w teorii sieci neuronowych, gdyż jej chwilowo nie ma. Dlatego w celu optymalizacji struktury perceptronu wielowarstwowego może być użycie procesu **walidacji**. Polega on na tym, że pewna liczba przypadków uczących jest zaliczana do oddzielnej grupy. Dane należące do tej wydzielonej grupy nie są bezpośrednio sto-

sowane w trakcie uczenia sieci (na przykład metodą wstecznej propagacji błędów). Natomiast są one wykorzystywane do przeprowadzenia niezależnej kontroli postępów algorytmu uczenia.

W każdym przypadku początkowa skuteczność sieci wyznaczona na podstawie ciągu uczącego i ciągu walidacyjnego jest taka sama (jest ona oczywiście bardzo kiepska, ponieważ przed rozpoczęciem uczenia sieć nie umie prawidłowo reagować na żadne dane – ani na te ze zbioru walidacyjnego, ani na dane uczące). Jeśli jakość odpowiedzi sieci na dane uczące i na dane walidacyjne nie jest przynajmniej w przybliżeniu identyczna, to najprawdopodobniej podział przypadków pomiędzy te dwa zbiory był obciążony jakąś ukrytą tendencją – zaleca się w takim przypadku przerwanie uczenia i ponowny (losowy) podział posiadanych danych na część uczącą i część walidacyjną.

W trakcie uczenia błąd popełniany przez sieć oczywiście się zmniejsza i, jeśli tylko proces uczenia minimalizuje prawidłowo zdefiniowaną funkcję błędu, to również zmniejsza się błąd walidacyjny. Na ogół po początkowym szybkim spadku obu rodzajów rozważnych błędów obserwujemy relatywnie szybszy spadek błędu dla zbioru uczącego i wolniejszy dla zbioru walidacyjnego. Jest to zjawisko normalne, ponieważ sieć poprawia swoje działanie wyłącznie w oparciu o dane ze zbioru uczącego, więc dopasowanie jej zachowania do tego właśnie zbioru **musi** być szybsze i dokładniejsze. Jeśli jednak w trakcie uczenia zaobserwujemy, że spadek błędu walidacyjnego zatrzymał się, lub też błąd ten zaczyna rosnąć, to świadczy to, że sieć zaczęła zbytnio dopasowywać się do danych uczących i traci zdolność do generalizacji wyników uczenia. W takim przypadku proces uczenia powinien zostać niezwłocznie zatrzymany, a nawet cofnięty do takiego punktu, w którym błąd walidacji miał najmniejszą wartość (istnieje taka możliwość w konfiguracji programu, aby uczenie było automatycznie zatrzymywane w chwili pojawienia się pierwszych przejawów przeuczenia). Przeuczenie sieci polega więc na pojawiającym się w trakcie uczenia nadmiernym dopasowaniu się sieci do danych uczących. W takim przypadku wskazane jest, aby zmniejszyć liczbę neuronów ukrytych i/lub liczbę warstw ukrytych, ponieważ pojawienie się przeuczenia sugeruje, że rozwiązując istniejący problem zastosowano sieć o zbyt dużych możliwościach. W sytuacji przeciwnej, gdy sieć nie posiada dostatecznych możliwości do modelowania rzeczywistej funkcji, przeuczenie się nie pojawi, ale wtedy mimo długiego uczenia ani błąd uczenia, ani błąd walidacyjny nie spadnie do satysfakcjonującego poziomu.

Problemy związane z występowaniem minimów lokalnych w funkcji błędu i problemy związane z koniecznością podjęcia decyzji dotyczącej wielkości sieci pociągają za sobą w praktyce konieczność przeprowadzenia szeregu eksperymentów z dużą liczbą sieci, z których każda jest wielokrotnie uczona (aby w ten sposób uniknąć fałszywego zatrzymania uczenia przez minimum lokalne). Każda z tych sieci jest oddzielnie uczona i niezależnie oceniana celem wyboru tej sieci, która może być uznana za optymalną. Najważniejszą informacją uwzględnianą przy ocenie sieci jest wartość błędu walidacyjnego.

Przy wyborze struktury sieci należy postępować zgodnie ze znaną w nauce regułą głoszącą, że w przypadku, gdy wszystkie inne własności dwóch modeli są identyczne, to należy wybrać model prostszy, odrzucając model o większej złożoności. To dążenie do prostoty jest w przypadku sieci neuronowych bardzo silnym imperatywem: nawet wówczas, gdy błąd walidacyjny mniejszej sieci jest nieznacznie większy od błędu uzyskanego dla sieci o większych rozmiarach, można wybrać model charakteryzujący się mniejszymi rozmiarami uzyskując (na ogół) lepsze wyniki podczas eksploatacji sieci.

Z podejściem opartym na powtarzaniu eksperymentów związany jest problem polegający na tym, że zbiór walidacyjny odgrywa kluczową funkcję przy wyborze modelu, co oznacza, że w rzeczywistości – w jakimś sensie – uczestniczy on w procesie uczenia sieci. W ten sposób wiarygodność zbioru walidacyjnego, jako **niezależnego** narzędzia oceny sieci, została podważona – gdyż przy dostatecznej liczbie eksperymentów można trafić na sieć, która będzie (przez przypadek) działać prawidłowo na zbiorze walidacyjnym, ale ogólnie nie będzie zadowalająca. Aby zwiększyć poziom zaufania do ostatecznego modelu zwykle (jeśli tylko wielkość zbioru uczącego na to pozwala) praktykuje się następujący sposób postępowania. Ze zbioru uczącego wydziela się dodatkowo **trzeci** zbiór przypadków – tak zwany ***zbiór testowy***. Ostateczna postać modelu (nauczonego przy pomocy zbioru uczącego i sprawdzonego przy pomocy zbioru walidacyjnego) jest dodatków **testowana** przy pomocy zbioru testowego. Takie dodatkowe sprawdzenie praktykuje się po to, aby upewnić się, że rezultaty uzyskane dla zbioru uczącego i zbioru walidacyjnego są zgodne z rzeczywistością, a nie są tylko mechanicznym wytworem procedury uczenia. Jest rzeczą oczywistą, że aby zbiór testowy mógł wypełnić tę rolę prawidłowo, to powinien być użyty tylko jeden raz – jeśli zostanie on ponownie użyty do uregulowania i ponownego przeprowadzenia procesu uczenia, to zacznie on spełniać faktycznie funkcję zbioru walidacyjnego i utracimy możliwość **niezależnej** weryfikacji uczenia.

Reasumując, można stwierdzić, że projektowanie sieci typu perceptronu wielowarstwowego składa się z następujących etapów (przy założeniu, że wcześniej dokonano wyboru zmiennych wejściowych):

- Wybór początkowej struktury sieci (zwykle jest to sieć z jedną warstwą ukrytą, w której początkowa liczba neuronów ukrytych jest równa połowie sumy liczby neuronów wejściowych i liczby neuronów wyjściowych).
- W sposób iteracyjny przeprowadza się szereg eksperymentów z każdą konfiguracją sieci i zachowuje się konfigurację najlepszą (w sensie błędu wyznaczonego dla ciągu walidacyjnego) ze znalezionych sieci. Nie trzeba się o to jakoś specjalnie troszczyć – w trakcie wykonywania eksperymentów program symulacyjny z reguły automatycznie zachowuje najlepszą sieć. Eksperymentów potrzebnych do znalezienia najlepszej sieci trzeba wykonać relatywnie dużo – z każdą możliwą strukturą sieci należy przeprowadzić kilka eksperymentów, aby uniknąć błędnych rezultatów spowodowanych zatrzymywaniem się algorytmu w minimum lokalnym.
- Jeśli wyniki eksperymentu świadczą o niedouczeniu sieci (sieć nie osiąga zadanego poziomu błędu) to należy podjąć próbę dodania nowych neuronów do warstwy ukrytej (warstw ukrytych). Jeśli to nie pomaga, to należy podjąć próbę dodania całej nowej warstwy ukrytej.
- Jeśli pojawi się zjawisko przeuczenia sieci (błąd walidacyjny zacznie znacząco rosnąć przed osiągnięciem zadowalającego poziomu wytrenowania sieci) to należy podjąć próbę usunięcia pewnej liczby neuronów ukrytych (lub całych ich warstw).

Ponieważ „ręczne" powtarzanie prób badawczych jest, łagodnie to określając, nudne, programy modelujące sieci neuronowe zawierają często algorytm automatycznego przeszukiwania, realizujący w sposób samodzielny cały ten proces. Ten moduł programu narzędziowego, określany czasem jako *Automatyczny projektant sieci* (*Automatic Network Designer*) sam przeprowadza eksperymenty z różną liczbą neuronów ukrytych, wielokrotnie powtarzając proces uczenia dla każdej badanej architektury, i całkowicie sam dokonuje wyboru najlepszego modelu sieci, kierując się wartością błędu walidacyjnego i wielkością sieci. W module tym zaimplementowane są zwykle różne, zarówno proste, jak i wyrafinowane algorytmy przeszukujące przestrzeń rozwiązań w poszukiwaniu minimum globalnego (na przykład algorytm symulowanego wyżarzania). Algorytmy te mogą automatycznie testować setki kombinacji struktur i parametrów sieci w poszukiwaniu szczególnie obiecującej sieci, która rozwiąże problem poszukiwania optymalnego modelu dla zgromadzonych danych. Algorytmy te również mogą znaleźć w bardzo krótkim czasie rozwiązanie przybliżone, pożyteczne wtedy, gdy znalezienie modelu dokładnego jest niewykonalne.

