AKADEMIA GÓRNICZO – HUTNICZA
IM. STANISŁAWA STASZICA

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I ELEKTRONIKI

Adrian Horzyk

Nowe metody uczenia sieci neuronowych bez sprzężeń zwrotnych

Praca doktorska
napisana pod kierunkiem
Prof. zw. dr hab. inż. Ryszarda Tadeusiewicza

Kraków 2001

SPIS TREŚCI

1. WSTĘP

1.1. Tworzenie nowych typów sieci neuronowych jako wyjście naprzeciw niektórym potrzebom dzisiejszej informatyki

W ostatnich latach mamy do czynienia z prawdziwym zalewem informacji pochodzących z różnych źródeł, a także ze związanym z tym problemem ich szybkiego i efektywnego przetwarzania oraz selekcji. Ogromne bazy danych, stające się coraz częściej niezbędnym elementem procesu sterowania lub zarządzania, wielkie zasoby wiedzy korporacyjnej, będące fundamentem funkcjonowania nowego typu przedsiębiorstw zorientowanych na telepracę, różne formy sieciowej działalności gospodarczej (tak zwany *e-business*), Internet jako źródło wiedzy używanej przez miliony ludzi – to tylko przypadkowo wybrane elementy, wskazujące na rozmiar i znaczenie tego zjawiska. Tradycyjne techniki przetwarzania informacji, oparte na dokładnym przetwarzaniu „wszystkiego", nie zawsze dają oczekiwany efekt w rozsądnym czasie, nie mówiąc już o problemach takiego pełnego przetwarzania informacji. Wskazane problemy nasilają się zdecydowanie w systemach, w których wymagana jest praca w czasie rzeczywistym (*real-time processing*), a więc zwłaszcza w systemach automatyki. Na domiar wszystkiego w szybko zmieniających się warunkach działania wielu przedsiębiorstw i związanych z nimi systemów informatycznych pojawia się konieczność szybkiego dostosowywania istniejących algorytmów do nowych warunków, a to nie zawsze okazuje się rzeczą łatwą. Często brak teoretycznych i praktycznych rozwiązań dla zarysowanych tu problemów powoduje stagnację rozwoju w danej dziedzinie przetwarzania informacji – ze szkodą dla techniki i gospodarki. Tymczasem rozwiązanie przynajmniej niektórych spośród naszkicowanych wyżej problemów może być znalezione, pod warunkiem, że w obszar rozwiązań dopuszczalnych włączone zostaną metody niestandardowe, takie właśnie, jak rozważane w tej pracy. Odwołując się do sieci neuronowych jako do narzędzia rozwiązywania wybranych problemów informatycznych musimy oczywiście mieć świadomość ich ograniczeń, na przykład tego, że wyniki obliczeń neuronowych są zwykle raczej jakościowe, niż ilościowe, stąd ich dokładność jest ograniczona.

Nie zawsze jednak w zastosowaniach informatyki potrzebne są dokładne, precyzyjne rozwiązania, a nawet można powiedzieć, że w większości przypadków wystarcza, żeby dane rozwiązanie charakteryzowało się jakąś z góry określoną dokładnością (*accuracy*), tzn. nie było obarczone większym błędem niż zakłada pewien dopuszczalny próg. Ważniejszym od dokładności atutem obliczeń neuronowych jest bowiem elastyczność tych rozwiązań,

wynikająca z możliwości, jakie niesie proces uczenia. Często zdarza się sytuacja, że mamy pewien zbiór danych, zbiór doświadczeń i empirycznie obserwowanych ich wzajemnych powiązań (korelacji), natomiast brak nam odpowiedniego algorytmu przetwarzania lub jawnie zdefiniowanej reguły wnioskowania na ich podstawie. Zazwyczaj istnieje pewien zbiór wiedzy, tj. dane, ich korelacje oraz znana jest dokładność, z jaką przetwarzanie tych danych ma zachodzić. Najbardziej odpowiednie byłoby, gdyby dla dowolnych danych i dla zadanej dokładności można było przy pomocy jakiejś czarnej skrzynki (*black box*) po rozsądnym czasie otrzymać pożądane wyniki w postaci stwierdzenia, co właściwie z tych danych wynika. W takich przypadkach chętnie sięgamy do metod heurystycznych, opartych na procesach uczenia maszynowego, w tym ostatnio coraz chętniej właśnie do sieci neuronowych.

Praktyka informatyki (zwłaszcza w obszarze zastosowań komputerów do sterowania i zarządzania) wymusza obecnie coraz szybsze przetwarzanie coraz większej ilości informacji o coraz bardziej zawiłych powiązaniach. W takich przypadkach nawet wtedy, gdy znamy modele przyczynowo-skutkowe, celowe może być użycie sieci neuronowych, gdyż prowadzą one do rozwiązań szybszych – chociaż często tylko suboptymalnych.

Sieci neuronowe nie bazują na z góry opracowanych algorytmach, lecz na procesie uczenia, który powinien umożliwiać im jak najlepsze dostosowanie się do zadanego problemu na podstawie pewnej wiedzy dostarczonej w formie przykładów poprawnych rozwiązań, tzw. wzorców uczących (*learning patterns*). Wzorce te tworzą pewien zbiór zwany ciągiem uczącym (*LS – learning sequence*, *DS – data set*) – stanowiący główne źródło wiedzy sieci o zadanym problemie. W zależności od specyfiki problemu i od zastosowanej metody uczenia ciąg uczący składa się:

- z wzorców wejściowych dla metod uczenia bez nauczyciela (*unsupervised learning*),
- z wzorców wejściowych i z prawidłowych odpowiedzi sieci neuronowej, zwanych też sygnałami uczącymi dla danych wzorców, co znajduje zastosowanie w przypadku metod uczenia z nauczycielem (*supervised learning*),
- z wzorców wejściowych i odpowiedzi w postaci nagrody lub kary dla metod uczenia z krytykiem (*reinforcement learning*).

W przypadku metod uczenia z nauczycielem, rozważanych w tej pracy, ciąg uczący jest zazwyczaj definiowany jako zbiór par $(x_i, y_i)_{i=1}^{N}$, gdzie N oznacza liczność tego zbioru, a pary (x_i, y_i) są przykładami poprawnego odwzorowania x_i w y_i. Na podstawie zbioru uczącego

przeprowadzany jest proces uczenia (*learning*) sieci neuronowej, który ma na celu wydobycie korelacji pomiędzy wzorcami (często ukrytych w wejściowych danych), a także odpowiedniej zależności pomiędzy sygnałami wejściowymi i wyjściowymi w sieci.

Celem końcowym uczenia sieci jest znalezienie optymalnego ustawienia parametrów wolnych sieci (głównie wag) tak, by sieć neuronowa w miarę możliwości spełniała postulaty zdefiniowane ciągiem uczącym. Gdyby jednak sieć neuronowa miała służyć tylko do zapamiętywania wzorców ciągu uczącego (i to jeszcze z ograniczoną dokładnością), stosowanie tych sieci nie miałoby głębszego sensu, albowiem istnieją bardziej efektywne i szybsze metody (wykorzystywane w bazach danych) służące do dokładnego zapamiętywania, przetwarzania, i przeszukiwania informacji na podstawie relacji pomiędzy nimi zawartych. Prawdziwa siła sieci neuronowych ukryta jest w ich zdolności do uogólniania (*generalization*) wiedzy zdobytej w procesie uczenia. Uogólnianie pozwala – po nauczeniu sieci – na rozwiązywanie zadań podobnych do tych, które były prezentowane wcześniej w ciągu uczącym. Umożliwia to w efekcie zadawanie sieci neuronowej pytań spoza zbioru uczącego i oczekiwanie, że sieć neuronowa da na nie sensowną odpowiedź.

Pomysł z sieciami neuronowymi i ich uczeniem ma wiele zalet, jednak w związku z jego wdrożeniem do konkretnego praktycznego zadania związanych jest kilka zasadniczych problemów:

- Jak ma wyglądać optymalna sieć neuronowa dla danego ciągu uczącego?
- Jak obrać jej stan początkowy, tj. jak ją zainicjować?
- W jaki sposób i jak długo ją uczyć?
- Jakich wyników możemy oczekiwać?
- Jak ocenić jakość uzyskanej zdolności do uogólniania?
- Czy możemy nauczonej sieci neuronowej zaufać, jeśli chodzi o jej odpowiedzi na zadawane jej pytania?

Aby sieci neuronowe mogły stać się użyteczne, godne zaufania i powszechnie stosowane, konieczne jest odpowiedzenie na te zasadnicze pytania i znalezienie efektywnych metod realizacji sieci zdolnych do pokonania wyżej wymienionych problemów.

Okazuje się, że współczesna wiedza na temat metod i technik obliczeniowych nie gwarantuje możliwości zbudowania systemu, który w oczywisty i naturalny sposób mógłby sprostać tak sformułowanemu zadaniu. Niewątpliwie biologiczny odpowiednik sieci

neuronowych – mózg człowieka (a nawet znacznie mniej skomplikowany mózg zwierzęcia, na przykład psa) – radzi sobie ze wszystkimi tymi problemami. Jednak jak na razie barierę postępu stanowi niepełna wiedza związana z jego funkcjonowaniem. Mimo wielu wspaniałych odkryć ostatniego stulecia, związanych z badaniem układów nerwowych istot żywych, wciąż brak dostatecznie ogólnej wiedzy, umożliwiającej sztuczne modelowanie takich struktur z dużą dokładnością. Wysiłki wielu znakomitych badaczy z różnych dziedzin nauki zmierzają do odkrycia tajemnic mózgu i do ich wykorzystania.

Warto zauważyć, że w przeszłości wiele tych osiągnięć w obszarze neurobiologii, obejmowanym przez szeroko rozumianą biocybernetykę, które wiązały się z największym rozwojem podstaw sztucznej inteligencji, uhonorowano najwyższą możliwą godnością naukową - przyznawaną przez Szwedzką Akademię Nagrodą Nobla. Przypomnijmy niektóre z nich, gdyż warto może uświadomić sobie, jak solidne podstawy biocybernetyczne ma prezentowana w tej pracy dziedzina informatyki. I tak, chronologicznie wyróżnić można następujące osiągnięcia ukoronowane nagrodą Nobla, będące z całą pewnością wynikami badań biocybernetycznych w swojej istocie (chociaż nie zawsze tak nazywanych). Podano datę nadania nagrody, nazwisko laureata i skrótową informację o istocie nagrodzonego osiągnięcia.

- 1904 - *Pavlov I.P.* - teoria odruchów warunkowych
- 1906 - *Golgi C.,* - badanie struktury układu nerwowego
- 1906 - *Ramón Y Cajal S.* - odkrycie, że mózg składa się z sieci oddzielnych neuronów
- 1920 - *Krogh S.A.* - opisanie funkcji regulacyjnych w organizmie
- 1932 - *Sherrington Ch. S.* - badania sterowania nerwowego pracy mięśni
- 1936 - *Dale H., Hallett L.O.* - odkrycie chemicznej transmisji impulsów nerwowych
- 1944 - *Erlanger J., Gasser H. S.* - procesy w pojedynczym włóknie nerwowym
- 1949 - *Hess W.R.* - odkrycie funkcji śródmózgowia
- 1963 - *Eccles J.C., Hodgkin A.L., Huxley A.F.* - mechanizm elektrycznej aktywności neuronu
- 1969 - Granit R., Hartline H.K., Wald G. - fizjologia widzenia
- 1970 - *Katz B., Von Euler U., Axelrod J.* - transmisja humoralnej informacji nerwowej w zakończeniach nerwowych
- 1974 - *Claude A., De Duve Ch., Palade G.* - badania strukturalnej i funkcjonalnej organizacji komórki.
- 1977 - Guillemin R., Schally A., Yalow R. - badania hormonów mózgu

- 1981 - *Sperry R.* - odkrycia dotyczące funkcjonalnej specjalizacji półkul móżdżku
- 1981 - *Hubel D.H., Wiesel T.* - odkrycie zasad przetwarzania informacji w systemie wzrokowym
- 1991 - *Neher E., Sakmann B.* - funkcje kanałów jonowych w komórkach nerwowych

Jak wynika z podanego wyżej, bezspornie dosyć subiektywnie i pobieżnie dokonanego zestawienia, w ciągu ostatniego stulecia dokonał się ogromny postęp w badaniach mózgu, mający swoje bezpośrednie przełożenie na osiągnięcia i sukcesy sztucznej inteligencji. Szacuje się, że w szczególnie owocnych latach drugiej połowy XX wieku uzyskiwano w ciągu jednego miesiąca więcej nowych informacji o budowie i działaniu mózgu - niż ich zgromadzono od czasów starożytności do 1900 roku. Nic więc dziwnego, że na tak bogatej „glebie" biologicznych odkryć i biologicznych faktów pojawiały się próby przeniesienia coraz lepiej rozumianych zasad działania mózgu do wnętrza maszyn matematycznych, które równolegle i równocześnie zyskiwały dojrzałość i stawały się coraz doskonalszymi imitatorami naturalnego ludzkiego intelektu.

Potrzeby techniki wyprzedziły jednak ten zasób informacji o mózgu, jaki zgromadzono w trakcie badań neurocybernetycznych i który realnie można było zastosować jako bezpośrednie źródło biologicznych inspiracji dla konkretnych rozwiązań technicznych – i wtedy pojawiło się miejsce na tworzenie struktur sieci neuronowych częściowo tylko wzorowanych na biologicznych pierwowzorach, mających jednak tę zaletę, że są one bardzo sprawne w rozwiązywaniu konkretnych problemów informatycznych. Takie właśnie sieci zaproponowano w tej pracy jako swoiste „wyjście naprzeciw" pewnym (wybranym) potrzebom współczesnej informatyki.

1.2. Koncepcje badawcze i teza pracy

Realizując zapowiedziany w końcu poprzedniego rozdziału ambitny program badawczy w tej pracy zaprezentowano dwie oryginalne metody pozwalające tworzyć, formować oraz uczyć sieci neuronowe, próbujące zmierzyć się z wybranymi problemami wymienionymi w poprzednim rozdziale. Najpierw przedstawiono metodę automatycznej konfiguracji sieci neuronowej dla problemów sprowadzających się do rozpoznawania wzorców binarnych. Następnie opisano znacznie bogatszą metodę, nazwaną metodą sterowanych kompromisów, która działa na danych uczących (i testowych) o wartościach rzeczywistych. Obydwie opracowane przez autora metody próbują wyeliminować niektóre mankamenty, występujące podczas używania obecnie stosowanych metod uczenia sieci neuronowych. Rozważania ograniczono wyłącznie do sieci o topologiach nie zawierających sprzężeń zwrotnych, ponieważ takie właśnie sieci są aktualnie najczęściej i najchętniej wykorzystywane.

W pierwszej opisanej w pracy metodzie skoncentrowano się na maksymalnym przyspieszeniu procesu kreowania architektury i parametrów sieci neuronowej dla pewnej wąskiej grupy problemów, w których dane wejściowe i wyjściowe sieci mają charakter wektorów binarnych. W drugiej, bardziej ogólnej metodzie wyeksponowano próbę poprawienia właściwości ekstrapolacyjnych nauczonej sieci poprzez zastosowanie nietypowych funkcji aktywacji neuronów, a także podjęto próbę przyspieszenia procesu uczenia poprzez wyeliminowanie współczynnika uczenia η. Najważniejszą innowacją wnoszoną przez opisaną w pracy metodę sterowanych kompromisów jest to, że stanowi ona próbę określenia zupełnie nowego sposobu uczenia sieci, dążącego do bardziej efektywnego zmierzania parametrów i struktury sieci do optimum postawionego ciągiem uczącym, powiązanego z tzw. minimum globalnym funkcji błędu.