## 4.3. Sieci o radialnych funkcjach bazowych

W poprzedniej sekcji przedstawiono w jaki sposób zoptymalizować strukturę sieci typu perceptron wielowarstwowy, modelującą zadaną funkcję. W sieciach takich wymagane odpowiedzi sieci uzyskuje się korzystając ze złożenia funkcji typu „urwisko sigmoidalne" (Tadeusiewicz 1993). W przypadku problemów klasyfikacyjnych, odpowiada to podzieleniu przestrzeni sygnałów wejściowych (przestrzeni wzorców) przy pomocy hiperpłaszczyzn. Użycie hiperpłaszczyzn do dokonania podziału przestrzeni jest podejściem odwołującym się do intuicji i bazuje na zasadniczej prostocie linii jako struktury geometrycznej i formuły algebraicznej.

Równie pociągającym i intuicyjnym podejściem jest także podział przestrzeni przy użyciu okręgów lub (bardziej ogólnie) hipersfer. Hipersferę określa jej środek oraz promień. Uogólniając, można powiedzieć, że podobnie jak neuron w sieci MLP reaguje (nieliniowo) na odległość punktów od „sigmoidalnego urwiska", tak w sieci o radialnych funkcjach bazowych (tzw. sieci *RBF – Radial Basis Function network*) neurony reagują (nieliniowo) na odległość punktów od "centrum", które jest reprezentowane przez podlegające uczeniu parametry neuronu radialnego. Powierzchnia odpowiedzi pojedynczego neuronu radialnego ma charakter funkcji gaussowskiej (dzwonowej), o wierzchołku położonym nad centrum i o malejącej wartości funkcji wraz z oddalaniem się od tego punktu. Podobnie jak stromość występującej w perceptronie krzywej sigmoidalnej może być zmieniana, tak również można zmienić nachylenie funkcji gaussowskiej neuronu radialnego.

Neuron MLP jest definiowany przez swoje wagi i wartość progową, które razem dają równanie określonej prostej oraz określają tempo zmian wartości funkcji wraz z oddalaniem się od wyznaczonej prostej. Przed zastosowaniem sigmoidalnej funkcji aktywacji poziom pobudzenia neuronu definiuje hiperpłaszczyznę i z tego powodu w programie neurony takie są określane jako liniowe (chociaż funkcja aktywacji jest zazwyczaj nieliniowa). W przeciwieństwie do tej sytuacji neuron radialny jest zdefiniowany przez swoje centrum oraz parametr określany jako „promień". Punkt w przestrzeni $N$-wymiarowej jest definiowany przy użyciu $N$ liczb, co dokładnie odpowiada liczbie wag w neuronie liniowym, z tego powodu centrum neuronu radialnego jest przechowywane w zestawie parametrów określanych w programie również jako „wagi" (chociaż przy wyznaczaniu łącznego pobudzenia neuronów nie dokonuje się mnożenia składowych sygnałów przez te „wagi" tylko wyznaczana jest odległość wektora wag i wektora sygnałów wejściowych). Promień (lub inaczej odchylenie) jest przechowywa-

ny w neuronie jako tak zwana „wartość progowa". Warto podkreślić, że zarówno „wagi" jak i „wartość progowa" w neuronie radialnym są całkowicie czymś innym niż w neuronie liniowym, co powoduje, że stosowanie tej terminologii jest niebezpieczne, jeśli się nie pamięta o tym zróżnicowaniu. Wagi radialne definiują punkt, zaś radialna wartość progowa jest w rzeczywistości odchyleniem – ale nazwy stosowane są te same, co w neuronach perceptronów wielowarstwowych.

Sieć o radialnych funkcjach bazowych posiada zwykle jedną warstwę ukrytą, zawierającą neurony radialne, z których każdy modeluje gaussowską powierzchnię odpowiedzi (Rojas i in. 2000). Z uwagi na silnie nieliniowy charakter tych funkcji, zazwyczaj wystarcza jedna warstwa ukryta do zamodelowania funkcji o dowolnym kształcie. Warunkiem utworzenia przez sieć RBF skutecznego modelu dowolnej funkcji jest jednak zapewnienie w strukturze sieci dostatecznej liczby neuronów radialnych. Jeśli jest ich wystarczająco dużo, można do każdego istotnego szczegółu modelowanej funkcji przywiązać odpowiedni neuron radialny, co gwarantuje, że uzyskane rozwiązanie będzie odwzorowywało zadaną funkcję z całkowicie satysfakcjonującą wiernością.

Pozostaje jeszcze pytanie, w jaki sposób należy połączyć wyjścia ukrytych neuronów radialnych aby uzyskać wymaganą wartość wyjściową sieci. Okazuje się, że wystarczy po prostu zastosować kombinację liniową tych wartości wyjściowych (tzn. ważoną sumę wartości funkcji gaussowskich). Sieć RBF posiada więc warstwę wyjściową zawierającą neurony liniowe z liniowa funkcją aktywacji.

Sieci RBF posiadają kilka zalet w porównaniu z sieciami typu MLP. Po pierwsze, jak już wcześniej stwierdzono, mogą one modelować dowolną funkcję nieliniową przy pomocy **pojedynczej** warstwy ukrytej, przez co eliminuje się konieczność podejmowania na etapie projektowania decyzji dotyczącej liczby warstw. Po drugie, prosta transformacja liniowa dokonywana w warstwie wyjściowej może być w całości zoptymalizowana przy użyciu tradycyjnych technik modelowania liniowego, które są szybkie i przy stosowaniu których nie pojawiają się takie problemy jak minima lokalne, które są plagą występującą w uczeniu sieci MLP. Z tego powodu sieci RBF mogą być uczone w bardzo krótkim czasie (różnica w szybkości uczenia dotyczy rzędów wielkości).

Z drugiej jednak strony, przed zastosowaniem optymalizacji liniowej w odniesieniu do warstwy wyjściowej sieci RBF należy zadecydować o wartościach parametrów dużej liczby neuronów radialnych, poprzez określenie dla każdego z nich wartości centrów i odchyleń. Chociaż stosowane do tego celu algorytmy są znacznie szybsze niż te, które stosuje się do uczenia sieci MLP, ale są one nie mniej skłonne do „odkrywania" kombinacji suboptymalnych (co jest odpowiednikiem minimów lokalnych w sieci MLP).

Inną cechą różnicującą działanie sieci RBF od sieci MLP jest inne podejście do modelowania przestrzeni. Model uzyskiwany w przypadku sieci RBF można określić jako „skupieniowy" natomiast model uzyskiwany w przypadku sieci MLP można określić jako „płaszczyznowy". Ta odmienność modeli ma swoje praktyczne konsekwencje. Eksperymenty pokazują, że bardziej wyrafinowane kształty modelowanych funkcji i wyższe oczekiwania odnośnie dokładności odpowiedzi wymagają zastosowania w sieciach RBF większej liczby neuronów. Oznacza to, że sieć RBF jest kosztowniejsza od sieci MLP i wymaga większej liczby neuronów w celu właściwego zamodelowania większości badanych funkcji. Oczywiście, zawsze można znaleźć takie kształty funkcji, które w prostszy sposób są reprezentowane przy pomocy jednego lub drugiego rodzaju sieci, ale ogólny bilans zdecydowanie nie preferuje sieci RBF. Wskutek tego, rozwiązania oparte na sieciach RBF będą miały skłonność do wolniejszego działania i będą wymagały większych obszarów pamięci niż odpowiadające im sieci MLP. Ale z drugiej strony trzeba przypomnieć, że sieci RBF wymagają krótszego uczenia, co czasami może być istotniejszym ograniczeniem.

Podejście „skupieniowe" powoduje również, że sieci RBF nie mają skłonności do ekstrapolacji modelowanych zależności poza obszarem wyznaczonym znanymi danymi. Jeśli wprowadzone dane testowe są istotnie oddalone od danych uczących to odpowiedź sieci obniża się gwałtownie do zera. W przeciwieństwie do tego, sieć MLP jest bardziej zdecydowana w swojej odpowiedzi nawet w przypadku użycia danych testowych daleko oddalonych od danych uczących. Czy cecha ta zostanie uznana za wadę, czy za zaletę, zależy od konkretnej aplikacji, ale, biorąc pod uwagę wszystkie elementy, realizowana przez MLP bezkrytyczna ekstrapolacja jest zwykle uważana za cechę negatywną. Przyjmuje się bowiem, że ekstrapolacja modelowanej funkcji daleko poza dane uczące jest zwykle niebezpieczna i nieuprawniona. Sieci RBF są również bardziej wrażliwe na „problem wymiarowości" i mają większe kłopoty jeśli liczba neuronów wejściowych jest duża.

### 4.4. Probabilistyczne sieci neuronowe

W przypadku problemów klasyfikacyjnych użyteczną interpretacją wartości wyjściowych sieci jest ich traktowanie jako **prawdopodobieństw** przynależności do klas. W takiej sytuacji sieć w trakcie uczenia w rzeczywistości uczy się estymować funkcję gęstości prawdopodobieństwa reprezentowaną przez zgro-

madzone dane. Podobną przydatną interpretację probabilistyczną można przyjąć w problemach regresyjnych, przy założeniu, że wartość wyjściowa sieci traktowana jest jako **wartość oczekiwana** modelu w danym punkcie przestrzeni wejściowej. Ta wartość oczekiwana jest związana z łączną funkcją gęstości prawdopodobieństwa – zarówno wyjścia jak i wszystkich wejść.

Estymacja funkcji gęstości prawdopodobieństwa na podstawie danych ma w statystyce długą historię. Rozważania na ten temat najczęściej prowadzone są na gruncie statystyki bayesowskiej. Statystyka konwencjonalna, mając znany model, może zazwyczaj dostarczyć informacji na temat tego, **jaka jest szansa** na zaistnienie pewnej wartości wyjściowej. Na przykład, wiemy, że szansa na pojawienie się wartości *sześć* na nieobciążonej kostce jest równa 1/6. Statystyka bayesowska formułuje odwrotne zadania, stawiając niejako całą sytuację „na głowie": badacz przy tym podejściu z reguły estymuje przydatność modelu w oparciu o posiadane dane. Uogólniając to stwierdzenie w jeszcze większym stopniu można stwierdzić, że statystyka bayesowska umożliwia estymację funkcji gęstości prawdopodobieństwa parametrów modelu przy wykorzystaniu dostępnych danych. W wyniku analizy bayesowskiej wybierany jest ten model, którego parametry maksymalizują wspomnianą funkcję gęstości prawdopodobieństwa, co pozwala zmniejszyć globalny błąd.

W kontekście zagadnień klasyfikacyjnych, jeśli można wyznaczyć estymatory funkcji gęstości prawdopodobieństwa parametrów (zmiennych wejściowych) dla możliwych klas, to można porównać wynikające z nich prawdopodobieństwa *a'posteriori* (prawdopodobieństwa różnych klas) i wybrać jako rozwiązanie problemu klasę najbardziej prawdopodobną.

Przy zaakceptowaniu tego podejścia nauka rozwiązywania przez sieć problemu klasyfikacyjnego polega w rzeczywistości na próbie nauczenia się przez nią aproksymacji funkcji gęstości prawdopodobieństwa (jest to z reguły najtrudniejsza część zadania), resztę procesu rozpoznawania może realizować prosty system postprocessingu danych wyjściowych.

Opisany wyżej sposób rozwiązywania sformułowanego zadania przy wykorzystaniu sieci neuronowych jest stosowany od niedawna. Bardziej tradycyjne podejście polegało na konstrukcji estymatora funkcji gęstości prawdopodobieństwa na podstawie danych. Technika ta polega na arbitralnym przyjęciu pewnej postaci funkcji gęstości prawdopodobieństwa (zwykle zakłada się, że jest to rozkład normalny) i oszacowaniu parametrów modelu. Rozkład normalny jest przy tym najczęściej używany między innymi z tego powodu, że umożliwia on oszacowanie parametrów modelu (którymi są: wartość średnia i odchylenie standardowe) przy użyciu technik analitycznych. Problemem który przy takim podejściu najtrudniej jest rozstrzygnąć zagadnienie, czy założenie o normalności rozkładu danych jest uprawomocnione.

Alternatywne podejście do estymacji funkcji gęstości prawdopodobieństwa jest oparte na tak zwanej aproksymacji. Działanie wspomnianej metody można w sposób uproszczony opisać w ten sposób, że obecność pewnego przypadku w pewnym punkcie przestrzeni wejściowej oznacza dużą gęstość prawdopodobieństwa w tym punkcie. Skupienie przypadków znajdujących się blisko siebie wskazuje na obszar o wysokiej gęstości prawdopodobieństwa. Obszary odległe od jakichkolwiek znanych przypadków charakteryzuje się natomiast gęstością prawdopodobieństwa malejącą do zera. Można to zinterpretować następująco: Zbliżając się (w przestrzeni sygnałów wejściowych) do miejsca lokalizacji jednego z przypadków ciągu uczącego można mieć zaufanie do rosnącej gęstości prawdopodobieństwa, natomiast w większej odległości od jakiegokolwiek znanego przypadku poziom zaufania jest zdecydowanie mniejszy i zmniejsza się w miarę oddalania. W estymacji jądrowej, proste funkcje (tak zwane „jądrowe") są lokowane w miejscu wystąpienia każdego dostępnego przypadku, a następnie są one dodawane w celu uzyskania estymatora łącznej funkcji gęstości prawdopodobieństwa. W typowych przypadkach, każda funkcja jądrowa jest funkcją gaussowską (dzwonową). Jeśli dostępna jest dostateczna liczba punktów uczących, to istotnie daje to stosunkowo dobrą aproksymację rzeczywistej funkcji gęstości prawdopodobieństwa. Jeśli jednak punktów jest niewiele, a wymiar wejściowej przestrzeni jest duży – metoda ta daje raczej mało satysfakcjonujące wyniki.

Jądrowe podejście do aproksymacji funkcji gęstości prawdopodobieństwa jest bardzo podobne do stosowania sieci neuronowych o radialnych funkcjach bazowych, co stanowiło inspirację do stworzenia kategorii probabilistycznych sieci neuronowych (**PNN** – *probabilistic neural networks*) oraz sieci neuronowych realizujących regresję uogólnioną (**GRNN** – *generalized regression neural networks*).

Sieci PNN zostały zaprojektowane z myślą o rozwiązywaniu problemów klasyfikacyjnych, zaś sieci GRNN służą do rozwiązywania problemów regresyjnych. Oba te rodzaje sieci są w rzeczywistości modelami aproksymacji jądrowej przedstawionymi w postaci sieci neuronowej.

W sieci PNN występują przynajmniej trzy warstwy: wejściowa, radialna i wyjściowa. Neurony radialne mają parametry kopiowane bezpośrednio z danych uczących; każdy z nich odpowiada jednemu przypadkowi. Wygląda to w taki sposób, że każdy z neuronów

radialnych modeluje funkcję Gaussa wycentrowaną nad „swoim" przypadkiem uczącym. Natomiast w warstwie wyjściowej każdej klasie odpowiada jeden neuron. Do każdego z tych neuronów wyjściowych docierają połączenia od tych neuronów radialnych, które zostały ustawione nad punktami (zestawami danych wejściowych) należącymi do danej klasy; nie występują natomiast połączenia neuronów wyjściowych z innymi neuronami radialnymi. Neurony wyjściowe sumują więc po prostu wartości wyjściowe pojawiające się na wyjściach neuronów radialnych należących do klasy odpowiadającej danemu neuronowi wyjściowemu. Wartości wyjściowe neuronów wyjściowych są więc proporcjonalne do estymatorów jądrowych funkcji gęstości prawdopodobieństwa dla różnych klas, i po zastosowaniu normalizacji zapewniającej ich sumowanie do jedności stanowią wprost oszacowania prawdopodobieństwa przynależności do poszczególnych klas.

Podstawowy model sieci PNN może być zmodyfikowany na dwa sposoby.

- Pierwszy sposób polega na tym, że zmienia się sposób traktowania udziału reprezentacji poszczególnych klas w zbiorze uczącym. Podstawowe podejście zakłada, że udział przedstawicieli poszczególnych klas w zbiorze uczącym jest zgodny z rzeczywistym odsetkiem przypadków zaliczanych do tej klasy w modelowanej populacji (są to tak zwane prawdopodobieństwa a priori). Na przykład, w przypadku sieci diagnozującej pewne uszkodzenie walcarki, jeśli uszkodzenie to występuje jako przyczyna 2% awarii, to w zbiorze uczącym mniej więcej 2% przypadków powinno dotyczyć tego właśnie uszkodzenia. Jeśli prawdopodobieństwo a priori różni się od udziału przypadków należących do danej klasy w ciągu uczącym, to oszacowania rozkładów wyznaczane przez sieć będą także nieprawidłowe. Tymczasem w zbiorze danych o małej (z reguły) liczebności bardzo trudno zachować warunek dokładnej reprezentacji poszczególnych klas w takich liczbach egzemplarzy, które dokładnie odwzorowują prawdopodobieństwa a priori. Przy niewielkiej sumarycznej liczbie obiektów uczących i przy niewielkich prawdopodobieństwach a priori pewnych klas – może się to okazać wręcz niewykonalne. W celu zniwelowania wpływu takich problemów i związanych z nimi dysproporcji – można w sieci jawnie wyspecyfikować prawdopodobieństwa a priori (jeśli są one znane), co spowoduje zmianę wartości wag neuronów wyjściowych sieci dla ich wejść odpowiadających neuronom ukrytym poszczególnych klas. Taką korektę wag, prowadzącą do wyrównania istniejących różnic, można w sieci w miarę łatwo wprowadzić, ale wymaga to naruszenia „klasycznych" zasad uczenia sieci i zakłada wprowadzenie „ręcznej" modyfikacji parametrów (wag) sieci.
- Drugi sposób zmodyfikowania modelu sieci PNN może polegać na odmiennym traktowaniu różnych rodzajów błędów pojawiających się w trakcie uczenia i eksploatacji sieci. Ogólnie wiadomo, że sieć dokonująca estymacji nieznanego rozkładu gęstości prawdopodobieństwa w oparciu o dane zniekształcone przez szumy odtworzy potrzebną funkcję z błędami, w wyniku czego będzie nieuchronnie klasyfikować podawane jej dane w pewnych przypadkach błędnie. Skutek takich błędów jest oczywiście zawsze zdecydowanie niekorzystny, bo jest zawsze rzeczą naganną jeśli na przykład osoba w rzeczywistości chora zostanie uznana za zdrową. Jednakże pewne rodzaje błędnej klasyfikacji mogą być uznawane za „bardziej kosztowne" niż inne. Na przykład, uznanie maszyny sprawnej za zagrożoną awarią spowoduje jej chwilowe wyłączenie z ruchu i zbyteczny przegląd – ale nie zagraża żadnym bezpośrednim niebezpieczeństwem. Natomiast niepowodzenie w identyfikacji rzeczywiście występującej groźnej niesprawności może doprowadzić do zaniechania przeglądu i naprawy w sytuacji, kiedy są one rzeczywiście niezbędnie potrzebne i w efekcie może prowadzić do powstania wypadku, którego skutki są bardzo poważne i niebezpieczne.

W takich przypadkach, gdy zachodzi potrzeba zróżnicowania „cen błędów" można wprowadzić do programu procedurę „ważenia" surowych prawdopodobieństw wyznaczonych przez sieć. Dokonuje się tego poprzez zastosowanie w programie specjalnie ustalanych „czynników straty", które odzwierciedlają koszty błędnej klasyfikacji – ewentualnie różne w przypadku różnych możliwych typów błędów. W sieci PNN można w tym celu zdefiniować formalnie dodatkową, czwartą warstwę sieci, która zawiera macierz zadawanych przez użytkownika strat wiązanych z poszczególnymi rodzajami błędów. Macierz ta jest przemnażana przez prawdopodobieństwa oszacowane w warstwie trzeciej, a następnie wybierana jest klasa charakteryzująca się najmniejszym oszacowanym kosztem, co pozwala na podejmowanie decyzji z uwzględnieniem zróżnicowanych cen ewentualnych błędów, a nie tylko opartych na samym tylko rozkładzie prawdopodobieństw a posteriori.