Prezentacja oryginalnych koncepcji nowych struktur i nowych metod działania sieci połączona została w tej pracy z próbą oszacowania jakości uzyskanego uogólnienia przez skonfigurowaną sieć neuronową. Żadna z zaprezentowanych metod nie gwarantuje, co prawda, zupełnego rozwiązania wszystkich wyżej wymienionych problemów związanych ze stosowaniem sieci neuronowych, jednakże obydwie zaproponowane metody stanowią pewną próbę rozwiązania przynajmniej części z nich, więc niezależnie od tego, że wzbogacają naukową wiedzę o sieciach neuronowych i metodach ich uczenia, to dodatkowo dostarczają wyników praktycznych (których wyrazem są załączone do pracy programy komputerowe), które mogą w efekcie służyć w różnych zastosowaniach praktycznych do w miarę efektywnego rozwiązywania dobrze zdefiniowanych grup problemów.

W świetle opisanych założeń zdefiniowano następujące tezy pracy:

Możliwe jest połączenie procesu treningu i automatycznego formowania struktury prostej sieci neuronowej za pomocą proponowanej w pracy metody uczenia jednowarstwowych sieci liniowych, co prowadzi do szybkiego i efektywnego znajdowania optymalnych rozwiązań w rozważanej klasie zadań sprowadzających się do rozpoznawania prostych wektorów binarnych.

Uogólnienie metody uczenia wzmiankowanej w pierwszej tezie pozwala na całkowite oderwanie procesu uczenia jednokierunkowych sieci neuronowych od powszechnie stosowanych metod opartych na minimalizacji funkcji błędu. Sięgnięcie do proponowanej w pracy metody uczenia za pomocą sterowanych kompromisów pozwala

w wielu zadaniach znacząco przyspieszyć i uprościć proces rozwiązywanie zadania w stosunku do rutynowo stosowanych technik uczenia sieci.

Realizując tematykę badawczą sygnalizowaną przez sformułowane wyżej tezy w pracy przedstawiono dwie oryginalne metody uczenia, związane wprawdzie z tradycyjną tematyką sieci neuronowych jednokierunkowych (pozbawionych sprzężeń zwrotnych), ale rozwiązujące w zupełnie nowy sposób zagadnienia ich konfiguracji i uczenia. Zgodnie z kolejnością zasygnalizowanych tez najpierw przedstawiona jest metoda automatycznej konfiguracji sieci neuronowych dla problemów rozpoznawania wzorców binarnych a następnie zaproponowano bardziej ogólną metodę, nazwaną metodą sterowanych kompromisów. W związku z wprowadzeniem każdej metody podano wstępne założenia i przesłanki, jakie przyświecały ich skonstruowaniu, następnie opisano sposób ich działania i wyszczególniono ich zalety oraz wady w takim zakresie, w jakim zdołano to ustalić w trakcie prowadzonych badań. Opis dopełniono krótką charakterystyką aplikacji badawczych, służących do przetestowania słuszności założeń obydwu metod, jak i do prowadzenia badań porównawczych mających na celu poprawienie efektywności zaproponowanych metod.

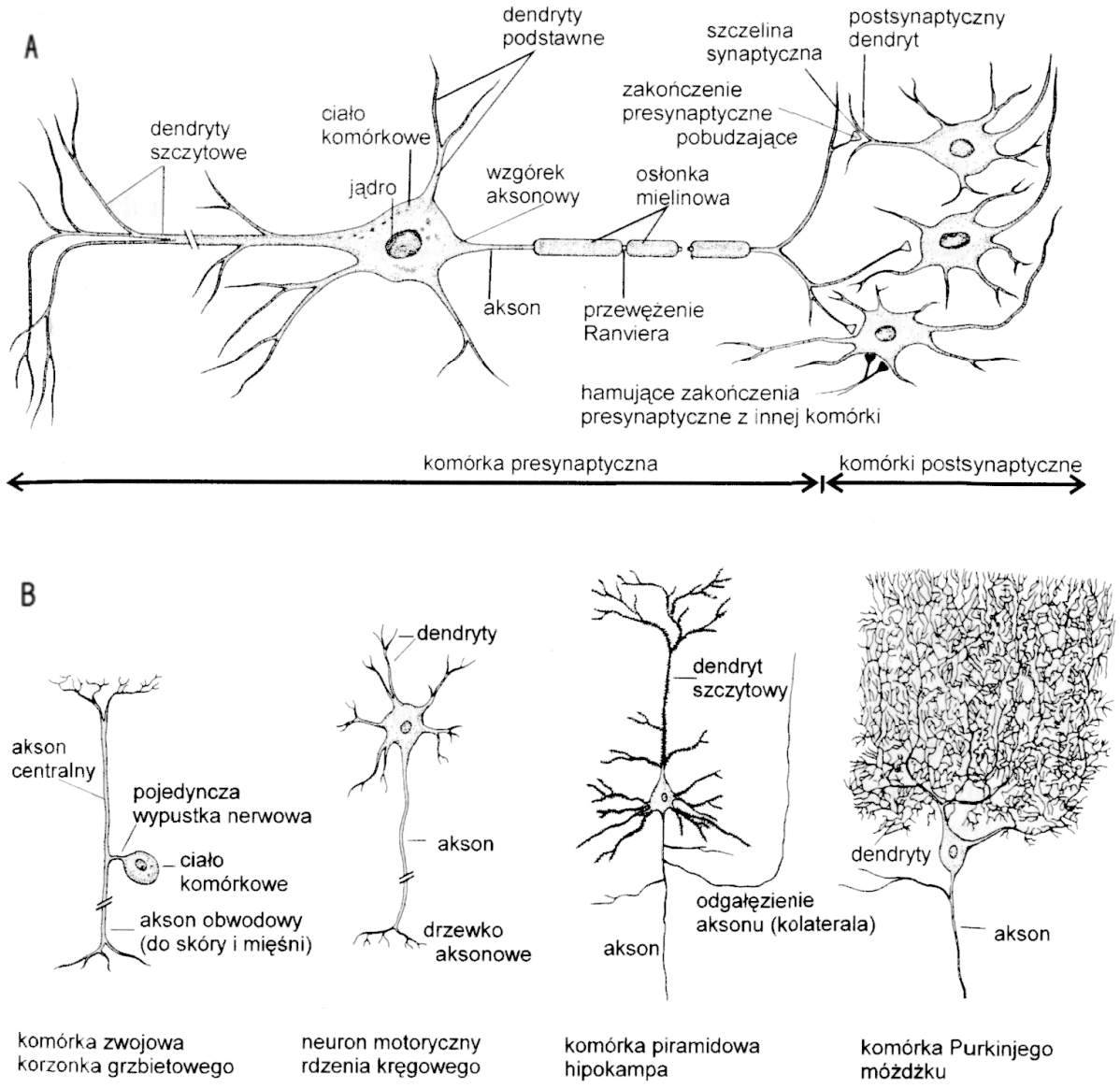
W końcowej części pracy przytoczone są przykłady zastosowania opracowanych metod do wybranych zadań pokazujących poglądowo ich zalety i wady, a także podane są wyniki ich działania, mogące służyć do porównań z osiągami uzyskanymi przez inne reguły uczące. Pracę zakończono podsumowaniem uzyskanych wyników i wyznaczeniem kierunków dalszych poszukiwań i badań zmierzających do udoskonalenia zaproponowanych metod.

2. WPROWADZENIE

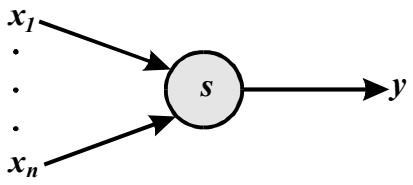
2.1. Mózg a sieci neuronowe

Mózg od dawien dawna fascynował wielu badaczy i był natchnieniem dla wielu filozofów, jednak możliwości poznawania tej galaretowatej struktury ograniczały się w minionych stuleciach zazwyczaj tylko do spostrzeżeń, jakie badacz mógł zgromadzić na podstawie obserwacji różnych zachowań behawioralnych lub w oparciu o analizę toku myślenia i rozumowania. Dopiero stosunkowo niedawno odkrycia naukowe z dziedzin fizyki, chemii i biologii dostarczyły pewnej aparatury służącej do badania tej struktury. W ostatnich latach możemy odnotować mnożenie się różnych technik nieinwazyjnego badania procesów zachodzących w żywym mózgu (takich jak PET tomografia pozytronowa, CT – tomografia komputerowa, SPECT – tomografia fotonowa, EEG – elektroencefalografia). Dzięki tym technikom możliwe stały się obserwacje żywego mózgu i procesów w nim zachodzących. Wynikiem różnorodnych badań są między innymi pewne modele matematyczne, które w sposób ilościowy próbują wyrazić naturę zjawiska zachodzącego w mózgu lub jego częściach. Ze względu na dużą złożoność mózgu i ogromną ilość neuronów (liczbę elementów składowych mózgu szacuje się na około 10^{10}), modelowaniu podlegają zazwyczaj tylko pewne jego części, które ze względu na strukturę jaką tworzą nazwano „**sieciami neuronowymi**”. Przesyłanie i przetwarzanie informacji pomiędzy biologicznymi neuronami odbywa się na bazie skomplikowanych procesów fizyko-chemicznych za pośrednictwem tzw. neurotransmiterów.

Niestety wiedza o żywym mózgu i jego poszczególnych podzespołach jest na razie zbyt skromna, żeby można było skonstruować jego funkcjonujący model matematyczny. Działanie mózgu związane jest nie tylko z procesami myślowymi, lecz również z wieloma innymi aspektami, które wywierają na niego duży wpływ. Potężna moc obliczeniowa mózgu bierze się z jego zdolności do równoległego wykonywania wielu zdań. Wprawdzie szybkość działania jego poszczególnych składników (neuronów) jest niewielka, jednak razem tworzą potężną maszynę obliczeniową.



Rysunek 2.1. Biologiczne neurony. **A** – podstawowe elementy komórki nerwowej kręgowca. Z ciała komórki wychodzą dwa rodzaje wypustek: dendryty i akson. Aksony mogą mieć różną długość, sięgającą do 1m i są na ogół bardzo cienkie (0,2-20 μm). Potencjały czynnościowe powstają na wzgórku aksonowym i odnawiają się w kolejnych przewężeniach Ranviera (w aksonach z osłonką mielinową). Końcowe rozgałęzienia aksonów (tzw. drzewka aksonowe) zakończone są kolbkami synaptycznymi (na rysunku białe trójkąty symbolizują kolbki synaptyczne komórki pobudzającej, czarne trójkąty – kolbki synaptyczne komórki hamującej) na wielu (nawet do 1000) komórkach postsynaptycznych. **B** – komórki z rozmaitymi morfologicznymi typami wypustek aksonowych i dendrytycznych. [KANDEL 1996].



Rysunek 2.2. Klasyczny model neuronu z wejściami x_1, ..., x_n, wyjściem y i stanem wewnętrznym s.

Z punktu widzenia dzisiejszej informatyki ważne jest nie tyle całościowe modelowanie mózgu, ile możliwość wykorzystania sposobów, jakimi on przetwarza informacje. Wiadomo już wiele na temat tych procesów, lecz chęć szybszego pokonania bariery niewiedzy pociąga za sobą próby tworzenia modeli, które tylko częściowo są oparte na odkrytych biologicznych przesłankach. Takie modele czasami pomagają wtórnie w badaniach neurobiologicznych i *vice versa*. Ciekawe z punktu widzenia informatyki jest fakt, że dzięki tym modelom można rozwiązywać zadania, z którymi z trudem radzą sobie inne techniki obliczeniowe. Głównym czynnikiem przemawiającym za praktycznym stosowaniem sieci neuronowych jest ich zdolność do uogólnień zdobytej wiedzy, która daje im jak gdyby pewną dozę inteligencji. Ciekawym i równie ważnym czynnikiem jest to, że sieci neuronowe są wyposażone w swoje wewnętrzne algorytmy przetwarzania informacji, które umożliwiają im rozwiązywanie nawet gatunkowo różnych zadań. Sposób, w jaki sieć neuronowa zyskuje wiedzę o zadanym problemie, polega na nauce na podstawie znanych poprawnych przykładów (zwanych **wzorcami uczącymi**) rozwiązania danego problemu, bądź nawet prościej na bazie obserwacji prezentowanej jej wiedzy. Celem nauki jest swoisty (dla sieci neuronowej) sposób opisania wewnętrznych korelacji zachodzących pomiędzy wzorcami uczącymi. Na tej podstawie nauczona sieć neuronowa potrafi odpowiadać na pytania, zadawane jej z i z poza zakresu wzorców uczących. Do nauki sieci neuronowej mogą zostać zastosowane różne reguły uczące, które są omówione w następnym rozdziale.

2.2. Reguły uczące

Od dawna intrygował człowieka jego własny umysł jako narzędzie zdobywania i gromadzenia wiedzy, czyli uczenia się. Od kiedy nauka zdobyła pewne narzędzia umożliwiające badanie funkcjonowania mózgu, nie ustają próby zmierzające do wyjaśnienia sposobu jego działania i ewentualnego wykorzystania tej wiedzy w systemach technicznych. Na bazie przesłanek płynących z neurobiologii, fizyki, chemii organicznej, psychologii używając modeli matematycznych wielu badaczy próbowało opisać procesy zachodzące w

mózgu istot żywych i zgłębić tajemnice myślenia i zapamiętywania. Na podstawie tych dociekań sformułowano między innymi następujące reguły uczące, które stanowią swoiste źródło wyjścia dla badań procesów pamięciowych i przetwarzania informacji w sieciach neuronowych [ŻURADA 1996, FIESLER 1997, B3.3.1.- 8.]:

Reguła Hebba (*Hebbian rule*) [HEBB 1949] jest zarazem najprostszą i najwcześniej odkrytą regułą uczenia. Jest ona przeniesieniem stwierdzenia z zakresu neurobiologii, które mówi: „Jeżeli akson neuronu A bierze systematycznie udział w pobudzaniu neuronu B powodując jego aktywację, to wywołuje to zmianę metaboliczną w jednym lub obu neuronach, prowadzącą do wzrostu skuteczności pobudzania neuronu B przez neuron A." [HEBB 1949] Najprostsza wersja tej reguły odnosi się do uczenia bez nauczyciela i przyjmuje, że sygnał wyjściowy neuronu jest sygnałem uczącym, czyli przyrost wektora wag wynosi:

$$\Delta w_{ij} = \eta \cdot y_i \cdot x_j$$

przy założeniu, że

$\eta > 0$ jest pewnym współczynnikiem uczenia,

y_i - wyjściem i-tego neuronu,

x_j - j-tym wejściem neuronów.

Nauka polega po prostu na modyfikacji wag pomiędzy każdą parą pobudzonych neuronów. Wynikiem działania tej reguły jest to, że dodatnia wartość składnika korelacyjnego $y_i \cdot x_j$ powoduje wzrost wagi w_{ij}, co w konsekwencji daje silniejszą (autoasocjacyjną) odpowiedź neuronu przy kolejnej próbie pobudzenia tym samym wzorcem wejściowym. Wzorce często powtarzające się na wejściu sieci dają więc najsilniejszą odpowiedź na jej wyjściu. Reguła ta jest więc często wykorzystywana w sieciach autoasocjacyjnych. Stosowanie reguły Hebba w czystej postaci powoduje nieskończony wzrost wag, więc w praktycznych realizacjach stosuje się często pewien współczynnik normalizujący przeciwdziałający temu nieograniczonemu wzrostowi. Wagi są zazwyczaj aktualizowane po każdym wzorcu uczącym (*on-line training*). Istnieje wiele rozszerzeń i modyfikacji reguły Hebba, która w efekcie może być stosowana do uczenia z nauczycielem (*supervised training*) i do uczenia bez nauczyciela (*unsupervised training*). Dobrze się sprawuje zarówno dla wzorców binarnych jak i bipolranych. W jednej z wariacji tej reguły (*neo-Hebbian learning*) wprowadzono również zdolność sieci do zapominania [KOSKO 1992].