Na marginesie tych rozważań warto wspomnieć, że omówiona wyżej macierz strat, typowo rozważana w kontekście bayesowskich metod rozpoznawania i podejmowania decyzji, może również być w programie dołączona do innych typów sieci klasyfikujących,

co stwarza bardzo ciekawe możliwości wzbogacania ich działania o czynnik różnicowania cen różnych decyzji.

Jedynym parametrem sterującym, wpływającym na proces uczenia sieci typu PNN, którego wartość musi być ustalona przez użytkownika, jest współczynnik wygładzania (*smoothing factor*). Współczynnik ten, reprezentujący odchylenie radialne odpowiednich funkcji gaussowskich, jest miarą zasięgu oddziaływania „wiedzy" zawartej w przypadkach tworzących ciąg uczący na otaczające obszary przestrzeni sygnałów wejściowych. Podobnie jak w sieciach RBF współczynnik ten musi być określony w taki sposób, aby funkcje gaussowskie zachodziły na siebie w "rozsądnym stopniu" – zbyt małe odchylenie powoduje, że aproksymacja jest bardzo szpiczasta, co uniemożliwia generalizację, zaś zbyt duża wartość tego współczynnika uniemożliwia prawidłowy opis szczegółów aproksymowanej funkcji rozkładu prawdopodobieństwa. Właściwa wartość współczynnika wygładzania (*smoothing factor*) może zostać w prosty sposób określona na drodze eksperymentalnej, poprzez wybór takiego współczynnika, który generuje akceptowalnie mały błąd walidacyjny. Na szczęście doświadczenie wskazuje, że sieci PNN nie są zbyt wrażliwe na wartość współczynnika skalowania i ewentualnie nietrafny wybór tej wartości nie zmniejsza (w znaczącym stopniu) szansy uzyskania poprawnego działania całej sieci.

Do największych zalet sieci PNN należy zaliczyć generowanie na wyjściach wartości prawdopodobieństw (a dokładniej ich oszacowań), a nie samych tylko „surowych" decyzji, co zdecydowanie ułatwia ocenę i interpretację wyników. Zaletą omawianych sieci jest też duża szybkość ich uczenia. Na proces uczenia tych sieci składa się bowiem w rzeczywistości wyłącznie kopiowanie przypadków uczących do odpowiednich neuronów sieci, co zgodnie z oczekiwaniami trwa bardzo krótko.

Największą wadą sieci typu PNN jest ich wielkość. Sieć tego typu musi bowiem zawierać w swojej strukturze neurony odpowiadające poszczególnym rozważanym przykładom, co powoduje, że w rzeczywistości w strukturze sieci odwzorowany jest cały zbiór uczący. Powoduje to z kolei, że wymagania odnośnie pamięci przy tworzeniu i eksploatacji takich sieci są bardzo duże oraz jest to przyczyną, że czas potrzebny na uruchomienie takiej sieci jest raczej długi.

Sieci PNN są szczególnie użyteczne w trakcie eksperymentów mających na celu zdefiniowanie prototypów sieci (na przykład wtedy, gdy podejmowane są decyzje dotyczące wyboru zmiennych wejściowych), ponieważ krótki czas uczenia tych sieci umożliwia wykonanie dużej liczby testów (na przykład z różnymi zestawami danych wejściowych) w krótkim przedziale czasu.

## 4.5. Sieci neuronowe realizujące regresję uogólnioną

Sieci neuronowe realizujące regresję uogólnioną (sieci *GRNN – Generalized regression neural networks*) pracują w sposób podobny jak sieci PNN, ale służą raczej do rozwiązywania zadań o charakterze regresyjnym, a nie klasyfikacyjnym. Podobnie jak w sieciach PNN, gaussowskie funkcje jądrowe są lokowane w tej sieci w poszczególnych neuronach warstwy ukrytej w taki sposób, że dla każdego przypadku uczącego jest dostępny neuron, który „rozpina" ponad tym przypadkiem „dzwon" odpowiedniej funkcji gaussowskiej.

Metoda budowy przez sieć potrzebnej funkcji regresji polega na tym, że każdy przypadek zbioru uczącego może być rozważany jako „dowód" tego, że powierzchnia budowanej odpowiedzi sieci ma w tym punkcie przestrzeni wejść pewną ustaloną wysokość. Ten lokalny (punktowy) dowód jest następnie przez sieć rozmywany na okoliczne punkty przestrzeni sygnałów wejściowych, z progresywnie zmniejszającą się pewnością występowania podanej wartości – maksymalną w bezpośrednim sąsiedztwie punktu o lokalizacji wynikającej z ciągu uczącego i szybko malejącą przy oddalaniu się od niego.

Działanie sieci GRNN polega na tym, że przypadki uczące, przekopiowane do neuronów warstwy ukrytej sieci, służą do estymacji odpowiedzi sieci zarówno dla samych punktów zbioru uczącego, jak i dla nowych punktów, które w zbiorze uczącym wcale nie występowały. Wartość wyjściowa dla tych nowych punktów jest szacowana przez sieć przy wykorzystaniu ważonej średniej wyjść dla przypadków uczących, gdzie wagi są uzależnione od odległości poszczególnych punktów uczących od punktu, dla którego przeprowadza się szacowanie. Z tego względu punkty zawarte w ciągu uczącym, położone blisko punktu, w którym określana jest wartość funkcji mocniej wpływają na szacowaną wartość, niż punkty położone dalej.

Pierwsza warstwa ukryta w sieci GRNN zawiera, jak już powiedziano, same tylko neurony radialne. Druga warstwa ukryta zawiera neurony pomagające oszacować średnią ważoną z wyjść neuronów poprzedniej warstwy. Stosowana jest przy tym specjalna procedura. Każdy neuron wyjściowy posiada w tej warstwie swojego odpowiednika, który wyznacza ważoną sumę dla odpowiadającego wyjścia. Aby uzyskać ważoną średnią z ważonej sumy, ważona suma musi zostać podzielona przez sumę współczynników wagowych. Brzmi to w sposób skomplikowany, ale zasadniczy pomysł jest w miarę prosty. Pojedynczy, specjalizowany neuron w drugiej warstwie ukrytej

wyznacza wartość ważonej sumy swoich wejść. Z kolei warstwa wyjściowa realizuje w rzeczywistości operację dzielenia (korzystając ze specjalizowanych neuronów realizujących operację dzielenia) w celu dokonania wymaganej normalizacji. Wynika z tego, że druga warstwa ukryta posiada zawsze dokładnie o jeden neuron więcej niż warstwa wyjściowa. W typowych problemach regresyjnych, szacowana jest najczęściej jedna wartość wyjściowa, co powoduje, że druga warstwa ukryta posiada zwykle dwa neurony.

Sieć GRNN może zostać zmodyfikowana poprzez zastosowanie neuronów radialnych, które reprezentują całe skupienia (grupy, ang. *cluster*) występujące w danych, a nie poszczególne odrębne przypadki uczące. W ten sposób (poprzez zastosowanie grupowania danych wejściowych) zmniejszany jest rozmiar sieci i zwiększana jest szybkość jej działania. Centra skupień mogą być wyznaczone przy użyciu odpowiednich algorytmów (w rachubę wchodzą algorytmy powtórnego próbkowania, k-średnich lub algorytmy angażujące oddzielne sieci neuronowe, mianowicie sieci Kohonena), zaś program jest tak zbudowany, że zmodyfikuje automatycznie w odpowiedni sposób wagi po wyznaczeniu centrów odpowiednich skupień.

Sieć GRNN posiada zalety i wady w większości podobne do odpowiednich cech sieci PNN – jedyna znacząca różnica polega na tym, że sieć GRNN może być używana tylko do problemów regresyjnych, podczas gdy sieć PNN używana jest wyłącznie do problemów klasyfikacyjnych. Sieć GRNN uczy się w bardzo krótkim czasie, ale posiada tendencję do tworzenia struktur o dużych rozmiarach i do związanego z tym powolnego działania (chociaż, inaczej niż w sieciach PNN, nie jest tu konieczne posiadanie jednego neuronu radialnego przypadającego na każdy przypadek uczący). Wynika to głównie z faktu, że liczba neuronów ukrytych musi być w tego typu sieci sumarycznie bardzo duża. Podobnie jak sieć RBF, sieć GRNN nie posiada zdolności do ekstrapolacji danych – na ogół zaraz poza zakresem zmienności danych wejściowych występujących w zbiorze uczącym sygnały wyjściowe sieci przyjmują wartości bardzo odbiegające od wartości prawidłowych (najczęściej jest to wartość wynosząca zero).

## 5. SIECI KLASYFIKACYJNE I REGRESYJNE

Przegląd zestawionych w rozdziale 2 zastosowań sieci neuronowych w metalurgii pozwala stwierdzić, że zadania, jakie pełnią sieci neuronowe dają się podzielić na zadania klasyfikacyjne i zadania regresyjne (aproksymacyjne). Omówimy je teraz krótko, charakteryzując występujące w nich problemy.

### 5.1. Sieci klasyfikacyjne

W zagadnieniach klasyfikacyjnych celem stawianym sieci jest przypisanie każdego przypadku reprezentowanego przez odpowiedni zestaw danych wejściowych do jednej z wybranych klas. Mówiąc bardziej ogólnie można zakładać, że zadaniem sieci jest estymacja prawdopodobieństwa przynależności przypadku do danej klasy. Klasyfikacja jest realizowana poprzez użycie nominalnej zmiennej wyjściowej, której wartości (będące nazwami wybieranymi z pewnego góry ustalonego zbioru wartości) odpowiadają różnym klasom, do których można zaliczać wejściowe dane.