Reguła perceptronowa (*perceptron rule*) [ROSENBLATT 1961] dotyczy nauki z nauczycielem i domyślnie zakłada warstwową architekturę sieci neuronowej. Sieć neuronowa oparta na klasycznym modelu perceptronu składa się z warstwy wejściowej zawierającej pewną ilość neuronów o progowej funkcji aktywacji (*linear threshold activation function*) oraz jednego neuronu w warstwie wyjściowej, której wyjście może być bipolarne lub binarne. Progowa funkcja aktywacji ma dla zadanego progu θ następującą postać:

$$f(z) = \begin{cases} 1 & for\ z \geq \theta \\ 0 & for\ z < \theta \end{cases}$$

Korekcja wag dla danego współczynnika uczenia $\eta > 0$ odbywa się według następującej zależności:

$$\Delta w_{ij} = \eta(t_i - y_i)x_j,$$

gdzie

t_i jest i-tym sygnałem uczącym dla i-tego wyjścia y_i,

x_j jest j-tym wejściem.

Reguła ta charakteryzuje się tym, że gdy wyjście i-tego neuronu y_i jest równe jego wartości pożądanej t_i pochodzącej od nauczyciela, zmiana wagi jest zerowa. W nawiązaniu do tej właściwości istnieje twierdzenie o zbieżności tej reguły, które mówi, że jeśli istnieje zbiór wag, który pozwala dawać perceptronowi poprawną odpowiedź dla wszystkich wzorców uczących, wtedy metoda ucząca oparta na tej regule znajdzie pożądany zbiór wag w skończonej liczbie iteracji.

Aktualizacja wag następuje zwykle po każdym wzorcu uczącym (*on-line training*) tak jak w przypadku reguły Hebba. Jakkolwiek reguła ta daje większe możliwości uczenia niż reguła Hebba, możliwość jej stosowania ogranicza się do problemów liniowo separowalnych, dla których istnieje taka hiperpłaszczyzna, dla której wszystkie punkty znajdujące się po jednej stronie tej hiperpłaszczyzny przyjmują jedną wartość funkcji, a punkty znajdujące się po jej drugiej stronie drugą wartość tej funkcji. Jeżeli sieć neuronowa składa się z perceptronów uformowanych jednowarstwowo, architekturę taką nazywany skrótowo **SLP** (*single-layer perceptron*), zaś gdy tworzą architekturę wielowarstwową nazywamy **MLP** (*multi-layer perceptron*).

Reguła delta (*delta rule, least-mean-square (**LMS**) rule*) [WIDROW & HOFF 1960] umożliwia rozwiązywanie szerokiej gamy problemów ze względu na możliwość operowania na ciągłych i na dyskretnych wejściach, Natomiast neurony posiadają ciągłe funkcje aktywacji. Reguła ta należy do grupy metod uczenia z nauczycielem, a aktualizacja wag następuje dla pewnego ustalonego $\eta > 0$ według następującej zależności:

$$\Delta w_{ij} = \eta(t_i - net_i)x_j,$$

gdzie:

$$net_i = w_{i0} + \sum_j w_{ij}x_{ij}, \qquad w_{i0} \text{ - szum (}bias\text{)}$$

Wagi mogą zostać zainicjowane dowolnie. Nauka sieci w oparciu o regułę delta trwa dopóki wagi sieci nie przyjmą takich wartości, że błąd średniokwadratowy (*least-mean-square*):

$$E(w) = \frac{1}{2}\sum_{j=1}^{k}(t_j - net_j)^2$$

jest zminimalizowany dla wszystkich wzorców uczących (j = 1, ..., k). Aktualizacja wag tą metodą może być przeprowadzana zarówno po każdym wzorcu uczącym (*on-line training*) jak i po przejrzeniu całego ciągu uczącego (*off-line training*). Model o pojedynczym wyjściu opartym o liniowy element adaptacyjny został nazwany ***adaline***, a jego rozszerzenie na wiele takich neuronów w warstwie wyjściowej ***madaline*** (*many adalines*).

Uogólniona reguła delta nazywana również **regułą delta** była zaproponowana przez kilku badaczy takich jak Werbos, Parker, Le Cun and Rumelhart [RUMMELHART & MCCLELLAND 1986]. Reguła ta jest zazwyczaj omawiana w powiązaniu z popularną kompletną metodą uczenia znaną pod nazwą ***backpropagation*** (metoda propagacji wstecznej). Metoda ta działa w oparciu o metody gradientowe, które dokonują małych kroków w przestrzeni wag w kierunku spadku gradientu. W przypadku tej metody i innych podobnych wymaga się, by funkcja aktywacji f była monotonicznie rosnąca i różniczkowalna. Zazwyczaj wykorzystuje się uni- lub bipolarne funkcje sigmoidalne, które spełniają w/w kryteria. Poprawka wag następuje według następującej zależności:

$$\Delta w_{ij} = \eta[t_i - f(net_i)]f'(net_i)x_j.$$

Powyższa formuła charakteryzuje się tym, że oprócz zatrzymania procesu zmian wag dla $t_i = f(net_i) = y_i$, zmiana wagi jest dodatkowo hamowana, gdy funkcja aktywacji f jest płaska w punkcie net_i. Naturalną konsekwencją zastosowania pochodnej w tej formule jest fakt, że gdy funkcja f osiągnie minimum dla pewnego net_i, zmiana wagi w tym punkcie będzie równa zeru. Taka sytuacja sprawia, że reguła ta w naturalny sposób nie opuszcza osiągniętego minimum (lokalnego lub globalnego), co ma swoje zalety, lecz również wady. Zastosowanie sigmoidalnej funkcji aktywacji powoduje, że dla dużych bezwzględnych wartości net_i wartość pochodnej w tym punkcie jest bliska zeru, a więc i zmiany wag są znikome, mimo że zamierzony cel nie został jeszcze osiągnięty. Następnym problemem tej metody jest to, że algorytm może zostać przerwany w minimum lokalnym zamiast minimum globalnym. Ponadto metoda ta jest narażona na problem wolnej zbieżności, który jest charakterystyczny dla wszystkich metod gradientowych. Korekta wag dla neuronów l-tej warstwy ukrytej jest obliczana następująco:

$$\Delta w_{l,ij} = \left[\sum_j \left(w_{l+1,ij} \cdot \Delta w_{l+1,j} \right) \right] f'(net_{l,i}) x_{lj} .$$

Cechą wspólną dla obydwu powyższych formuł korekty wag jest osiągnięcie poprawki błędu dopasowanej do wagi w stopniu, w jakim przyczyniła się do jego powstania.

Reguła Kohonena (*Kohonen rule*) [KOHONEN 1984] jest typową regułą uczenia nie nadzorowanego i dotyczy w szczególności nauki z rywalizacją (*competitive learning*), w której każda grupa neuronów bierze udział w rywalizacji. Teuvo Kohonen wynalazł regułę uczenia, która pozwala sieci neuronowej zorganizować się poprzez wybranie najbardziej reprezentatywnych neuronów. Takie sieci nazywają się sieciami samoorganizującymi się (*self-organizing networks*). Jedną z najsurowszych strategii opartych na rywalizacji jest kryterium „wygrywający bierze wszystko" (*winner take all*), które powoduje aktualizację tylko wag neuronu, który się cechuje największą wartością aktywacji net_i. Niestety często się zdarza, że neurony, które w wyniku wstępnej inicjalizacji są dalekie od danych wzorcowych, nigdy nie wygrywają i stąd nic się nie nauczą. Często spotykaną wariacją tego kryterium jest modyfikacja polegająca na tym, że oprócz wag zwycięzcy, są aktualizowane również wagi neuronów sąsiednich z dokładnością do pewnego współczynnika wyrażającego pewną ich geometryczną odległość od zwycięzcy.

Reguła gwiazdy wyjść (outstar rule) [GROSSBERG 1982] związana z pojęciami ***instar*** (gwiazda wejść) i ***outstar*** (gwiazda wyjść) określające zachowanie się neuronów. Gwiazda wejść odnosi się do neuronu, który otrzymuje (poprzez swoje dendryty) wejścia od wielu innych neuronów, zaś gwiazda wyjść odnosi się do neuronów, które wysyłają (poprzez swoje aksony) wyjścia do wielu innych neuronów sieci neuronowej.

Uczenie gwiazdy wejść jest nie dozorowane i związane z dostrajaniem wag połączeń w celu dopasowania się do wektora wejściowego. To może być przeprowadzone np. wykorzystując regułę Kohonena. Neurony gwiazdy wejść produkują wyjście, jeżeli pojawi się odpowiedni wektor na wejściu sieci. Z drugiej strony, gwiazda wyjść jeżeli jest pobudzona produkuje odpowiedni wzorzec, w celu wysłania go do innych neuronów. Stąd wynika nadzorowany sposób nauki. Jedynym sposobem na osiągnięcie uczenia gwiazdy wyjść jest dostrojenie jej wag do pożądanego wektora. Poprawka wag jest zdefiniowana dla malejącego współczynnika uczenia $\beta > 0$ następująco:

$$\Delta w_{ji} = \beta\left(t_j - w_{ji}\right).$$

Regułę gwiazdy wyjść używa się do uczenia powtarzających się właściwości relacji wejście-wyjście. Mimo że uczenie odbywa się z nauczycielem, od sieci oczekuje się zdolności wydobywania statycznych cech sygnałów wejściowych i wyjściowych.

Przedstawione reguły uczące stanowią zazwyczaj punkt wyjścia dla wielu różnych metod uczenia. Posiadają swoje zalety i wady, jednak żadna z nich nie rozwiązuje wszystkich problemów w całej swojej rozpiętości. W powyższych regułach skupiono się na sposobie modyfikacji wag. Jednak ten proces może zachodzić pomyślnie tylko przy pewnych dodatkowych założeniach odnośnie właściwej architektury sieci i jej odpowiedniej inicjalizacji. Z praktycznych prób implementacji różnych systemów w oparciu o sieci neuronowe wiadomo, że stosowanie tych sieci nie sprowadza się tylko do problemu ich nauki. Najpierw trzeba mieć stosowną sieć, którą można uczyć. Podejmowane są różne próby konstruowania bądź optymalizacji architektury sieci na bazie algorytmów genetycznych, metod ontogenicznych lub technik oczyszczających sieć ze zbędnych połączeń (*pruning*). Metody te są jednak bardzo czasochłonne i nie zapewniają otrzymania optymalnej architektury sieci neuronowej dla danego zadania. Nadal więc istnieje potrzeba szukania nowych metod uczenia jak również nowych sposobów konstruowania sieci neuronowych. Właściwym kierunkiem zdaje się być poszukiwanie dedykowanych metod do efektywniejszego rozwiązywania pewnych grup problemów. Jeśli sieci neuronowe mają się

stać narzędziem codziennego użytku, konieczne jest opracowanie kompleksowych metod ich konstruowania i uczenia tak, by sukces ich stosowania nie był zależny od czynników losowych. Dopiero wtedy sieci te mogą stać się godne zaufania i stosowania w dużych systemach informatycznych.

2.3. Podsumowanie

Podany wyżej przegląd informacji na temat struktury i metod uczenia obecnie używanych sieci neuronowych z pewnością nie jest pełny ani wyczerpujący. Nie było zresztą możliwe danie takiego wyczerpującego opisu w jednym (pomocniczym) rozdziale tej pracy doktorskiej, ponieważ przy niesłychanie bogatej wiedzy naukowej, jaką obecnie zgromadzono na temat mózgu i na temat działania jego technicznych modeli, czyli sieci neuronowych – dla wyczerpującej prezentacji tego tematu konieczne było by napisanie kilkutomowej monografii.

Zadaniem tego rozdziału było jednak wyłącznie zarysowanie tła dla pewnych problemów wymagających rozwiązania, które to rozwiązanie będzie poszukiwane w tej właśnie pracy doktorskiej. Z podanego przeglądu osiągnięć neurocybernetyki wynika bowiem wyraźnie, że mimo ogromnego postępu wiedzy o mózgu i mimo opracowania wielu bardzo skutecznych metod uczenia sieci neuronowych – wciąż jeszcze potrzebne są nowe pomysły i nowe metody pozwalające na budowę systemów neurocybernetycznych, które były by dostosowane do specyfiki konkretnego zadania obliczeniowego, jakie chcemy rozwiązać. Właśnie poszukiwanie takich nowych metod formowania „na zamówienie” struktury potrzebnej sieci neuronowej oraz dobierania jej parametrów w sposób szybszy, niż pozwalają to robić klasyczne metody uczenia – poświęcone będą dalsze rozdziały, aż do końca pracy.

Nadrzędną myślą, kierującą usiłowaniami autora było dopasowywanie narzędzia (sieci neuronowej) do właściwości zadania. Stąd metody tworzenia i uczenia sieci neuronowych opisane w pracy będą mniej uniwersalne, niż wiele innych koncepcji, opisywanych w literaturze. Jednak cechą opracowanych technik jest ich bardzo wysoka sprawność w odniesieniu do zadań ściśle określonego rodzaju. Takie właśnie nastawienie i taki cel sygnalizuje między innymi nazwa programu opracowanego podczas realizacji pracy, który nazwano „*Brain for problem”*.

3. METODA AUTOMATYCZNEJ KONFIGURACJI SIECI NEURONOWYCH DLA PROBLEMÓW ROZPOZNAWANIA WZORCÓW BINARNYCH

Przedstawiona w tym rozdziale metoda służy do automatycznej konfiguracji sieci neuronowych dla problemów rozpoznawania wzorców binarnych. Metoda ta stosuje odmienne podejście zarówno do procesu określania architektury sieci neuronowej jak również do procesu jej konfiguracji. Metoda ta nie bazuje na procesie uczenia, jak jest to w przypadku większości stosowanych obecnie metod, lecz na procesie formowanie struktury sieci i jej konfiguracji w oparciu o analizę ciągu uczącego i wyliczenie żądanego zbioru cech. Dzięki temu ominięty został cały proces iteracyjnego dostosowywania parametrów sieci, co pozwoliło znacznie zredukować czas niezbędny do przygotowania systemu rozpoznającego wzorce binarne – gdy tego rodzaju system jest nam właśnie potrzebny.

3.1. Założenia metody

Zaproponowana w pracy metoda w oryginalny sposób próbuje zmierzyć się z problemem doboru optymalnej architektury sieci neuronowej dla konkretnego problemu, zadanego ciągiem uczącym. Dobór architektur sieci odbywa się na drodze redukcji synaps według zadanych kryteriów w taki sposób, aby jakość uzyskanego rozpoznawania i uogólnienia uległa tylko minimalnemu pogorszeniu. Dodatkowo taka redukcja wpływa pokaźnie na obniżenie kosztów ewentualnej implementacji konstruowanego systemu rozpoznawania. Ponadto opisana metoda potrafi bardzo efektywnie skonfigurować sieć neuronową eliminując w zupełności długotrwały i żmudny proces nauki sieci, omijając zarazem trudności związane z nauką, jak np. problem minimów lokalnych. Proces konfiguracji sieci dokonywany jest w oparciu o analizę wszystkich wzorców wchodzących w skład zbioru uczącego. Analiza ta służy do wydobywania i ustalenia wartości specjalnego zbioru cech, który stanowi podstawę obliczania wag synaptycznych.

Warto podkreślić, że cały proces doboru architektury sieci neuronowej oraz konfiguracji odbywa się całkowicie automatycznie i z wysoką sprawnością obliczeniową. Zarówno struktura rozpoznającej sieci neuronowej jak i wartości wszystkich występujących w niej współczynników wagowych zostają ustalone w procesie zaledwie dwukrotnego przeglądnięcia rozważanej grupy rozpoznawanych wzorców, co powoduje, że zaproponowana metoda osiąga optymalne uformowanie struktury i parametrów sieci dostosowanej do

rozważanego zadania rozpoznawania w nieporównywalnie szybszy sposób, niż wszystkie inne znane metody uczenia sieci neuronowych. Dzieje się tak dlatego, iż opracowana metoda nie wymaga iteracyjnej optymalizacji parametrów sieci tak, jak to robią inne techniki uczenia sieci. W pracy rozważany jest również problem oszacowania jakości uogólniania i jakości rozpoznawania otrzymanej sieci. Problem oszacowania wyżej wymienionych jakości jest o tyle ważny, że w praktycznych zastosowaniach decyduje o zastosowaniu bądź odrzuceniu danej metody. Opisana w pracy metoda pozwala automatycznie obliczyć jakość rozpoznawania i jakość uogólniania, co pozwala na podjęcie decyzji o stopniu redukcji synaps w zależności od wymagań konstruowanego systemu.

3.2. Opis metody

3.2.1. Metoda estymacji cech wzorców binarnych

Cały proces doboru architektury sieci neuronowej oraz jej konfiguracji rozpoczyna się od etapu wartościowania cech binarnych poszczególnych wzorców wchodzących w skład ciągu uczącego. Dla potrzeb określenia cech binarnych konieczne jest wyznaczenie dwóch pomocniczych macierzy T i F, których wymiar (I, J) odpowiada wymiarowi wzorców uczących. Macierze te są zdefiniowane następująco:

$$\forall i, j \quad T[i, j] = \#\left(P_k[i, j] = true : k = 1,..., N\right)$$

$$\forall i, j \quad F[i, j] = \#\left(P_k[i, j] = false : k = 1,..., N\right)$$

gdzie

N – ilość wzorców ciągu uczącego

i = 1, ..., I

j = 1, ..., J

Poszczególne pola tych macierzy w myśl ich definicji zawierają odpowiednio ilości pól prawdziwych (dla macierzy T) i fałszywych (dla macierzy F) wszystkich wzorców uczących z uwzględnieniem ich pozycji w strukturze danej macierzy. Prawda jest przekładana na wartość +1, a fałsz na wartość –1.

Następnie bazując na tych dwóch macierzach i na ilości wzorców ciągu uczącego można obliczyć binarny estymator cechy E_k dla wszystkich pól każdego wzorca uczącego w następujący sposób:

$$\forall k = 1,..., N \quad \forall i, j \quad E_k[i, j] = \begin{cases} \dfrac{+1}{T[i, j]} & if \;\; P_k[i, j] = true \\ \dfrac{-1}{F[i, j]} & if \;\; P_k[i, j] = false \end{cases}$$

gdzie

$$\forall k = 1,..., N \quad \forall i, j \quad E_k[i, j] \in \left\langle -1, -\frac{1}{N} \right\rangle or \left\langle -\frac{1}{N}, -1 \right\rangle$$

Im większa bezwzględna wartość $E_k[i,j]$ dla danego pola macierzy rozważanego wzorca tym większe znaczenie ma to pole podczas procesu rozpoznawania dla prawidłowego rozpoznania danego wzorca, ponieważ większa bezwzględna wartość świadczy o większej unikalności wartości danego pola macierzy z punktu widzenia całego ciągu uczącego. Dla każdej macierzy tworzącej dany wzorzec można określić taką grupę jej pól, których wartości $E_k[i,j]$ są duże, czyli pól, które z punktu widzenia zadania rozpoznawania dobrze charakteryzują dany wzorzec na tle całego ciągu uczącego. Dla każdego wzorca uczącego może zostać w ten sposób wyznaczona taka grupa cech dobrze wyróżniająca go spośród innych zgodnie z przyjętymi jednolitymi kryteriami. Te grupy stanowią fundament dla następnych obliczeń, mianowicie dla redukcji synaps oraz dla obliczania wag synaptycznych.

3.2.2. Jakość rozpoznawania i jakość uogólniania

Problem redukcji synaps jest w ogólności bardzo złożony i ściśle związany z jakością rozpoznawania i jakością uogólniania, jaką wynikowa sieć neuronowa będzie się charakteryzować [FIESLER 1997, B3.5]. Redukcja synaps prowadzi zazwyczaj do zubożenia informacji jakie sieć posiada o danym problemie i prowadzi do większych lub mniejszych błędów rozpoznawania i uogólniania wzorców uczących. Można więc postawić następujące pytanie: Jak można maksymalnie zredukować ilość synaps dla zadanego problemu, żeby zarazem utrzymać jakość rozpoznawania i jakość uogólniania wzorców na akceptowalnym poziomie? W ogólności odpowiedź na to pytanie jest trudna. W tej pracy podjęto próbę zdefiniowania jakości rozpoznawania i jakości uogólniania dla rozważanego problemu rozpoznawania wzorców binarnych. Wartości tych jakości można obliczyć automatycznie i każdorazowo dla danej zredukowanej architektury sieci neuronowej. Wyznaczenie tych wartości pozwala podjąć decyzję o stopniu redukcji synaps sieci, co pozwala indywidualnie dobrać parametry redukcji synaps dla zadanego problemu i konkretnych oczekiwać stawianych sieci neuronowej. Dzięki temu znany jest stopień ufności, jakim możemy daną sieć neuronową obdarzyć. Podobnej cechy nie mają z reguły sieci uczone z wykorzystaniem typowych algorytmów.

W myśl definicji estymatora cech opisanego w rozdziale 3.2.1. każde pole macierzy cech każdego wzorca uczącego niesie jakąś informację w większym lub w mniejszym stopniu istotną dla zadania rozpoznawania. W związku z tym można przyjąć, że dla rozważanej grupy problemu i rozważanej grupy architektur sieciowych maksymalną jakość rozpoznawania i uogólniania uzyskiwać będziemy dla pełnej nie zredukowanej architektury sieci neuronowej. Taka pełna nie zredukowana architektura sieci będzie stanowić dla danego problemu punkt odniesienia oraz źródło miary i oceny stopnia utraty jakości rozpoznawania i jakości uogólniania dla konkretnej przeprowadzonej redukcji synaps. W związku z powyższym jakość rozpoznawania Q_R i jakość uogólniania (generalizacji) Q_G zostaną zdefiniowane następująco:

$$Q_R = \frac{\min\limits_{k=1,\ldots,N}\left(\min\limits_{p=1,\ldots,N\,\&\,p\neq l}\left(\max\limits_{l=1,\ldots,N}\left|Out_k^R[l]\right| - \left|Out_k^R[p]\right|\right)\right)}{\min\limits_{k=1,\ldots,N}\left(\min\limits_{p=1,\ldots,N\,\&\,p\neq l}\left(\max\limits_{l=1,\ldots,N}\left|Out_k^F[l]\right| - \left|Out_k^F[p]\right|\right)\right)},$$

$$Q_G = \frac{\underset{k=1,\ldots,N}{\text{average}}\left(\min_{p=1,\ldots,N\,\&\,p\neq l}\left(\max_{l=1,\ldots,N}\left|Out_k^R[l]\right| - \left|Out_k^R[p]\right|\right)\right)}{\underset{k=1,\ldots,N}{\text{average}}\left(\min_{p=1,\ldots,N\,\&\,p\neq l}\left(\max_{l=1,\ldots,N}\left|Out_k^F[l]\right| - \left|Out_k^F[p]\right|\right)\right)},$$

gdzie

$Out_k^R[l]$ oznacza l-te wyjście sieci neuronowej o zredukowanej architekturze powstałe w wyniku pobudzenia k-tym wzorcem uczącym,

$Out_k^F[l]$ oznacza l-te wyjście sieci neuronowej o pełnej nie zredukowanej architekturze powstałe w wyniku pobudzenia k-tym wzorcem uczącym.

Z wzorów tych wynika, że dla jednoznacznego rozpoznania wszystkich wzorców uczących potrzeba i wystarcza, żeby wartość $Q_R > 0$. Z kolei jeśli chodzi o wartość Q_G określającą jakość uogólniania (generalizacji) wymagania są bardziej rygorystyczne. Ze zrozumiałych względów jakość uzyskanego uogólnienia na wzorce z poza ciągu uczącego (i związana z nią ufność wobec sieci) będzie tym większa im większą wartość Q_G dla konkretnej wynikowej sieci neuronowej uzyskamy i już od konkretnego zadania zależy, na ile wartość Q_G można obniżyć, a w związku z tym, jak bardzo można architekturę sieci neuronowej zredukować.

3.2.3. Kryteria redukcji synaps

Redukcja synaps jest celowa ze względu na niższy koszt realizacji sieci zawierającej mniejszą liczbę elementów składowych. Proces redukcji jest jednak zawsze pewnym kompromisem pomiędzy uzyskaną jakością rozpoznawania i jakością uogólniania dla wynikowej sieci neuronowej. Dlatego przed przystąpieniem do redukcji synaps ważne jest ustalenie, jak dokładnych odpowiedzi od sieci oczekujemy. Odpowiedź na to pytanie pozwoli tak dobrać odpowiednie parametry redukcji, aby sieć neuronowa spełniała oczekiwania jej konstruktora i by w satysfakcjonujący sposób rozwiązywała postawione jej zadanie.

W tej pracy zdefiniowano trzy kryteria redukcji synaps pozwalające w różny sposób wpłynąć na proces redukcji synaps przypisując odpowiednio większą wagę różnym parametrom wynikającym z wyznaczonych zbiorów cech dla poszczególnych wzorców uczących. Mianowicie:

1. Kryterium maksymalnych cech ($C_F \in \langle 0{,}100 \rangle\%$)

Kryterium to wyznacza granicę wartości cech E_{Min}, powyżej której synapsy odpowiadające tym cechom nie mogą zostać zredukowane.

$$E_{Min} = \frac{1}{1 + (N-1) \cdot C_F / 100}$$

Kryterium maksymalnych cech redukuje ilość synaps dla poszczególnych wzorców uczących nierównomiernie, uwzględniając przy redukcji tylko wielkość cechy, lecz zarazem dbając o to, by najistotniejsze cechy dla każdego wzorca nie zostały zredukowane.

2. Kryterium minimalnej ilości synaps ($C_N \in \langle 0{,}100 \rangle \%$)

Kryterium to określa, jaka minimalna ilość synaps N_{Min} musi po redukcji dla każdego wzorca uczącego w sieci neuronowej występować.

$$N_{Min} = I \cdot J \cdot C_N / 100$$

Kryterium minimalnej ilości synaps zapewnia to, że każdy wzorzec niezależnie od wielkości przypisanych mu cech dla poszczególnych jego pól będzie w sieci reprezentowany odpowiednią liczbą synaps i nie dojdzie do sytuacji, że jakaś bardzo unikatowa cecha pewnego wzorca zdominuje cały proces redukcji synaps dla danego wzorca uczącego, co w pewnych sytuacjach może nie być korzystne w zależności od specyfiki rozwiązywanego przez sieć zadania rozpoznawania.

3. Kryterium minimalnej precyzji rozpoznawania ($C_P \in \langle 0{,}100 \rangle \%$)

Kryterium to mówi, jaka minimalna suma wyznaczonych cech P_{Min} musi być dla każdego wzorca uczącego w sieci reprezentowana.

$$P_{Min} = \sum_{i,j} \left| E_k[i,j] \right| \cdot C_P / 100$$

Kryterium minimalnej precyzji rozpoznawania stanowi gwarancję tego, że dla każdego wzorca uczącego pewna minimalna suma jego najistotniejszych cech będzie zawsze w sieci reprezentowana niezależnie od rozkładu ich wielkości i ilości jego cech większych i mniejszych. Taka minimalna reprezentacja sumy najbardziej unikalnych cech wzorców jest ściśle związana z minimalną precyzją ich rozpoznawania na tle innych wzorców ciągu uczącego.

Wszystkie powyżej opisane kryteria odgrywają pewną istotną rolę w procesie redukcji synaps. Każde z nich odpowiada za zachowanie pewnych specyficznych własności wzorców uczących na podstawie indywidualnych wielkości cech dla poszczególnych wzorców uczących. Wszystkie trzy kryteria mogą być używane łącznie w różnych konfiguracjach,

wymuszając łącznie odpowiednio zaplanowany proces redukcji uwzględniając ustalone kryteria według specyfiki rozwiązywanego zadania rozpoznawania. Dzięki łączności stosowania wszystkich wymienionych kryteriów można w prosty sposób zdefiniować ogólną charakterystykę przyjętej zasady redukcji synaps. Charakterystyka ta będzie dalej oznaczana symbolem *FNP (Feature, Number, Precision*) i dla rozważnej klasy architektur sieciowych będzie zapisywana następująco:

$FNP = C_F|C_N|C_P$,

więc np. gdy *FNP* = 52|28|47 oznacza to, że $C_F = 52\%$, $C_N = 28\%$ i $C_P = 47\%$.

Na podstawie zadanej *FNP* charakterystyki redukcji synaps architektura sieci neuronowej zostanie automatycznie skonstruowana, mało istotne synapsy z punktu widzenia opisanych wcześniej kryteriów będą zredukowane, a pozostałe nie zredukowane synapsy skonfigurowane poprzez odpowiednie nadanie wartości ich wagom na podstawie określonych wcześniej cech odpowiadającym tym synapsom. Charakterystyka redukcji synaps *FNP* w sposób całkowicie jednoznaczny wyznacza architekturę i całą konfigurację sieci neuronowej. Z tego też względu nie ma potrzeby zapamiętywania skonfigurowanej sieci, jeśli pamiętany jest ciąg uczący, albowiem dla danego ciągu uczącego i danej charakterystyki *FNP* zawsze zostanie wygenerowane taka sama sieć neuronowa. Dodatkowo z daną charakterystyką *FNP* i z danym ciągiem uczącym są ściśle związane wartości jakości rozpoznawania Q_R i jakości uogólniania Q_G, które też są jednoznacznie determinowane przez wartość *FNP*.

3.2.4. Automatyczna konfiguracja sieci neuronowej

Cały proces automatycznej konfiguracji sieci neuronowej włącznie z ustaleniem jej oszczędnej zredukowanej architektury przebiega w trakcie dwukrotnego przeglądnięcia ciągu uczącego następująco:

W 1. przeglądnięciu ciągu uczącego:

- Wyznaczane są macierze T i F.

W 2. przeglądnięciu ciągu uczącego dla każdego wzorca uczącego wykonywane są kolejno następujące czynności:

- Obliczenie macierzy estymatora cech E_k.
- Redukcja synaps na podstawie charakterystyki *FNP* związana z zerowaniem tych wartości E_k[i,j], które nie spełniają stawianych wymogów.

- Ustalanie wag dla nie zredukowanych synaps, które spełniają charakterystykę *FNP*.

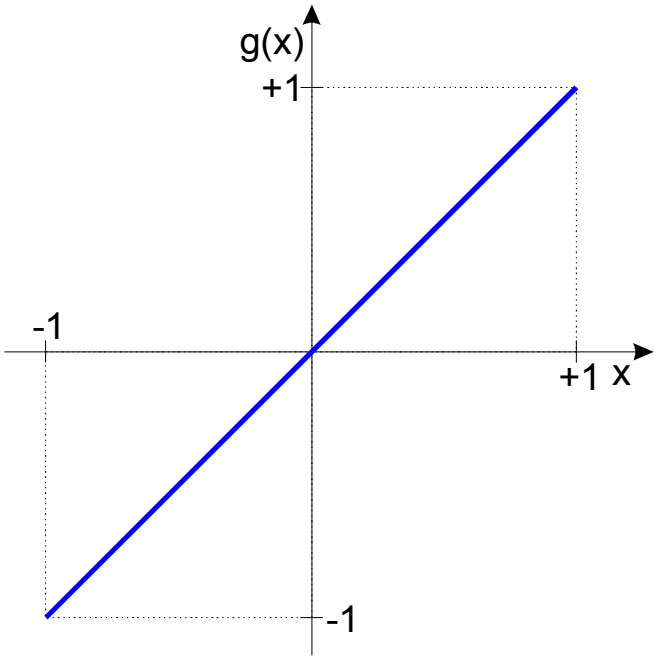
Ostatnią czynnością, jaka została do zrobienia jest obliczenie wag synaptycznych na podstawie wszystkich nie wyzerowanych (po etapie redukcji) wartości E_k[i,j] w następujący sposób:

$$W_k[i,j] = \frac{E_k[i,j]}{\sum_{m,n} |\, E_k[m,n]\,|}$$

W wyniku powyższych obliczeń wszystkie wartości wyjść sieci neuronowej są znormalizowane do przedziału domkniętego $\langle -1,+1 \rangle$, co sprawia, że maksymalna wartość określająca rozpoznanie wzorca pochodzącego z ciągu uczącego ma wartość prawdy (+1), zaś wartość negatywu wzorca uczącego wartość fałszu (-1). Wszystkie zaburzone wzorce pochodzące z poza ciągu uczącego przyjmują różne wartości z przedziału $\langle -1,+1 \rangle$ w zależności od stopnia ich podobieństwa do poszczególnych wzorców uczących.