Klasyfikacja może być zrealizowana przy użyciu następujących typów sieci: MLP, RBF, sieci Kohonena (nie omawianych tu), PNN oraz sieci liniowych. Jedynym typem sieci, który **nie** jest zaprojektowany z myślą o klasyfikacji jest GRNN, chociaż oczywiście można próbować zastosować do klasyfikacji także sieci tego typu (ale podejście takie nie jest polecane).

Na ogół nie ma problemów z określenie wartości wejściowych do sieci klsyfikacyjnej – i to niezależnie od tego, czy mamy do czynienia z uczeniem, czy z egzaminowaniem sieci. Natomiast przy określaniu wartości odpowiadających nominalnym zmiennym wyjściowym sytuacja przy uczeniu i przy egzaminie jest odmienna. Przy uczeniu dobrze wiemy, do jakiej klasy obiekt należy, odwzorowanie tego faktu nie stanowi więc również problemu w trakcie uczenia sieci. Jednakże w trakcie eksploatacji sieci określenie klasy wyjściowej wyznaczonej przez sieć wymaga na ogół więcej wysiłku. Musimy ten problem dokładniej rozważyć, żeby nie mieć kłopotów z interpretacją zachowania sieci w trakcie jej działania.

Każdy z neuronów wyjściowych może generować **ciągłe** wartości z przedziału od 0,0 do 1,0. W celu precyzyjnego wyznaczania odpowiedzi sieci (to znaczy klasy, do której należy zaliczyć rozważany obiekt) trzeba na podstawie takich wartości wyjściowych ustalić, jaką wartość powinniśmy przyporządkować wyjściowej zmiennej nominalnej. W tym celu sieć musi zadecydować, czy wartości wyjściowe są „w sensowny sposób" bliskie wartości 0.0 czy też bliższe są wartości 1.0. Jeśli wyjście pewnego neuronu nie jest bliskie żadnej z tych wartości, to klasa wskazywana (rozpoznawana) przez sieć jest „nieokreślona".

Aby podjąć decyzję dotyczącą sposobu interpretacji wyjść sieci klasyfikującej dane korzysta się zwykle z zadawanych przez użytkownika wartości progowych, czyli tak zwanych **poziomów ufności.** Wyróżniamy dwa takie poziomy: **próg akceptacji** oraz **próg odrzucenia**. Interpretacja tych progów jest odmienna dla reprezentacji dwustanowej i reprezentacji *jeden-z-N*. Przedstawimy teraz te obie interpretacje, pomimo że różnią się one od siebie nieznacznie:

- **Reprezentacja dwustanowa**. Jeśli wartość wyjściowa neuronu jest *powyżej progu akceptacji*, to przyjmuje się jako odpowiedź sieci klasę przypisaną do wartości 1,0. Jeśli wartość wyjściowa neuronu jest *poniżej poziomu odrzucenia*, to jako sygnał wyjściowy sieci wybierana jest klasa odpowiadająca wartości 0.0. Jeśli natomiast wartość wyjściowa jest *pomiędzy* dwoma wymienionymi wartościami progowymi to klasa jest nieokreślona (sieć odpowiada „nie wiem").
- **Reprezentacja *jeden-z-N***. Konkretna klasa jest wybierana (jako odpowiedź całej sieci) wówczas, gdy wartość wyjściowa neuronu odpowiadającego tej właśnie konkretnej klasie jest *powyżej poziomu akceptacji* i równocześnie wartości wyjściowe *wszystkich* pozostałych neuronów są *poniżej poziomu odrzucenia*. Jeśli podany warunek (często trudny do spełnienia!) nie występuje, to klasa jest nieokreślona (sieć znowu odpowiada „nie wiem").

Omówione wyżej wartości progowe mogą być swobodnie modyfikowane przez użytkownika w celu spowodowania, aby jego sieć była bardzie lub mniej „drobiazgowa" przy określaniu wyniku procesu klasyfikacji. Istnieje jednak pewna osobliwość ujawniająca się w trakcie korzystania z kodowania *jeden-z-N,* na którą teraz zwrócimy uwagę. W trakcie pobieżnego czytania przytoczonych wyżej wywodów Czytelnik mógłby oczekiwać, że „najmniej drobiazgową" (najrzadziej odmawiającą odpowiedzi) będzie taka sieć, która posiadać będzie najbardziej liberalny poziom akceptacji (wynoszący 0,5) i najmniej rygorystyczny poziom odrzucenia równy także 0,5. W rzeczywistości w przypadku takiej sieci z kodowaniem *jeden-z-N* sytuacja wygląda inaczej – sieć o parametrach pozornie bardzo ułatwiających podjęcie jednoznacznej decyzji nadal często odmawia jednoznacznej klasyfikacji pokazywanych jej danych! Co ciekawe – powyższe stwierdzenie sprawdza się także w przypadku sieci z kodowaniem dwustanowym.

Aby osiągnąć pełną „kooperatywność" sieci (to znaczy aby zmusić ją do jednoznacznej klasyfikacji w każdym przypadku) należy określić poziom akceptacji – paradoksalnie – **niższy** niż poziom odrzucenia. Oznacza to, że „najmniej drobiazgowa" sieć posiada poziom akceptacji równy 0,0 i poziom odrzucenia równy 1,0. Te na pozór nonsensowne ustawienia parametrów sieci pociągają za sobą następujący sposób jej działania: klasę wskazywaną jako odpowiedź sieci określa neuron o **najwyższej** wartości wyjściowej. Ten neuron „zwycięski" określa klasę będącą rozwiązaniem (odpowiedzią sieci) **niezależnie** od wartości sygnałów wyjściowych wszystkich pozostałych neuronów sieci. Taki sposób postępowania gwarantuje, że sieć zawsze poda jednoznaczną odpowiedź – chociaż czasem jej wartość (użyteczność) może być problematyczna...

W powyższej dyskusji przyjęto założenie, że "pozytywna" klasyfikacja jest określana przez liczbę bliską 1,0, zaś „negatywna" klasyfikacja przez liczbę bliską 0,0. Inne wartości sygnałów wyjściowych nie są wykorzystywane i dlatego nie powinny występować. Warunek ten jest prawdziwy wtedy, gdy w neuronach wyjściowych korzysta się z binarnej funkcji aktywacji, to znaczy takiej funkcji, w której występują tylko te właśnie dwie wyróżnione wartości sygnału. Jeśli chcemy korzystać także z wartości sygnałów wyjściowych przyjmujących dowolne pośrednie wartości pomiędzy granicznymi wartościami 0,0 i 1,0 – to powinniśmy skorzystać z tak zwanej *sigmoidy*, czyli mówiąc bardziej poprawnie z *logistycznej funkcji aktywacji.*

Przyjęte wyżej założenie odnośnie do zakresu wartości przyjmowanych przez zmienne wyjściowe (0,0 do 1,0) ma tę zdecydowaną zaletę, że w opisywanym przypadku zakres występujących w sieci sygnałów jest w naturalny zgodny z zakresem przyjmowanym przez prawdopodobieństwo (tzn. także od 0,0 do 1,0), co ułatwia interpretację niektórych form działania sieci. Jest to jeden z wielu powodów, przemawiających za powszechnym stosowaniem w sieciach neuronowych neuronów o sigmoidalnych charakterystykach, produkujących sygnały o wartościach pochodzących właśnie z przedziału od 0,0 do 1,0.

Jednakże w pewnych okolicznościach bardziej dogodne może być użycie innego zakresu sygnałów wyjściowych. Na przykład w wielu zastosowaniach zdecydowanie korzystne jest stosowanie funkcji **bipolarnej**, to znaczy takiej, w której wartości sygnałów wyjściowych mogą zmieniać się w przedziale od -1,0 do +1,0. Przy użyciu tej funkcji rezultaty uczenia mogą być znacząco ulepszone ponieważ, funkcja bipolarna (w przeciwieństwie do unipolarnej funkcji logistycznej) jest symetryczna względem zera, co powoduje, że wartość sygnału podawanego do dalszych elementów sieci rzadko przyjmuje wartość zero. Przy funkcji unipolarnej wartość 0,0 może pojawić na wyjściu sieci z prawdopodobieństwem zbliżonym do 50%, co powoduje, że wartości współczynników wagowych neuronów do których sygnały są wysyłane nie mają wpływu na zachowanie sieci (w końcu sygnał wynoszący zero mnożony przez **jakąkolwiek** wartość wagi synapsy daje zawsze zero), a przecież w następstwie procesu uczenia właśnie we współczynnikach wagowych zgromadzona jest cała wiedza sieci. W związku z tym używanie bipolarnej charakterystyki neuronu ma istotna wyższość nad stosowaniem charakterystyki unipolarnej, ponieważ przy charakterystyce bipolarnej w blisko 50% przypadków sygnały wyjściowe będą miały wartość zbliżoną do +1, również w blisko 50% przypadków sygnał będzie przyjmował wartość -1, zaś pośrednie przypadki (w tym w szczególności sytuacja, że sygnał ma wartość 0,0) będą się zdarzały relatyw-

nie rzadko. Taka sytuacja jest zdecydowanie korzystna z punktu widzenia ekspresji wiedzy zawartej w sieci.

Czasami uporządkowanie wartości sygnałów wyjściowych może być (przy interpretacji działania sieci) korzystnie zmienione na odwrotne, na przykład wtedy, kiedy mniejsze wartości oznaczają pozytywną klasyfikację. Okolicznością wskazującą na celowość korzystania z odwróconego uporządkowania wartości jest użycie opisanej wyżej macierzy kosztów związanych z poszczególnymi decyzjami. Jak pamiętamy, macierz taka może być dodana do sieci PNN na etapie jej tworzenia, lub też może zostać dołączona ręcznie do innych typów sieci. Kiedy korzysta się z macierzy kosztów, wyjścia sieci określają oczekiwany koszt związany z wyborem każdej z klas, zaś celem jest wybór klasy o **najniższym** koszcie.