3.2.5. Porównywalność uzyskanych wyników

Otrzymane wyniki rozpoznawania są zwykle szacowane i porównywane przez człowieka, który dokonuje tego, wykorzystując intuicyjną miarę liniową. Ponadto klasyfikacja negatywów wzorców uczących powinna również w sposobie ich klasyfikacji odpowiadać intuicji, tzn. negatywy powinny zostać rozpoznane jako odpowiadające im pozytywy z zaznaczeniem ich odwrotności. Z opisanych powyżej powodów funkcja aktywacji neuronów została zdefiniowana tak, jak to pokazano na rys. 3.1.:



Rysunek 3.1. Funkcja aktywacji neuronów: g*(x) = x*; *Dom g* $\in$ *[-1,1]*, *Im g* $\in$ *[-1,1]*.

Funkcja przedstawiona na rys. 3.1. mimo swojej prostoty doskonale odzwierciedla jak liniową miarę używaną intuicyjnie do porównań, tak odpowiednią klasyfikację negatywów wzorców uczących nadając im poprawną bezwzględną wartość ich rozpoznania dodatkowo oznaczając negatywy znakiem minus „-”.

3.3. Zalety i wady metody

W zaprezentowanej koncepcji automatycznej konfiguracji sieci neuronowych dla wzorców binarnych można dostrzec wiele zalet, do których należą przede wszystkim:

- Zautomatyzowany proces doboru architektury sieci przy uwzględnieniu kryteriów redukcji odnoszących się do mało ważnych synaps z punktu widzenia poszczególnych wzorców uczących.
- W pełni automatyczna konfiguracja sieci neuronowej dla zadanego problemu.
- Wysoka sprawność obliczeniowa związana z koniecznością zaledwie dwukrotnego przeglądnięcia ciągu uczącego.
- Wyeliminowanie procesu uczącego wymagającego iteracyjnej optymalizacji parametrów związanego z wieloma problemami, np. problemu minimów lokalnych.
- Prosta, intuicyjna, liniowa interpretacja miary podobieństwa rozpoznawanych wzorców względem wzorców uczących.

- Właściwe odwzorowanie negatywów wzorców uczących.
- Możliwość oceny jakości uzyskanego uogólnienia oraz sprawdzenie, czy jakość rozpoznawania jest zachowana na wymaganym niezerowym poziomie.

Wśród zalet metody można też wskazać, że możliwe jest w niej rozpoznawanie wzorców, które nie zawierają wartości z założenia w pełni prawdziwe, bądź też fałszywe, ale charakteryzujące się pewnym stopniem wiarygodności w przełożeniu na pewne wartości pośrednie pomiędzy prawdą a fałszem.

Metoda ta ma także swoje wady i ograniczenia, do których można przede wszystkim zaliczyć:

- Metoda ta silnie bazuje na pozycji poszczególnych pól w macierzy wzorców uczących i w związku z tym jest narażona (szczególnie w przypadku rozpoznawania obrazów) na przesunięcia, rotacje i skalowania. Problem ten może być częściowo rozwiązany w przypadku, gdy owe przesunięcia, rotacje i skalowania nie są zbyt duże, poprzez zastosowanie grubszego rastra, który może zamortyzować drobne niezgodności.
- Innym podstawowym ograniczeniem metody jest to, iż działa ona tylko na wzorcach binarnych, które czasami nie dają możliwości zapisu pewnych zależności charakteryzujących się pewną ciągłą zależnością.
- Wynikiem działania tej metody jest sieć neuronowa zawierająca tyle neuronów ile jest wzorców uczących, co w przypadku dużych zbiorów danych może być pewnym ograniczeniem ze względu na ograniczenia pamięciowe, mimo że ilość alokowanej pamięci dla jednego neuronu sieci w przypadku tej metody jest niewielka

Chociaż opisana metoda nie pozwala na rozwiązanie w ogólności wszystkich stawianych jej problemów, z powodzeniem można ją stosować w wielu różnych aplikacjach ze względu na szybkość jej działania, zdolność automatycznego doboru architektury sieci oraz intuicyjnie prostą interpretację otrzymanych wyników. Metoda ta jest również godna polecenia ze względu na możliwość oceny uzyskanego uogólnienia, co wiąże z tą metodą wyższy stopień wiarygodności w porównaniu do metod, które co prawda dają pewne uogólnienie, ale jest ono niepewne, niemierzalne i czasami nawet niezgodne z intuicją.

3.4. Kierunki rozwoju metody

Metoda ta stanowi pewne wprowadzenie do badań związanych z automatyczną konfiguracją sieci neuronowych na bazie wcześniejszej analizy wzorców uczących. W

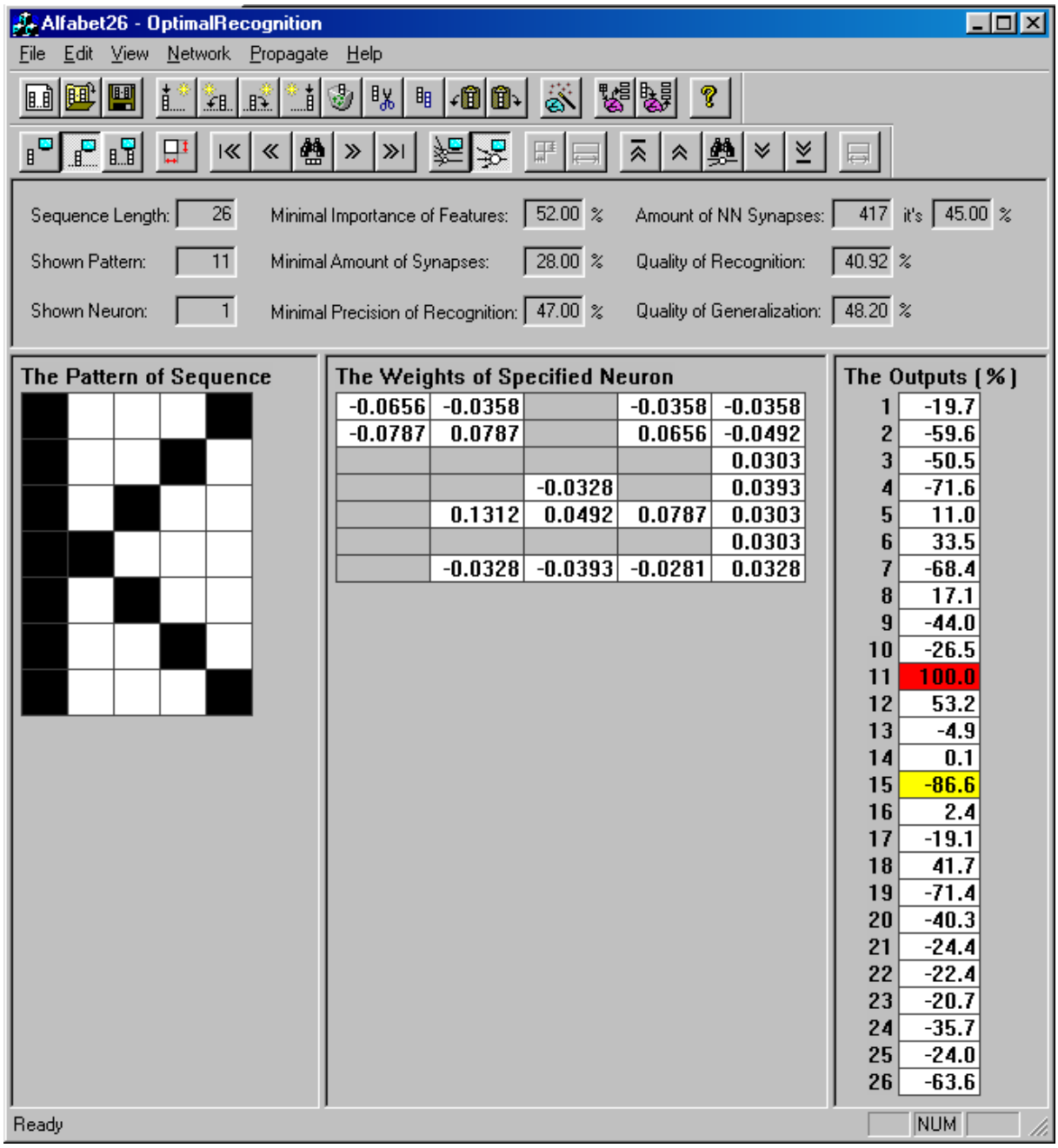
przygotowaniu znajduje się uogólnienie opisanej metody umożliwiające jeszcze większą kompresję informacji pamiętanych w sieci. Kompresja ta polegać będzie na przyporządkowaniu jednego lub kilku wyjściowych neuronów klasyfikujących wielu wzorcom wejściowym należącym do jednej klasy, czyli odpowiedniość N : K, gdzie K << N, co znacząco zmniejszy rozmiar pamięci potrzebnej na ewentualną implementację systemu rozpoznawania. Dodatkowo metoda umożliwiać będzie automatyczną konfigurację sieci dla danych z ciągłego przedziału pomiędzy prawdą i fałszem. Usprawniona metoda dysponować będzie też możliwością korzystania z niepełnych danych uczących. Rozważane jest również uogólnienie automatycznej konfiguracji na sieci wielowarstwowe oparte na perceptronach (o nieliniowej funkcji aktywacji) dla problemów rozpoznawania wzorców.

Metodyka automatycznej konfiguracji sieci, zaprezentowana w przedstawionej wyżej koncepcji, wychodzi z założenia, że skoro można nauczyć sieć neuronową odwzorowania pewnego niezmiennego zbioru wzorców uczących, powinna istnieć metoda pozwalająca na szybkie obliczenie parametrów sieci na ich podstawie. W przypadku biologicznego mózgu sytuacja jest inna. Mózg zmuszony jest pracować na danych, które się cały czas dynamicznie zmieniają, i musi być zdolny do dostosowywania się do zmieniającego się środowiska. Jeśli jednak dane, które chcemy sieć nauczyć mają charakter statycznego, skończonego i niesprzecznego zbioru uczącego, wtedy musi istnieć przekształcenie odwzorowujące dane wejściowe w dane wyjściowe, jak również pewna aproksymacja tego odwzorowania możliwa do zaimplementowania na bazie sieci neuronowych.

3.5. Praktyczna realizacja metody - aplikacja „Optimal Recognition”

Dla zilustrowania funkcjonowania zaproponowanej w tej pracy metody automatycznej konfiguracji sieci neuronowych dla wzorców binarnych, a także dla praktycznego sprawdzenia jej efektywności została przez autora pracy opracowana i oprogramowana aplikacja nazwana „Optimal Recognition” (Rysunek 3.2.). Aplikacja ta w praktyczny sposób umożliwia zbadanie słuszności założeń oraz sprawdzenie jakości uzyskanego uogólnienie, jakie wynikowa sieć neuronowa daje dla konkretnego problemu zadanego ciągiem uczącym. Aplikacja ta umożliwia tworzenie nowych ciągów uczących, jak również przeprowadzenie w pełni automatycznej konfiguracji architektury i parametrów sieci neuronowej (rozdział 3.2.1. i 3.2.4.) według zadanych kryteriów zgodnie z rozdziałem 3.2.3. tej pracy. Kolejnym udogodnieniem zaimplementowanym w tej aplikacji jest możliwość automatycznego

obliczania jakości uzyskanego uogólnienia jak i wyznaczenie jakości rozpoznawania (rozdział. 3.2.2.), co daje możliwość natychmiastowej weryfikacji stopnia dokonanej redukcji i podjęcia kolejnych kroków w myśl wymagań stawianych przez konkretnie rozwiązywane zadanie.



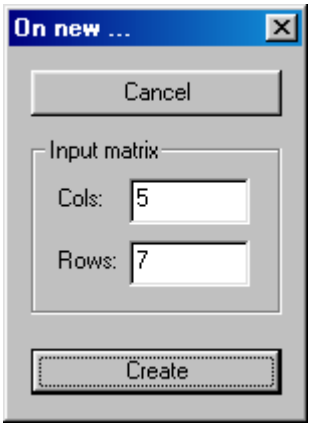
Rysunek 3.2. Główne okno aplikacji „Optimal Recognition” z uwidocznieniem jednego wzorca ciągu uczącego, skonfigurowanych wag synaptycznych dla niego oraz wyniku rozpoznawania go przez wynikową sieć neuronową z zaznaczeniem 100% zgodności z rozpoznanym wzorcem z ciągu uczącego. Żółty kolor oznacza drugi najbliższy wzorzec pod względem bliskości rozpoznania.

3.5.1. Przykład rozwiązania problemu zadanego ciągiem uczącym

Dla lepszego zilustrowania działania aplikacji „Optimal Recognition” posłużymy się prostym przykładem 26-literowego alfabetu, który zostanie wprowadzony jako ciąg uczący.

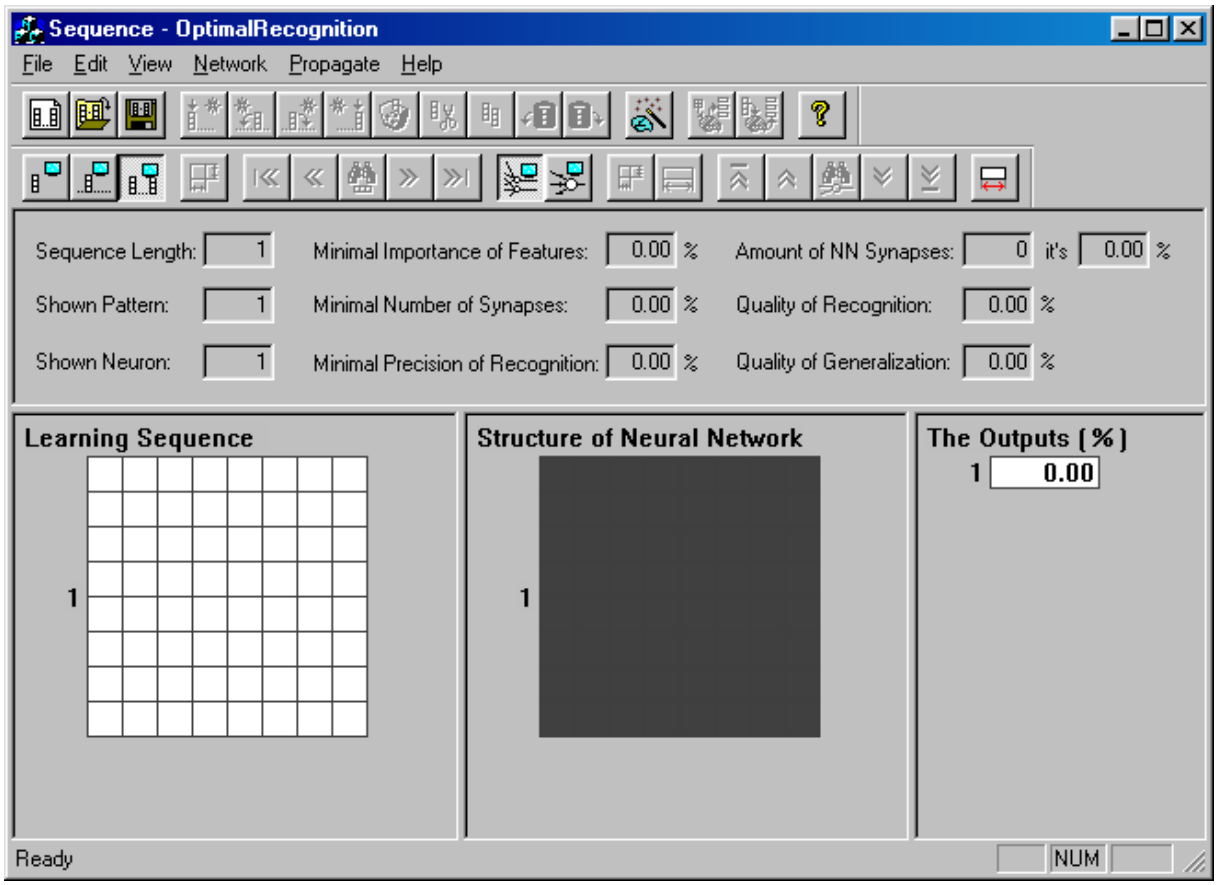
Następnie zostanie automatycznie skonfigurowana sieć neuronowa i obliczona jakość rozpoznawania i jakość uzyskanego uogólnienia. W końcu pokazane będą przykładowe wyniki rozpoznawania różnych wzorców przy użyciu skonfigurowanej sieci neuronowej.