## 5.2. Sieci regresyjne

W poprzednim punkcie dyskutowano optymalizację struktury sieci klasyfikacyjnych, czyli takich, które na wyjściu produkują **decyzje.** Możliwe jest jednak również tworzenie sieci dla rozwiązywania problemów regresyjnych. Jak już wcześniej sygnalizowano, w problemach regresyjnych celem jest oszacowanie **wartości** ciągłej zmiennej wyjściowej, w sytuacji gdy znane są wartości zmiennych wejściowych. Problemy regresyjne w programie mogą być rozwiązywane przy użyciu następujących typów sieci: MLP, RBF, GRNN oraz sieci liniowych.

Problemy regresyjne reprezentowane są przez zbiory danych, w których zmienna wyjściowa (lub zmienne wyjściowe, bo sieć może mieć kilka wyjść) nie jest zmienną nominalną, lecz przeciwnie – ma charakter numeryczny. W związku z tym do szczególnie istotnych zagadnień w problemach regresyjnych należą: **skalowanie** wartości wyjściowych oraz skutki stosowania **ekstrapolacji**. Omówimy teraz kolejno oba wskazane zagadnienia.

Jak wiadomo, najpopularniejsze architektury sieci neuronowych zawierają neurony, których wartości wyjściowe zawarte są w pewnym ograniczonym przedziale (na przykład [0; 1] w przypadku logistycznej funkcji aktywacji). Nie stwarza to żadnych problemów ani nie powoduje ograniczeń w przypadku zagadnień klasyfikacyjnych, w których zadana wartość wyjściowa (wskazująca na zaliczenie obiektu do klasy lub na jego brak przynależności do zadanej klasy) może być wygodnie przedstawiona właśnie w takim przedziale. Jednak w przypadku problemów regresyjnych oczekujemy, że sieć poda rozwiązanie w postaci konkretnej wartości, w związku z tym kwestia skalowania tej wartości w taki sposób, by zawsze należała do podanego przedziału musi zostać precyzyjnie rozwiązana. Nie jest to sprawa prosta, zaś pewne jej konsekwencje są bardzo subtelne.

W pierwszej kolejności stosowany jest algorytm skalowania zapewniający, że wartości wyjściowe sieci będą w „rozsądnym" przedziale. Najprostszą taką funkcją skalującą jest funkcja *minimax*: wyznacza ona na podstawie zbioru uczącego wartości minimalną i maksymalną skalowanej zmiennej i przeprowadza przekształcenie liniowe w celu dostosowania wartości tej zmiennej do wymaganego przedziału (zwykle jest to przedział [0,0; +1,0]). Jeśli przekształcenie to zostanie wykonane na ciągłej zmiennej wyjściowej, to z pewnością wszystkie wartości wchodzące w skład zbioru uczącego zostaną przekształcone do zakresu wartości możliwych do zaakceptowania z punktu widzenia wartości dozwolonych w wyjściowych neuronach sieci.

Również od wartości wyjściowych pojawiających się w czasie późniejszego uruchamiania sieci podczas realizacji właściwych obliczeń na nieznanych z góry danych wejściowych wymaga się, aby znajdowały się w tym samym przedziale, dopuszczalnym dla używanego typu neuronów. Konieczność spełnienie tego wymogu może zostać uznana za cechę korzystną, ponieważ umożliwia to wykorzystywanie tych samych formuł skalujących zarówno podczas uczenia, jak i w czasie interpretacji produkowanych przez sieć wyników, ale konieczność używania tych samych formuł skalujących może być także oceniana jako czynnik utrudniający. Jest to związane z zagadnieniem ekstrapolacji neuronowych prognoz poza obszar zdefiniowany dla danych uczących, przy którym obowiązują pewne zasady, które omówimy w skrócie dla różnych rozważanych typów sieci.

- Krzywa wytyczona przez sieć MLP absolutnie **nie może** być w prosty sposób (na zasadzie przedłużenia) ekstrapolowana poza zakres danych uczących. Dotyczy to nawet punktów leżących w pobliżu zbioru uczącego, chociaż przy innych metodach aproksymacji w przypadku bardzo nieznacznego wychylenia się poza zakres wyznaczony przez dane uczące ekstrapolacja miałaby swoje uzasadnienie.
- Musimy się spodziewać, że jeśli w rozważanym przypadku stosowania sieci MLP zmusimy sieć aby podała jakieś oszacowanie nieznanych wartości funkcji poza przedziałem znanych danych uczących, to nie poda ona zalecanego wyżej optymalnego oszacowania poprzez wartość średnią. Wartość, która zostanie wyznaczona w tym przypadku przez sieć, będzie najprawdopodobniej poziomem nasycenia, równym odpowiednio wartości maksymalne lub minimalnej obserwowanej zmiennej wyjściowej – w zależności od tego, czy zbliżając się do rozważanego krańca znanego obszaru krzywa wznosiła się, czy też opadała.

Istnieją różne sposoby postępowania mające na celu korektę tych niedostatków w sieci MLP. Po pierwsze, w miejsce logistycznej funkcji aktywacji w neuronach wyjściowych zastosować można liniową funkcję aktywacji, która po prostu przekazuje dalej niezmieniony poziom pobudzenia neuronu i nie podlega ograniczeniom związanym z pojawianiem się nasycenia. Warto zauważyć, że tylko funkcje aktywacji w warstwie wyjściowej są zmieniane; w warstwach ukrytych w dalszym ciągu korzysta się z logistycznych lub hiperbolicznych funkcji aktywacji. Ponieważ liniowa funkcja aktywacji nie ma poziomu nasycenia, dlatego może ekstrapolować wykryte w danych zależności na większe odległości od zbioru uczącego. Mechanizmu tego nie należy jednak przeceniać, gdyż nieliniowe neurony, mające logistyczne funkcje aktywacji, znajdujące się we wcześniejszych warstwach, nadal spowodują efekt nasycenia w wartościach aproksymowanej funkcji, tylko na pewnym, zdecydowanie wyższym poziomie.

Proponowane rozwiązanie, związane z linearyzacją neuronów, nie jest niestety tak całkiem wolne od wad. Liniowa funkcja aktywacji w sieci MLP może powodować pewne problemy numeryczne podczas realizacji algorytmu wstecznej propagacji błędów. Dlatego w przypadku korzystania z tej metody musi być zastosowany niewielki współczynnik uczenia (mniejszy niż 0,1). Natomiast zaletą opisanego podejścia jest fakt, że gdy jest ono stosowane, może być realizowana ekstrapolacja.

Drugim rozwiązaniem problemów związanych ze stosowaniem sieci MLP do ekstrapolacji wyników uczenia, może być następujący „trick". Otóż można sztucznie zmienić **docelowy** (tzn. występujący po przeskalowaniu) zakres wartości funkcji dla operacji skalowania techniką *minimax*. Na przykład można zażądać, żeby po przeskalowaniu aktualny sygnał wyjściowy (wynikający z danych zawartych w ciągu uczącym) zmieścił się w całości w węższym przedziale, na przykład [0,25; 0,75]. Wszystkie przypadki uczące są wówczas przekształcane do poziomu, który odpowiada tylko środkowej części zakresu wyjściowego neuronów wyjściowych. Jest rzeczą interesującą, że jeśli zakres ten jest mały, i obie wartości ograniczające zbliżają się do 0,5, to eksploatowany zakres charakterystyki neuronu odpowiada samej tylko środkowej części krzywej sigmoidalnej, która jest „prawie liniowa". Łatwo zauważyć, że podejście takie jest wtedy bardzo podobne do omówionego wyżej sposobu polepszenia działania sieci przy użyciu liniowej warstwy wyjściowej. Po zastosowaniu omówionej wyżej sztuczki sieć może realizować (znowu – w ograniczonym zakresie) ekstrapolację zbudowanej funkcji poza zakres danych uczących, ale w końcu i w tym przypadku dojdzie do stanu nasycenia.

Omówione wyżej podejście ma bardzo intuicyjną interpretację. Dzięki zastosowaniu technik „wypychających" efekt nasycenia daleko poza obszar roboczy działania sieci nie dochodzi do deformacji zależności, które sieć ustaliła na podstawie danych uczących. Zależności te – jak można oczekiwać – obowiązują nie tylko w obszarze wyznaczonym przez dane uczące sieć, ale i poza nim. Posługując się tą wyuczoną zależnością można więc próbować „odgadywać" wartości funkcji poza wyuczonym obszarem – zakładając niejawnie, że tam, w nieznanych obszarach obowiązują te same zależności i te same prawa, co w obszarze, który poznaliśmy (z ciągu uczącego). Jednak w miarę oddalania się od znanego obszaru maleje szansa na to, że badana funkcja będzie zachowywała się „grzecznie" i nie zmieni swojego charakteru. Ekstrapolacja jest więc uzasadniona na pewną odległość, poza którą powinna zostać przerwana.

Można dojść do wniosku, że jeśli stosowane jest pierwsze podejście i neurony liniowe są używane w warstwie wyjściowej, to nie ma w zupełności potrzeby użycia algorytmu skalowania, ponieważ neurony mogą osiągnąć dowolny poziom sygnału wyjściowego bez żadnego skalowania. Rzeczywiście, w programie możliwe jest całkowite wyłączenie skalowania z uwagi na zapewnienie wydajności (przy dużej liczbie sygnałów i przy dużej liczbie neuronów w sieci, proces skalowania może zajmować dużo czasu i może wymagać znaczącego zaangażowania komputera). Jednakże w rzeczywistości całkowite pominięcie skalowania jest zwykle niemożliwe, ponieważ powoduje powstanie trudności dla algorytmu uczącego. Brak skalowania sygnałów powoduje, że poszczególne wagi w sieci operują na sygnałach wyrażonych w bardzo zróżnicowanych skalach. W rezultacie zarówno proces inicjalizacji wag jak również (w pewnym stopniu) proces ich zmian w trakcie procesu uczenia jest przy zróżnicowaniu skal bardziej kłopotliwy i bardziej złożony. Z tego powodu wyłączanie skalowania zdecydowanie nie jest polecane, chyba że mamy do czynienia z zadaniem, w którym zakres przewidywanych wartości wyjściowych jest bardzo mały i zbliżony do zera.

Te same argumenty uzasadniają użycie skalowania podczas *preprocessingu* danych wejściowych dla sieci MLP. W zasadzie można uznać, że skalowanie jest tu zbyteczne, ponieważ wagi pierwszej warstwy ukrytej mogłyby w prosty sposób być zmodyfikowane tak, aby same realizowały wymagany sposób skalowania. Jednak z wyżej wymienionych powodów korzystniejsze jest także i w tym przypadku skalowanie sygnałów, niż skalowanie wag.

Powyższa dyskusja skupiona była na zachowaniu się sieci MLP w zagadnieniach regresyjnych, a w szczególności na ich zachowaniu w zastosowaniu do

zadania ekstrapolacji. Sieci korzystające z neuronów radialnych (RBF oraz GRNN) zachowują się zupełnie inaczej i wymagają oddzielnego traktowania.

Nieodłączną cechą sieci radialnych jest ich absolutna niezdolność do ekstrapolacji. Jeżeli przypadek wejściowy odbiega dalej od punktów uczących przechowywanych w neuronach radialnych, to aktywacja neuronów radialnych zmniejsza się bardzo szybko do zera, co prowadzi ostatecznie do zmniejszania się także wartości wyjściowej sieci. W rezultacie niemal każdy zestaw sygnałów wejściowych ulokowanych daleko od centrów radialnych wypracowanych w toku procesu uczenia, generuje wyjście sieci o wartości zerowej, niezależnie od tego jaka jest prawdopodobna wartość ekstrapolowanej funkcji. W rezultacie wyniki ekstrapolacji są złe i nie ma sposobu, by w rozważanej klasie sieci mogły być lepsze.

Brak tendencji sieci RBF do ekstrapolacji może być uznany za cechę pozytywną (sieć nie „udaje", że zna wartość funkcji, skoro jej nie zna), jest to jednak uzależnione od dziedziny zastosowań i punktu widzenia. Generalnie tendencja sieci RBF do szybkiego zmniejszania obliczanych wartości wyjściowych z praktycznym osiąganiem zera tuż poza obszarem umożliwiającym prostą interpolację danych uczących – nie jest przez użytkowników oceniana pozytywnie. Jak wspomniano wyżej, nawet jeśli użytkownik podejmuje decyzję o niestosowaniu ekstrapolacji, to jednak oczekuje, że w przypadku wprowadzenia do sieci całkowicie nowego punktu z obszarów przestrzeni wejściowej odległych od danych uczących, otrzyma na wyjściu jakiś w miarę sensowny sygnał - na przykład zostanie wygenerowana i podana jako odpowiedź wartość średnia ze wszystkich elementów uczących. Tymczasem sieć RBF poza obszarem danych uczących całkowicie odmawia jakiejkolwiek kooperacji, co trudno uznać za jej zaletę.

Aby osiągnąć bardziej wygodne dla użytkownika zachowanie sieci RBF należy przy rozwiązywaniu problemów regresyjnych z wykorzystaniem sieci radialnych przeskalować dane w taki sposób, by średnia wartość wyjściowa była równa 0,0, zaś sposób skalowania pozostałych wartości uzależniony jest od odchylenia standardowego wartości wyjściowych, ustalonego na podstawie przeprowadzonej analizy statystycznej rozkładu ich wartości. Przy takim skalowaniu mamy bardzo dogodną sytuację: jeśli punkty wejściowe, dla których uruchamiana jest sieć podczas testów, położone są na zewnątrz zakresu reprezentowanego w neuronach radialnych, to wartość wyjściowa sieci posiada automatyczną tendencję powrotu do wartości średniej (ponieważ tą wartością średnią po przeskalowaniu jest zero, a sieci RBF mają omówioną wyżej „skłonność" do produkowania zerowego sygnału wyjściowego w obszarze odpowiadającym brakowi danych uczących). Jakość działania sieci regresyjnej może być sprawdzana na kilka sposobów.

Po pierwsze, dla każdego przypadku (lub dla każdego **nowego** przypadku, który został wybrany do testowania) wyznaczyć można wartość wyjściową sieci. Jeśli przypadek ten jest częścią zbioru danych przeznaczonych do testowania (wtedy znana jest poprawna wartość wyjściowa jako tak zwana wartość zadana), to wyświetlana jest różnica pomiędzy wartością obliczoną i zadaną. Analizując te różnice można zorientować się, na ile dokładnie sieć realizuje stawiane jej zadania.

Po drugie, można wyznaczyć statystyki sumaryczne, pokazujące jak dobrze sieć odtwarza zadaną funkcję. Statystyki te zestawiają wartość średnią i odchylenie standardowe dla wartości zawartych w ciągu uczącym jak i dla błędu predykcji. Interpretacja wartości tych statystyk daje bardzo dużo użytecznych informacji, wymaga jednak pewnego zastanowienia, bo uproszczone rozumowanie może w tym przypadku prowadzić do paradoksów.

Rozważmy to bliżej. Powszechnie oczekuje się, że głównym wskaźnikiem jakości działania sieci będzie średnia wartość błędu predykcji. Może się przy tym wydawać, że najlepszą siecią będzie ta, której średni błąd będzie niezwykle bliski wartości zero. Tymczasem tego rodzaju „sukces" w istocie niczego nie dowodzi, ponieważ możliwe jest uzyskanie w prosty sposób zerowego **średniego** błędu predykcji przez sieć szacującą wartość wyjściową stale na tym samym poziomie: średniej wartości dla ciągu uczącego. Taka sieć, nie uwzględniająca wcale wartości zmiennych wejściowych będzie miała **średnio** najmniejszy (zerowy) błąd i może być uznana za lepszą od dowolnej innej sieci neuronowej, realizującej znacznie inteligentniejsze prognozy, ale **średnio** popełniającej większy błąd.

Przytoczone rozumowanie wskazuje na to, że najistotniejszą wartością, pozwalającą ocenić jakość działania sieci jest **odchylenie standardowe dla błędów predykcji**. Jeśli nie jest ono lepsze (czyli mniejsze) niż odchylenie standardowe dla danych uczących, to wówczas działanie sieci nie jest lepsze niż szacowanie dające jako rezultat stale i niezmiennie wartość średnią. Dopiero znacząco mniejsza wartość odchylenia standardowego predykcji upoważnia do twierdzenia, że sieć neuronowa dokonuje sensownego przewidywania.

Dla ułatwienia opisanego wyżej wnioskowania często prezentuje się – obok „surowych" wartości średnich i odchyleń standardowych – również **stosunek** odchylenia standardowego dla błędów predykcji do odchylenia standardowego danych wchodzących w skład ciągu uczącego. Wartość tego wskaźnika w znaczący sposób mniejsza od 1,0 wskazuje na **dobrą** re-

alizację regresji, gdyż rozrzut predykcji jest mniejszy, niż rozrzut „surowych" danych. Dla konkretyzacji ocen przyjmuje się, że naprawdę dobrej regresji odpowiada wartość omawianego wskaźnika mniejsza od 0,1. Ten wskaźnik regresji (a raczej, mówiąc precyzyjnie, wartość obliczana jako jeden minus ten wskaźnik) jest czasami interpretowany jako stopień zdeterminowania badanego zjawiska. Potocznie mówi się, że wartość wyliczona w wyżej opisany sposób pokazuje wyjaśnioną przez model część wariancji danych. Im ta część jest większa, tym lepszy (oczywiście) jest model.

## 6. PODSUMOWANIE

W pracy dokonano przeglądu zagadnień związanych ze stosowaniem sieci neuronowych w zadaniach związanych z szeroko rozumianą metalurgią, wskazując na uwarunkowania, jakie wskazane zadania narzucają na wybór struktury używanej sieci. W powiązaniu z wcześniejszymi pracami Autora, dyskutującymi zagadnienia metod uczenia sieci oraz zasady wyboru danych tworzących zbiór uczący – tworzy to zasób wiedzy wystarczający do tego, by podjąć samodzielne eksperymenty z użyciem sieci neuronowych jako narzędzi w dalszych zagadnieniach metalurgicznych. Właśnie takie jest przeznaczenie tej pracy – tworzącej naukowe przesłanki dla dalszych praktycznych zastosowań neurokomputingu w metalurgii.

## LITERATURA


Aistleitner K., Mattersdorfer L.G., Haas W., Kugi A., 1996, Neural network for identification of roll eccentricity in rolling mills, *J. Mat. Proc. Techn.*, 60, 387-392.

Cerqueira J.J.F., Palhares A.G.B., Madrid M.K., 2000, A complement to the back-propagation algorithm: an upper bound for the learning rate, *Proc. IEEE-INNS-ENNS Int. Joint Conf. on Neural Networks*. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium. *IEEE Comput. Soc.*, 4, 517-522

Chidambaram P.R., Jones J., Cao X.Y., Rohdes, V., 1995, Modelling of weld metal properties as a function of weld metal composition. Welding and Weld Automation in Shipbuilding, Cleveland, 123-133.

Cios K.J., Baaklini G.Y., Vary A., Tjia R.E., 1994, Radial basis function network learns ceramic processing and predicts related strength and density, *ASTM J. Testing and Evaluation*, 22, 343-350.

Cios K.J., Berke L., Vary A., Sharma S., 1996, Soft Computing Methods in Design of Superalloys, *Report NASA Nr. TM-106888.*

Colla V., Reyneri L.M., Sgarbi M., 2000, Neuro-wavelet parametric modeling. Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000, Neural Computing: New Challenges and Perspectives for the New Millennium, *IEEE Comput. Soc.*, 4, 499-504.

Cooper D.J., Woll S.L.B., Souder B.V., 1996, Online pattern-based part quality monitoring of the injection molding process, *Polymer Eng Sci.*, 36, 1477-1488.

Cuo Z. i in., 1995, Neural network approach in research of metal corrosion in soil. *Corrosion Science and Protection Technique*, 7, 258-262.