Najpierw posługując się przyciskiem z paska narzędzi  lub korzystając z menu File/New... obieramy wymiary macierzy wzorców ciągu uczącego tak, jak to zostało pokazane na rysunku 3.3.



Rysunek 3.3. Okienko umożliwiające obranie wymiarów macierzy wzorców ciągu uczącego

Po wydaniu polecenia zbudowania ciągu uczącego w oparciu o podane wymiary w głównym oknie aplikacji pojawi się pusty jednoelementowy ciąg uczący, jak to zostało pokazane na rysunku 3.4.

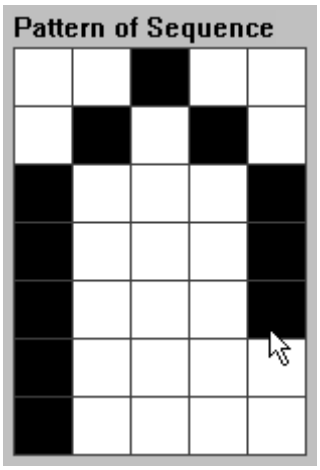


Rysunek 3.4. Widok głównego okna aplikacji po wydaniu polecenia budowy nowego ciągu uczącego

Następnie po wyborze widoku kolejnych wzorców ciągu uczącego  można przejść do przeglądania i edycji wzorców ciągu uczącego dodając nowe, usuwając, wycinając, kopiując i wklejając korzystając z odpowiednich przycisków pokazanych na rysunku 3.5.

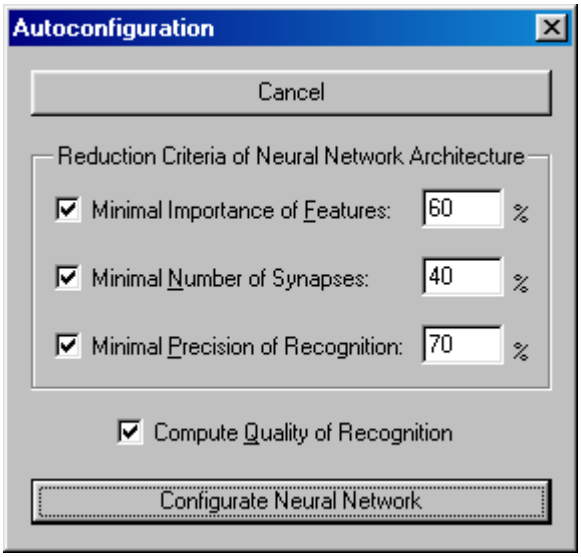
Rysunek 3.5. Przyciski umożliwiające przeglądanie i edycję ciągu uczącego.

Podczas edycji wzorców uczących posługujemy się myszką odpowiednio przyciskając na lewy przycisk myszki zaznaczając, bądź odznaczając odpowiednie pola, bądź nacisnąwszy przycisk myszki ciągnąc i malując lub usuwając dowolne linie jak to pokazano na rysunku 3.6.



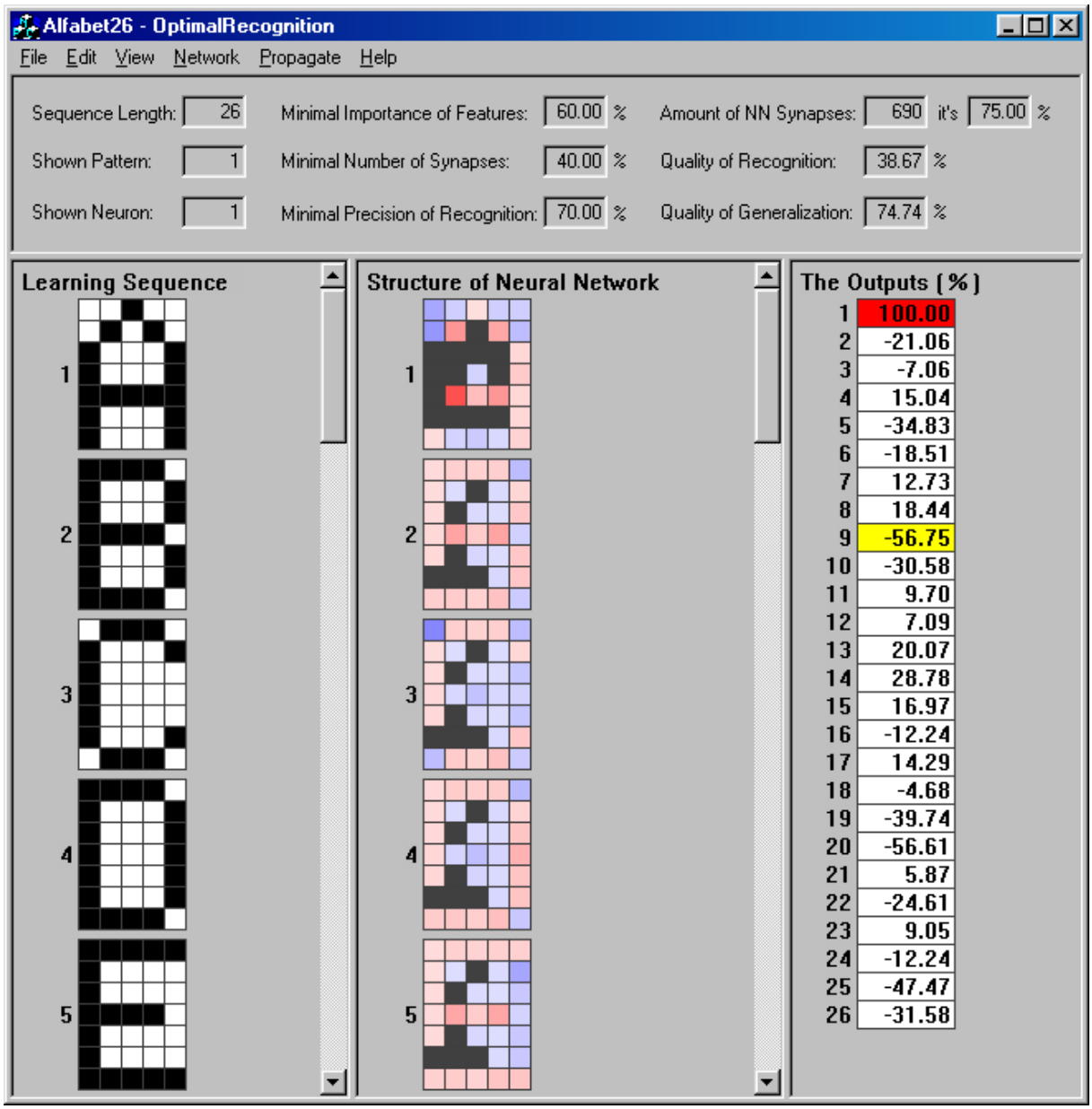
Rysunek 3.6. Do edycji poszczególnych wzorców uczących jest wykorzystywana myszka.

Gdy edycja ciągu uczącego jest zakończona, można przystąpić do etapu automatycznej konfiguracji architektury i parametrów sieci neuronowej poprzez naciśnięcie przycisku  . W rezultacie pojawi się okienko dialogowe (Rysunek 3.7.) umożliwiające wybranie i ustalenie parametrów redukcji architektury sieci neuronowej według opisanych w rozdziale 3.2.3. kryteriów.



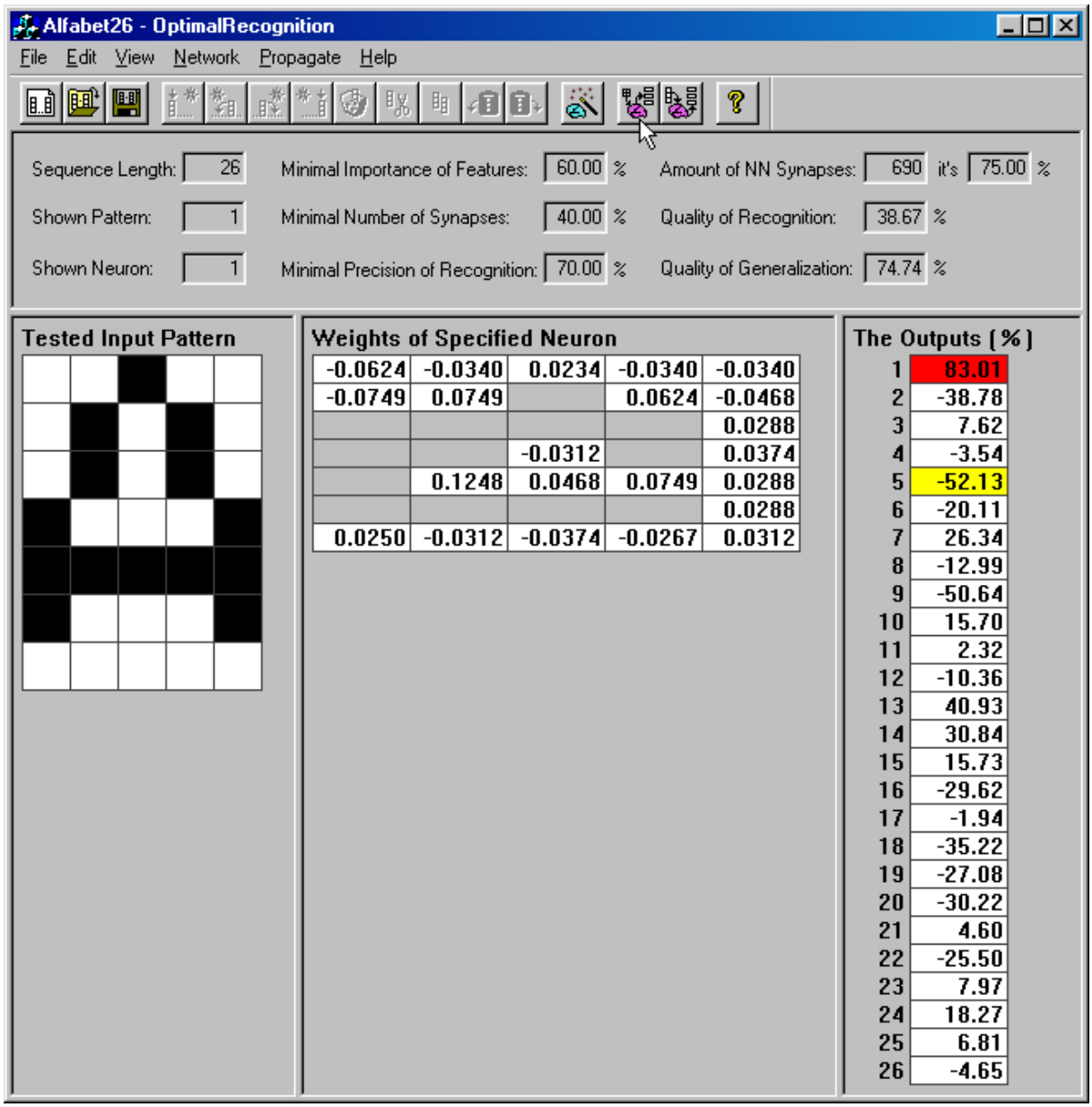
Rysunek 3.7. Okienko dialogowe umożliwiające określenie parametrów redukcji architektury sieci neuronowej.

Po wydaniu polecenia konfiguracji sieci neuronowej błyskawicznie pojawi się skonfigurowana architektura sieci wraz z obliczonymi wagami synaptycznymi w środkowej części dzielonego widoku jak to pokazuje Rysunek 3.8.


Alfabet26 - OptimalRecognition
File Edit View Network Propagate Help
Sequence Length: 26
Minimal Importance of Features: 60.00 %
Amount of NN Synapses: 690 it's 75.00 %
Shown Pattern: 1
Minimal Number of Synapses: 40.00 %
Quality of Recognition: 38.67 %
Shown Neuron: 1
Minimal Precision of Recognition: 70.00 %
Quality of Generalization: 74.74 %
Learning Sequence
1
2
3
4
5
Structure of Neural Network
1
2
3
4
5
The Outputs (%)
1 100.00
2 -21.06
3 -7.06
4 15.04
5 -34.83
6 -18.51
7 12.73
8 18.44
9 -56.75
10 -30.58
11 9.70
12 7.09
13 20.07
14 28.78
15 16.97
16 -12.24
17 14.29
18 -4.68
19 -39.74
20 -56.61
21 5.87
22 -24.61
23 9.05
24 -12.24
25 -47.47
26 -31.58

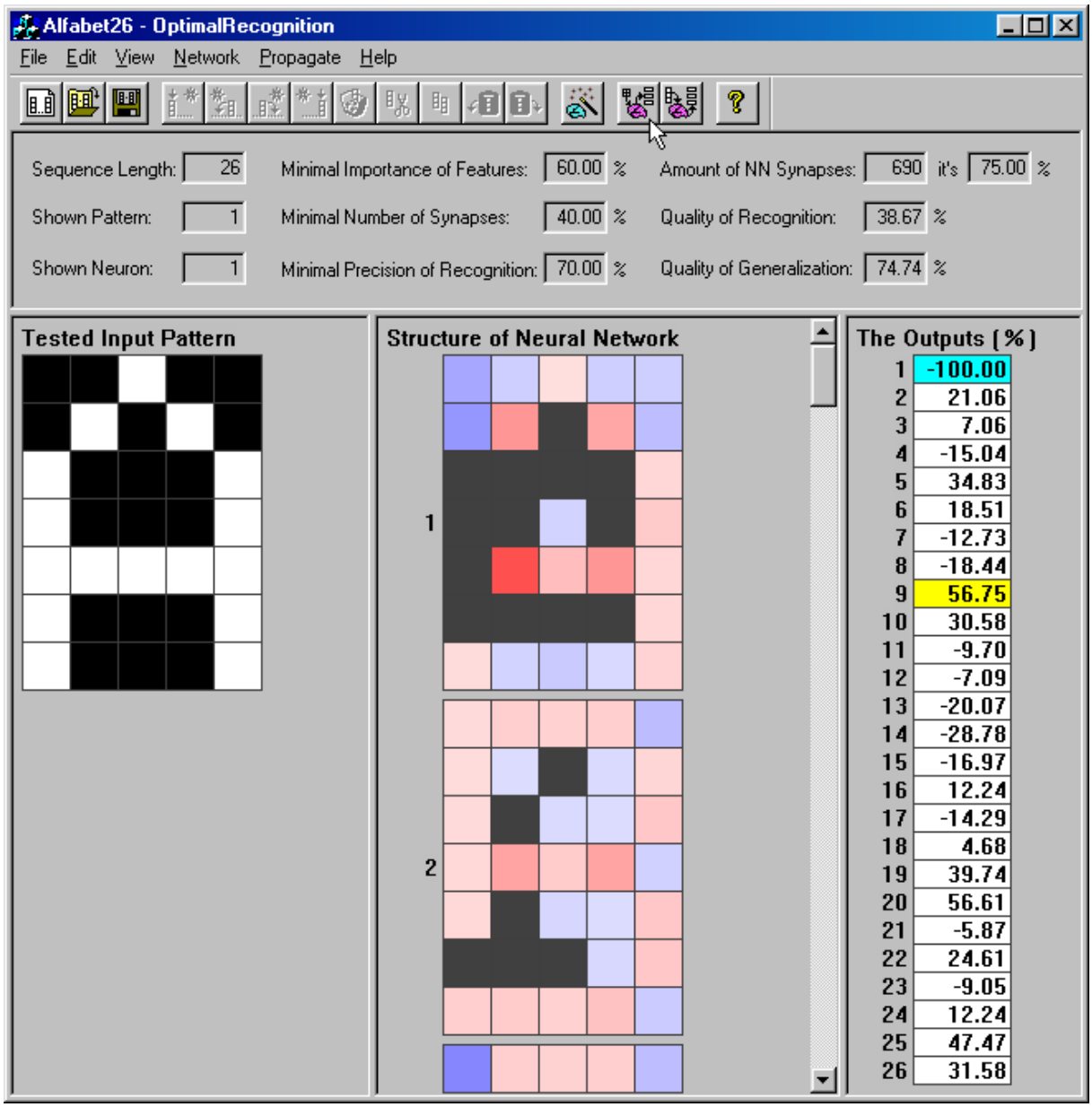
Rysunek 3.8. Widok ciągu uczącego (z lewej) i widok zredukowanej architektury sieci neuronowej według zadanych kryteriów (w środku). Szare pola odpowiadają zredukowanym synapsom, czerwone pola oznaczają synapsy o charakterze pobudzającym, a niebieskie pola synapsy o charakterze hamującym. Stopień zabarwienia kolorem czerwonym i niebieskim odzwierciedla bezwzględną wartość wagi synaptycznej odpowiednio dla synaps pobudzających i hamujących. Zarazem im bardziej intensywny jest kolor, tym bardziej unikalne jest dane pole dla danego wzorca na tle całego ciągu uczącego. Z prawej strony widać odpowiedź sieci na pobudzenie jej pierwszym wzorcem uczącym, co odpowiada w tym przypadku literce „A". Kolorem czerwonym oznaczona jest maksymalna bezwzględna wartość wyjścia, co jest równoznaczne z prawidłowym rozpoznaniem literki „A". Żółtym kolorem oznaczona jest druga największa w sensie rozpoznania bezwzględna wartość, jaka pojawiła się na wyjściu sieci. W tym przypadku sieć neuronowa została zredukowana o 25% synaps przy zachowaniu dobrych wskaźników jakości rozpoznawania (Q_R = 38,67% > 0) i jakości uzyskanego uogólnienia (Q_G = 74,74%).