Demirci H.H., Coulter J.P., Guceri S.I., 1996, A numerical and experimental investigation of neural network-based intelligent control of molding processes, *J. Manuf. Sci. Eng.* 119, 88-94.

Di S., Thomson P.F., 1997, Neural network approach for prediction of wrinkling limit in square metal sheet under diagonal tension. *ASTM, J. Testing and Evaluation*, 25, 74-81.

Draeger A., Gesthuisen R., 1996, Neuronal network for estimation of the degree of polymerization in the polycondensation of polyethyleneterephthalate, *Chemie Ingenieur Technik*, 68, 1320-1323.

Fujii H., Mackay D.J.C., Bhadeshia H.K.D.H., 1996, Bayesian neural network analysis of fatigue crack growth rate in nickel base superalloys, *ISIJ Int.*, 36, 1373-1382.

Gavard L., Bhadeshia H.K.D.H., MacKay D.J.C., Suzuki S., 1996, Bayesian neural network model for austenite formation in steels, *Mat. Sci. Techn.*, 12, 453-463.

Giles A.E., Aldrich C., van Deventer J.S.J., 1996, Modelling of rare earth solvent extraction with artificial neural nets. *Hydrometallurgy*, 43, 241-255.

Gradisek J., Govekar E., Grabec I., 1996, A chaotic cutting process and determining optimal cutting parameter values using neural networks, *Int. J. Mach. Tools & Manufact.*, 36, 1161-1172.

Hattori M. i in, 1996, Estimation of thermal-deformation in machine tools using neural network technique, *J. Mat. Proc. Techn.*, 56, 765-772.

Hill E.v.K, Walker J.L., Rowell G.H., 1994, Neutral network burst pressure prediction in graphite/epoxy pressure vessels from acoustic emission amplitude data, *NDE, An Answer for an Aging World*, New Orleans, 161-163.

Hill E.v.K, Walker J.L., Rowell G.H., 1996, Burst pressure prediction in graphite/epoxy pressure vessels using neural networks and acoustic emission amplitude data, *Materials Evaluation*, 54, 744-748, 754.

Isalgue A. i in., 1994, Meso-scale model of a Cu-Zn-Al single crystal SMA, *Mechanics of Phase Transformations and Shape Memory Alloys*, Chicago, 71-84.

Izworski A., Tadeusiewicz R., 2000, Classification of auditory brainstem response signals by application of neural networks, in: Halici U., Leblebicioglu K., Atalay V., Nalcaci E. (eds.), *Brain Machine, METU*, Ankara, 146-156.

Jin B., Soeda M., Oshima K., 1996, Control of weldpool width and cooling time in TIG welding using a neural network model, *Welding Int.*, 10, 614-621.

Johansen B.S., Lilholt H., Liisberg C., 1995, Design for durability of polymeric composites, *Tenth Int. Conf. on Composite Materials*, I. Fatigue and Fracture, Whistler, 683-690.

Jung J -Y, Im Y-T., Lee-Kwang H., 1996, Fuzzy-control simulation of cross-sectional shape in six-high cold-rolling mills, J. Mat. Proc. Techn. 62, 61-69.

Keshavaraj R., Tock R.W., Narayan R.S., Bartsch R.A., 1995, Neural-network-based model approach for density of high-molecular-weight esters used as plasticizers. *Adv. Polym. Technol.*, 14, 215-225.

Kim B.H., Hwang T.W., Farooqi S., Witkus, S., 1995a, Prediction of injection molded plaque properties using artificial neural networks and flow simulation, *ANTEC '95. Vol. I-Processing*, Boston, 714-721.

Kim B.H., Hwang T.W., Park, H J, 1995b, Predicting mechanical properties of acrylonitrile-butadiene-styrene terpolymer in injection molded plaque and box, *Polym. Eng. Sci.*, 35, 1252-1259.

Konishi M., 1993, Development of intelligent processes in iron and steel industry, *Met. Techn.* (Japan), 63, 11-15.

Larkiola J., Myllykoski P., Nylander J., Korhonen A.S., 1996, Prediction of rolling force in cold rolling by using physical models and neural computing, *J. Mat. Proc. Techn.*, 60, 381-386.

Lee B.H., Kim B.H., 1995, Automated selection of gate location based on desired quality of injection molded part, *ANTEC '95. Vol. I-Processing*, Boston, 554-560.

Lee, L.J., Twu J.T., 1995, Application of artificial neural networks for the optimal design of sheet molding compound (SMC) compression molding, *Polym. Compos.*, 16, 400-408.

Liao T.W., Chen L.J., 1994, A neural network approach for grinding processes: modelling and optimization. *Int. J. Mach. Tools Manuf.*, 34, 919-937.

Matsuda K. i in., 1994, Sintering process control using fuzzy inference and neural network. *6th Int. Symp. on Agglomeration*, Nagoya, 421-426.

Middle J.E., Khalaf, G.H., 1995, Neural network modeling of temperature distribution for control of gas metal arc welding. Modeling of Casting, *Welding and Advanced Solidification Processes VII*, London, 397-405.

Myllykoski P., Larkiola J., Nylander J., 1996, Development of prediction model for mechanical properties of batch annealed thin steel strip by using artificial neural network modelling, *J. Mat. Proc. Techn.*, 60, 399-404.

Payne R.D., Rebis R.E., Moran A.L., 1993, Spray forming quality predictions via neural networks, *J. Mat. Eng. Perform.*, 2, 693-701.

Rademan J.A.M. i in., 1996, Neural net based knowledge extraction from the historical data of an industrial leaching process, *Hydrometallurgy*, 43, 95-116.

Rao H.S., Mukherjee A., 1996, Artificial neural networks for predicting the macromechanical behaviour of ceramic-matrix composites, *Comp. Mat. Sci.*, 5, 307-322.

Rojas I., Pomares H., Gonzalez J., Ros E., Salmeron M., Ortega J., Prieto A., 2000, A new radial basis function networks structure: application to time series prediction, *Proc. IEEE-INNS-ENNS Int. Joint Conf. on Neural Networks, IJCNN 2000*. Neural Computing: New Challenges and Perspectives for the New Millennium, *IEEE Comput. Soc.*, 4, 449-454.

Samways N.L., 1996, In-line gage enhances galvannealing, *Iron Steel Eng.*, 73, 169-170.

Sayeh M.R., Viswanathan R., Dhali S.K., 1990, Neural networks for the assessment of impact on composite materials, *Composite-Technology, 6th Annual Conf.*, Carbondale, 77-86.

Sergi V., 1996, A neural network for optimization of laser welding of sheet steel, *LAMIERA*, 33, 72-77.

Shen Y., Moon S., 1996, Error compensation of coordinate measurements in computer-integrated manufacturing using neural networks, *J. Mat. Proc. Techn.*, 61, 12-17.

Slattery K.T., 1995, Randon-damage finite element for studying micromechanical failure mechanisms in advanced composites, *J. Thermoplast. Compos. Mater.*, 8, 260-271.

Stone R., Krishnamurthy K., 1996, A neural network thrust force controller to minimize delamination during drilling of graphite-epoxy laminates, *Int. J. Mach. Tools Manuf.*, 36, 985-1003.

Tadeusiewicz R., 1993, *Sieci Neuronowe*, Akademicka Oficyna Wydawnicza, Warszawa.

Tadeusiewicz R., 1997, Możliwości i problemy wykorzystania sieci neuronowych w prognozowaniu procesów gospodarczych, *Zeszyty Naukowe Akademii Ekonomicznej w Krakowie*, nr 493, 5-18.

Tadeusiewicz R., 1998, Badanie funkcjonowania mózgu za pomocą sieci neuronowych, w pracy zbiorowej: *Kognitywistyka i Media w Edukacji*, 1, 169 – 203.

Tadeusiewicz R., Izworski A., Wszołek W., Wszołek T., 1999, Processing and Classification of Deformed Speech Using Neural Networks, w: Fouke J.M., Nerem R.M., Blanchard S.M., Yoganathan A.P., (eds.): Serving Humanity, Advancing Technology, *Proc. 1st Joint BMES/EMBS Conf.* (IEEE Catalog Number 99CH37015C), Atlanta, 927-928.

Tadeusiewicz R., Wszołek T., Izworski A., Wszołek W., 2000, Recognition of defects in High Voltage Transmission Lines Using the Acoustic Signal of Corona Effect, in: Widrow B. (i in.. eds.): *Neural Networks for Signal Processing X, IEEE*, New York, 869-875.

Teshima T. i in., 1993, Estimation of cutting tool life by processing tool image data with neural network, *Ann. CIRP, Manuf. Techn.*, 42, 59-62.

Thavasimuthu M., Rajagopalan C., Kalyanasundaram P., Raj B., 1996, Improving the evaluation sensitivity of an ultrasonic pulse echo technique using a neural network classifier, *NDT & E Int.*, 29, 175-179.

Torres V.M. Chaves A.P. Meech J.A., 2000, Intelligold-an expert system for gold plant process design, *Cybernetics & Systems*, 31, 591-610.

Walker J.L., Hill E.v.K., 1994, Back propagation neural networks for predicting ultimate strengths of unidirectional graphite/epoxy tensile specimens, *NDE: An Answer for an Aging World*, New Orleans, 158-160.

Workman G.L., Walker J.L., 1996, Materials characterization of powder metallurgy products using acousto-ultrasonics [and neural nets]. *Mat. Sci. Forum*, 210-213, 679-86.

Xia Z.N., Lai S.G., Sun Y.Z., Lu Y.W., 1996, Study on property prediction for sealing alloys. *Acta Metall. Sinica* (English Letters), 9, 307-309.

Yo K., Okigata K., 1993, Application of expert systems in metal working processes. Met. Techn. (Japan), 63, 38-43.

Zgonc K., Achenbach J.D., 1996, A neural network for crack sizing trained by finite element calculations, *NDT & E Int.*, 29, 147-155.