Po udanej konfiguracji sieci neuronowej można przejść już tylko do sprawdzenia zdolności sieci odnośnie rozpoznawania i uogólniania wzorców uczących. Jak pokazano na rysunku 3.8. sieć prawidłowo rozpoznała 1. wzorzec uczący po naciśnięciu przycisku  z głównego paska narzędziowego. Zaburzymy teraz ten wzorzec naciskając przycisk  , służący do edycji wzorców testujących. Następnie wydamy polecenie klasyfikacji tego wzorca (nie należącego do ciągu uczącego) sieci neuronowej naciskając przycisk  , jak to pokazano na rysunku 3.9.



Rysunek 3.9. Przedstawia prawidłowe rozpoznanie zaburzonego wzorca uczącego „A”. W środku okienka widoczna jest zredukowana macierz wag dla prawidłowo rozpoznanego wzorca uczącego „A”.

Następnie można sprawdzić, jak sieć sobie poradzi z rozpoznaniem negatywu wzorca uczącego i czy założone postulaty metody wobec intuicyjnej interpretacji tak rozpoznanych wzorców się sprawdzą. W związku z tym na wejście skonfigurowanej sieci neuronowej podamy negatyw rozważanego wzorca uczącego „A” i analogicznie jak w poprzednim przypadku damy sieci polecenie klasyfikacji tego wzorca, jak to pokazano na rysunku 3.10.



Rysunek 3.10. Przedstawia prawidłowe rozpoznanie negatywu wzorca uczącego „A”. Znak „-” określa intuicyjną odwrotność rozpoznawanego wzorca względem wzorca uczącego, zaś wartość „100%” określa pełną jego zgodność z kształtem.

Jak widać przedstawiona sieć neuronowa prawidłowo sklasyfikowała przedstawione jej wzorce przy zachowaniu zdolności do uogólnień przy redukcji synaps o ¼ ich całkowitej ilości.

4. METODA STEROWANYCH KOMPROMISÓW

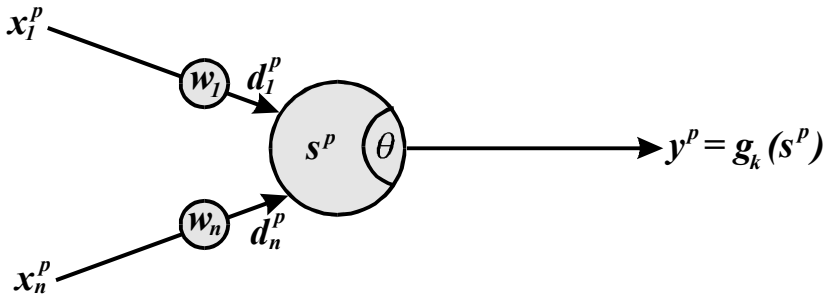
Proponowana w tej pracy metoda sterowanych kompromisów (MGC – *Method of Guided Compromises*) jest metodą należącą do grupy metod uczenia z nauczycielem (*supervised learning*). W metodzie tej nie korzysta się jednak z informacji o gradiencie, jak jest to np. w popularnej metodzie propagacji wstecznej błędu (BP – *Back Propagation*). Proces uczenia sieci neuronowej zaproponowaną w tej pracy metodą bazuje na wyznaczanych podczas uczenia tzw. odchyleniach od uzyskanego kompromisu, określanych w danym miejscu sieci podczas procesu uczącego, służących do wyznaczania kierunku zbliżania się do finalnego rozwiązania, zadanego ciągiem uczącym. Opisana metoda znajduje zastosowanie do stosunkowo szerokiego zakresu zadań uczenia sieci neuronowych o topologiach nie zawierających sprzężeń zwrotnych. Zaproponowana metoda jest wciąż rozwijana. Jej aktualna wersja pozwala na efektywne uczenie sieci dwuwarstwowych. Rozwój metody wiąże się z pewnymi jej uogólnieniami, pozwalającymi w ciekawy sposób zmierzyć się z nawet bardzo trudnymi zadaniami (np. klasyczny problem 2 spiral). W pracy tej przedstawiono nie tylko w pełni funkcjonalną, dopracowaną użytkowo część metody sterowanych kompromisów, lecz również tą jej część, która jest nadal w sferze dalszych badań - dla wyeksponowania zalet metody, ale również i problemów związanych z wprowadzanymi jej uogólnieniami. Metoda wydaje się perspektywicznie ciekawa, uznano więc, że celowe jest przybliżenie kierunków dalszego jej rozwoju, gdyż może to stanowić źródło inspiracji dla innych badaczy. Problematyka tworzenia nowych metod uczących jest na tyle obszerna i ciekawa, że postanowiono w pracy przedstawić także analizę trudności napotkanych przy tworzeniu opisywanej metody, wychodząc z założenia, że wszystkie informacje dotyczące problemów z dostosowaniem parametrów sieci do postawionego jej zadania stanowić mogą wartościowe źródło informacji dla kolejnego konstruktora konkretnej adaptacji sieci do rozwiązania innego konkretnego zadania. Na podstawie takich informacji, obiektywnie przedstawiających zalety i wady proponowanych (różnych) technik uczenia sieci neuronowych, konstruktor neuronowego rozwiązania wybranego praktycznego problemu może bardziej świadomie dokonywać wyboru stosowanej przez siebie metody i nabyć większej pewności w związku z jej stosowaniem i gwarancji poprawności uzyskanych wyników.

Schemat tego rozdziału jest następujący: Po krótkim opisie stosowanych modeli i użytej notacji następuje wprowadzenie pewnych założeń dla zaproponowanej metody. Są to założenia odnośnie szybkości nauki, sposobu korekty powstałych w procesie nauki błędów,

parametrów wpływających na sukces procesu uczenia sieci neuronowej i pewnej szczególnej cechy, na której bazuje w swojej „filozofii" opisana metoda sterowanych kompromisów. Po tym krótkim wprowadzeniu następuje gruntowny opis metody, który rozpoczyna się od opisu jej najprostszej postaci, ze względu na ułatwienie systematycznego jej poznania. W dalszym rozdziale metoda podlega licznym rozszerzeniom, wprowadzanym systematycznie przez autora w celu poprawienia i rozszerzenia jej możliwości obliczeniowych i zakresu stosowalności. Następnie rozważane są zalety i wady przedstawionej metody i wyznaczone są kierunki poszukiwań dla dalszego jej rozwoju. Końcowe podsumowanie zawiera również informacje o problemach związanych z kolejnymi uogólnieniami opisanej metody, które mogą posłużyć również innym badaczom jako źródło inspiracji i poszukiwań. W końcu opisana jest aplikacja „*Brain for Problem*", umożliwiająca praktyczne sprawdzenie słuszności założeń prezentowanej metody.

4.1. Zastosowany model neuronu i używane słownictwo i notacja

Ze względu na różnorodność notacji stosowanej przez wielu autorów oraz ze względu na wykorzystywanie różnych modeli neuronu w różnych pracach - dla uściślenia dalszego opisu przyjęto następujący model neuronu i notację przedstawione na rysunku 4.1.



Rysunek 4.1. Model neuronu wraz z użytymi oznaczeniami

Na rysunku tym (i dalej w tekście) używać będziemy następujących oznaczeń:

$x_1^p, ..., x_n^p$ – sygnały wejściowe (*input signals*) rozważanego neuronu lub efektora, wyznaczone dla ustalonego wzorca p,

$w_1, ..., w_n$ – wagi synaptyczne rozważanego neuronu lub efektora (*weights*),

$d_1^p, ..., d_n^p$ – postsynaptyczne pobudzenie dendrytyczne rozważanego neuronu lub efektora, wyznaczone dla ustalonego wzorca p,

θ – próg rozważanego neuronu lub efektora (*threshold*),

s^p – stan pobudzenia neuronu rozważanego neuronu lub efektora, wyznaczony dla ustalonego wzorca *p*

y^p – sygnał wyjściowy (*output signal*) rozważanego neuronu lub efektora, wyznaczony dla ustalonego wzorca *p*

Sygnał wyjściowy neuronu y^p wyznaczany jest jako funkcja stanu pobudzenia neuronu:

$$y^p = g_k\left(s^p\right)$$

Stan pobudzenia neuronu s^p obliczany jest jako suma pobudzeń poszczególnych dendrytów; następnie od tej sumy odejmowana jest wartość progu:

$$s^p = \sum_{i=1}^{n} d_i^p - \theta \qquad (4.1.1)$$

Pobudzenie dendrytyczne d_i^p jest ważoną wartością sygnału wejściowego:

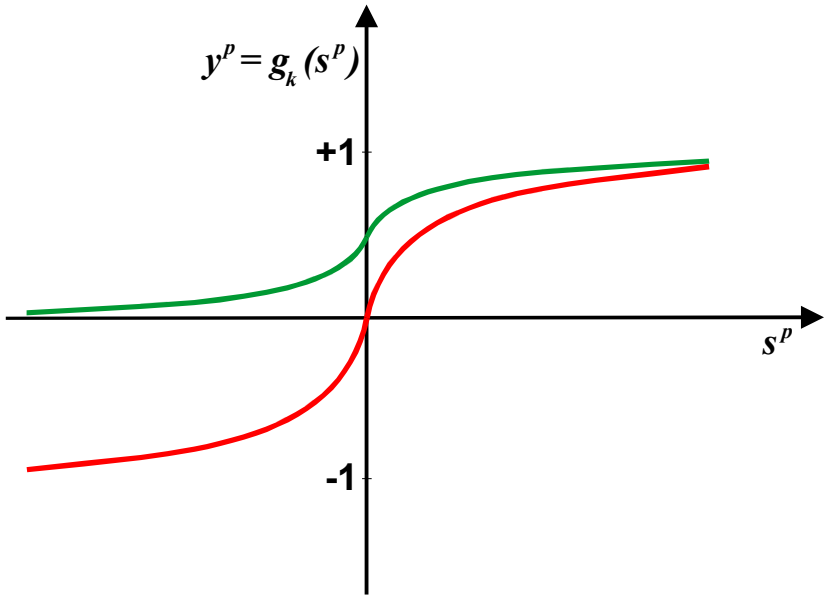
$$d_i^p = x_i^p \cdot w_i$$

Bardzo często jako funkcję aktywacji neuronu wybiera się funkcję o charakterystyce sigmoidalnej (*generic sigmoid activation function* (FIESLER B3.2:4)) (Rys. 4.2.), zdefiniowaną najczęściej w następujący sposób:

$$y^p = g_{sigm}\left(s^p\right) = \frac{1}{1+\exp\left(s^p\right)} \qquad \text{taka, że} \qquad g : R \to [0, 1]$$

lub

$$y^p = g_{\tanh}\left(s^p\right) = \tanh\left(s^p\right) \qquad \text{taka, że} \qquad g : R \to [-1, 1]$$



Rysunek 4.2. Często stosowane funkcje sigmoidalne jako funkcje aktywacji neuronu

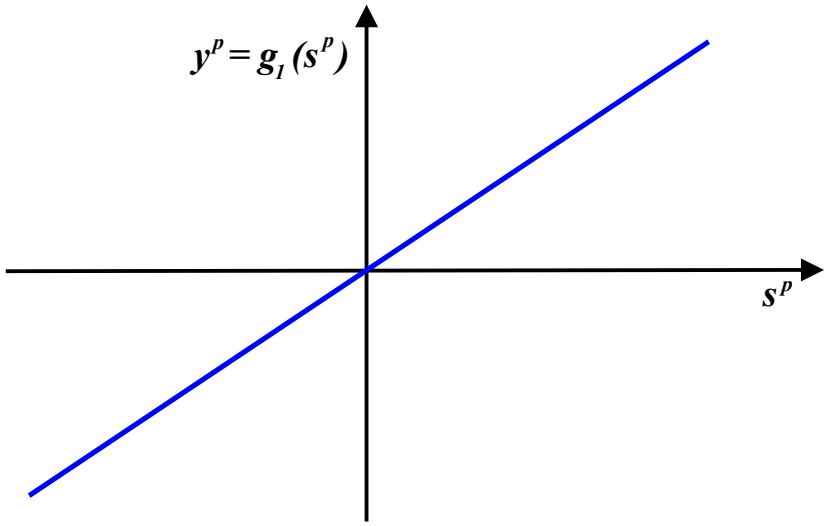
Jakkolwiek funkcje te w pewien sposób przybliżają funkcję progową (*threshold function*), jaka jest obserwowana u biologicznych odpowiedników sztucznych neuronów, jednak z matematycznego punktu widzenia sieć neuronowa złożona z neuronów o tak zdefiniowanych funkcjach aktywacji nie będzie dawała dobrego uogólnienia dla szeroko pojętej ekstrapolacji, lecz może służyć głównie do interpolacji. Ponadto użycie funkcji o ograniczonym zbiorze wartości funkcji nastręcza pewne trudności przy konwersji rzeczywistych wartości doświadczalnych na wartości ciągu uczącego i odwrotnie.

W tej pracy zdefiniowano odmienny model neuronu, przyjęty w proponowanej postaci właśnie ze względu na poprawienie właściwości ekstrapolacyjnych tworzonej sieci neuronowej. Model ten operuje na pełnym zbiorze liczb rzeczywistych jako na zbiorze dozwolony wartości funkcji, więc jego użycie nie wymusza sztucznego skalowania danych, jak się to dzieje w klasycznym modelu wykorzystującym funkcje sigmoidalne. Podstawowe funkcje aktywacji dla tego modelu zostały zdefiniowane następująco:

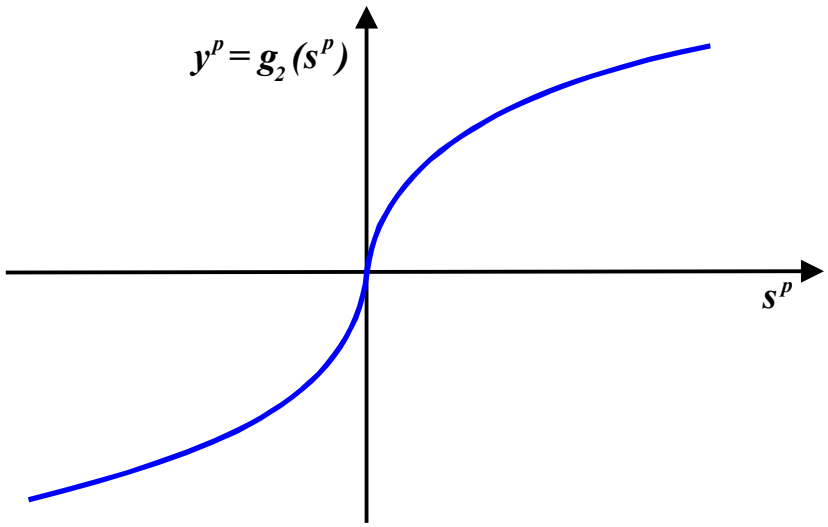
$y^p = g_1(s^p) = \beta \cdot s^p$ - funkcja liniowa (Rys. 4.3.)

$y^p = g_2(s^p) = \mathrm{sgn}(\beta \cdot s^p) \cdot \sqrt{|\beta \cdot s^p|}$ - funkcja pierwiastkowa (Rys. 4.4.)

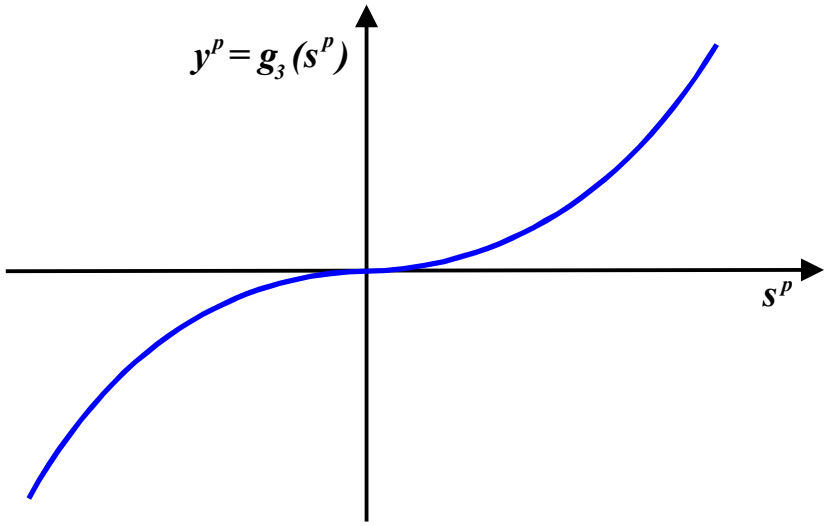
$y^p = g_3(s^p) = \beta \cdot s^p \cdot |\beta \cdot s^p|$ - funkcja potęgowa (Rys. 4.5.)



Rysunek 4.3. Funkcja aktywacji - liniowa: $g_1(s^p) = \beta \cdot s^p$ taka, że $g_1 : R \to R$



Rysunek 4.4. Funkcja aktywacji - pierwiastkowa: $g_2(s^p) = \mathrm{sgn}(\beta \cdot s^p) \cdot \sqrt{|\beta \cdot s^p|}$ taka, że $g_2 : R \to R$



Rysunek 4.5. Funkcja aktywacji - potęgowa: $g_3\left(s^p\right) = \beta \cdot s^p \cdot \left|\beta \cdot s^p\right|$ taka, że $g_3 : R \to R$

W dalszych rozważaniach zastosujemy nietypową notację, zwiększającą jednak czytelność opisów formalnych, mianowicie założymy, że wartości zmiennych związanych z sygnałem uczącym, mającym swe źródło w ciągu uczącym, będą oznaczane dużymi literami:

Y^p – wyjściowy sygnał uczący pochodzący od nauczyciela rozważanego neuronu lub efektora, wyznaczony dla ustalonego wzorca uczącego p,

S^p – uczony stan pobudzenia neuronowego rozważanego neuronu lub efektora, wyznaczony dla ustalonego wzorca uczącego p,

D_i^p – uczone i-te postsynaptyczne pobudzenie dendrytyczne rozważanego neuronu lub efektora, wyznaczone dla ustalonego wzorca uczącego p,

W_i^p – nowa wartość i-tej wagi wyznaczona w wyniku korekty rozważanego neuronu lub efektora dla ustalonego wzorca uczącego p,

Θ^p – nowa wartość progu wyznaczona w wyniku korekty rozważanego neuronu lub efektora dla ustalonego wzorca uczącego p,

X_i^p – nowy i-ty wejściowy sygnał uczący obliczony w wyniku wstecznej propagacji sygnału uczącego pochodzącego od nauczyciela na podstawie ciągu uczącego rozważanego neuronu lub efektora, wyznaczony dla ustalonego wzorca uczącego p (w przypadku sieci neuronowych zawierających neurony pośrednie i/lub warstwy ukryte).

Korekty błędów będą oznaczane grecką literką σ z indeksem dolnym odnoszącym się do parametru, który jest korygowany. Poniższe oznaczenia przyjmują więc następujące znaczenie:

σ_y^p – korekta błędu sygnału wyjściowego y^p neuronu, wyznaczona dla ustalonego wzorca uczącego p; $\sigma_y^p = Y^p - y^p$,

σ_s^p – korekta błędu stanu pobudzenia neuronu s^p, wyznaczona dla ustalonego wzorca uczącego p; $\sigma_s^p = S^p - s^p$,

σ_θ^p – korekta progu θ, wyznaczona dla ustalonego wzorca uczącego p; $\sigma_\theta^p = \Xi^p - \theta^p$,

$\sigma_{d_i}^p$ – korekta błędu i-tego postsynaptycznego pobudzenia dendrytycznego d_i^p, wyznaczona dla ustalonego wzorca uczącego p; $\sigma_{d_i}^p = D_i^p - d_i^p$,

$\sigma_{w_i}^p$ – korekta i-tej wagi w_i, wyznaczona dla ustalonego wzorca uczącego p; $\sigma_{w_i}^p = W_i^p - w_i^p$,

$\sigma_{x_i}^p$ – korekta błędu i-tego sygnału wejściowego x_i^p, wyznaczona dla ustalonego wzorca uczącego p (używane w przypadku sieci neuronowych zawierających neurony pośrednie i/lub warstwy ukryte); $\sigma_x^p = X_i^p - x_i^p$.

W odniesieniu do ciągu uczącego zastosowano następującą notację:

p – numer porządkowy wzorca uczącego należącego do ciągu uczącego,

Q – ilość wzorców uczących, czyli długość ciągu uczącego,

Dla zwiększenia czytelności następujących opisów wprowadzimy kilka pojęć:

Proces uczenia – składa się zazwyczaj z wielu epok uczących i ma na celu taką stopniową modyfikację stanu sieci neuronowej (tzn. jej parametrów wolnych), aby odwzorowanie

(dokonane przy jej pomocy) wzorców uczących w odpowiednie wyjścia sieci było jak najbardziej zgodne z ciągiem uczącym.

Epoka ucząca (nazywana dalej krótko epoką) – składa się z jednego lub więcej etapów uczących. Jej celem jest korekta wszystkich parametrów wolnych sieci dla wszystkich wzorców uczących, przy czym każdy parametr sieci jest modyfikowany w danej epoce uczącej dokładnie raz dla każdego wzorca uczącego.

Etap uczący (nazywany dalej krótko etapem) – składa się z fazy pobudzenia sieciowego i z fazy wstecznej propagacji sygnału uczącego dla każdego wzorca uciągu uczącego. Każdy etap kończy się po wykonaniu tych dwóch faz kolejno dla wszystkich wzorców uczących fazą kompromisowej aktualizacji wag i progów. Etap uczący dotyczy zawsze tylko jednej warstwy sieci. O ile sieć posiada więcej warstw, wtedy na epokę uczącą składa się kilka etapów uczących.

Wirtualna warstwa uczenia (nazywana dalej krótko warstwą) – jest to taki zbiór sensorów, neuronów i efektorów sieci neuronowej, który jest aktualizowany w jednym etapie uczącym. Warstwa taka nie koniecznie musi być zgodna z fizycznymi warstwami zaprojektowanymi przez konstruktora sieci, lecz jest bardziej zależna od istniejących w sieci połączeń pomiędzy neuronami i efektorami. Ponadto w przypadku metod dynamicznie rekonfigurujących architekturę sieci neuronowej (np. metody ontogeniczne) również warstwy uczenia mogą ulegać zmianom, tj. zbiór sensorów, neuronów i efektorów składający się na daną warstwę uczenia może ulegać modyfikacjom. Można powiedzieć, że warstwy uczenia są wirtualnie tworzone w trakcie procesu uczenia, a ich żywotność jest ściśle związana z danym etapem uczącym. Prawdziwa wirtualność tych warstw ujawnia się dopiero w przypadku metod dynamicznie zmieniających architekturę sieci w trakcie procesu uczenia, albowiem w przypadku metod o sztywnej architekturze warstwy te są każdorazowo (tj. w każdej epoce uczącej dla każdego etapu uczącego) wyznaczane w taki sam sposób.

Po wprowadzeniu tych oznaczeń w kolejnym podrozdziale przedstawimy istotę proponowanej metody.

4.2. Założenia metody

Poszukiwanie nowych metod uczenia sieci neuronowych jest nierozłącznie związane z uświadomieniem sobie mankamentów i słabych stron metod już stosowanych. Odnalezienie „wąskich gardeł” metod istniejących może stać się wspaniałą inspiracją do badań

prowadzących w efekcie do eliminacji niekorzystnych lub nieefektywnych algorytmów obecnie stosowanych. Z tego też względu przed przystąpieniem do gruntownego opisu proponowanej tu metody sterowanych kompromisów, przyjrzyjmy się i porównajmy popularne metody z postulowanymi założeniami zaproponowanej metody.

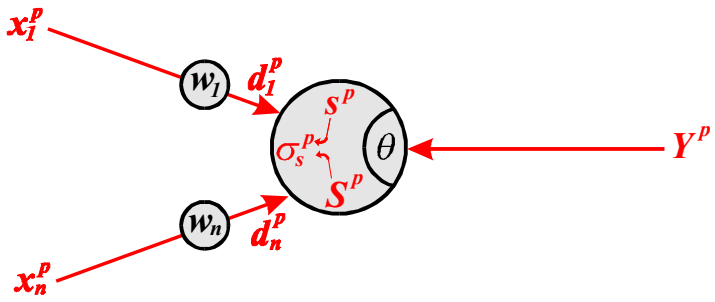
4.2.1. Szybkość nauki

Głównym celem przyświecającym autorowi pracy przy projektowaniu metody sterowanych kompromisów była próba przyspieszenia procesu uczenia sieci neuronowej poprzez korygowanie powstałego na jej wyjściu błędu w całości dla rozważanego wzorca uczącego. Oznacza to, że w opisanej metodzie nie występuje żaden współczynnik uczenia η (*learning rate*) wpływający na szybkość nauki. Współczynnik ten stosowany jest we wielu popularnych metodach, jednak w związku z jego stosowaniem istnieje również wiele zasadniczych problemów. Trudność w jego stosowaniu polega na tym, iż w żadnej epoce uczącej nie znana jest jego optymalna wartość. Wiadomo tylko, że ani za duża, ani za mała jego wartość nie wpływa korzystnie na proces uczenia. Za duża wartość powoduje zazwyczaj fluktuacje w procesie uczenia a nawet problemy z zachowaniem jego zbieżności, zaś za mała zwalnia ten proces. Stosowanie współczynnika uczenia η jest ściśle związane ze stosującymi go metodami, które wymagają jego istnienia. Metoda sterowanych kompromisów nie wymaga istnienia jakiegokolwiek arbitralnie czy algorytmicznie wybieranego współczynnika uczenia, a wręcz jego istnienie byłoby dla procesu uczenia niekorzystne. Dzieje się tak dlatego, iż zaproponowana w tej pracy metoda posługuje się innymi przesłankami w procesie dążenia do rozwiązania określonego ciągiem uczącym niż popularne metody stosowane obecnie.

4.2.2. Sposób korekty błędu

W popularnej metodzie wstecznej propagacji błędu, jak sama nazwa mówi, obliczany jest błąd, jaki powstaje na wyjściu sieci neuronowej, i następnie na jego podstawie korygowane są wagi i progi w procesie jego wstecznej propagacji przez sieć. W ten sposób działają wszystkie algorytmy uczenia zbudowane w oparciu o metody gradientowe. Metoda sterowanych kompromisów nie propaguje wstecz **błędu** powstałego na wyjściu, lecz pożądany sygnał uczący Y^p, zadany ciągiem uczącym. Propagacja ta odbywa się na bazie informacji płynących ze wcześniejszego etapu propagacji sygnału pobudzenia dla ustalonego wzorca uczącego p przez sieć neuronową. Obliczany jest rozkład wstecznie propagowanego sygnału uczącego Y^p na poszczególne dendryty, które następnie obliczają swój udział w powstałym

błędzie na podstawie sygnałów płynących od ich wejścia ($x_1^p,..., x_n^p$). Zatem w prezentowanej metodzie także ma miejsce obliczanie błędu, jednak odbywa się ono lokalnie w każdym neuronie sieci neuronowej na podstawie lokalnych informacji, jakie neuron posiada ze swojego najbliższego otoczenia dla ustalonego wzorca uczącego *p*, jak to pokazano na rysunku 4.6..



Rysunek 4.6. Lokalna korekta błędu σ_s^p następuje po otrzymaniu przez neuron wartości pobudzenia $x_1^p,..., x_n^p$ płynących od dendrytów oraz wartości sygnału uczącego Y^p płynącego od aksonu.

Właściwa korekta parametrów wolnych sieci (tutaj wag i progów) następuje dopiero po obliczeniu błędów lokalnych dla wszystkich wzorców biorących udział w procesie uczenia sieci. Na podstawie obliczonych błędów poszukiwany jest **kompromis** będący zarazem nowym stanem sieci neuronowej. Kompromis ten powinien zapewnić końcową zbieżność nauki, jej stabilność i możliwie maksymalną szybkość. Podczas obliczania kompromisu sprawą priorytetową nie jest sama tylko stabilnie malejąca charakterystyka funkcji błędu sieci, ponieważ ona świadczy tylko o zbieżności do pewnego minimum – być może minimum lokalnego, które nie musi być zarazem poszukiwanym minimum globalnym. Poszukiwanie kompromisu powinno więc być sterowane w taki sposób, aby zapewnić sieci możliwość osiągnięcia poszukiwanego minimum globalnego funkcji błędu sieci dla całego ciągu uczącego.

4.2.3. Sukces procesu uczenia

Dla dowolnego wzorca uczącego i dowolnej nietrywialnej sieci neuronowej teoretycznie istnieje nieskończona ilość rozwiązań (tj. ustawień wag i progów), dla których sieć prawidłowo odwzorowuje zadanie, wyrażone poprzez wzorzec uczący w jego pożądane rozwiązanie, które musi się pojawić na wyjściu sieci. Dzięki tej właściwości sieć neuronowa

po procesie nauki nie rozwiązuje tylko jednego, określonego zadania, ale może – na zasadzie kompromisu – być dostosowywana do rozwiązywania pewnej klasy zadań, a proces uczenia wybiera parametry sieci w celu jak najlepszego odwzorowania wielu wzorców wejściowych w ich pożądane wyjścia. Dokładność tego odwzorowania zależna jest od wielu czynników, do których zaliczyć można między innymi:

- Topologię sieci neuronowej,
- Modele neuronów użytych do budowy sieci,
- Zróżnicowanie i ilość wzorców uczących,
- Reprezentację danych w sieci neuronowej,
- Metodę uczenia sieci.
- Rozważymy teraz kolejno wszystkie wymienione wyżej uwarunkowania.

Topologia sieci. Neurony można łączyć ze sobą w różny sposób. Popularną topologią sieci neuronowej jest topologia wielowarstwowa pełniąca funkcję perceptronu, tzw. MLP (*multilayer perceptron*), w której ilość neuronów i warstw zwykle dobierana jest arbitralnie na podstawie doświadczenia konstruktora, zaś neurony w połączonych między sobą sąsiadujących warstwach łączone są ze sobą na zasadzie „każdy z każdym" (*fully interlayer-connected topology* [FIESLER 1997, B2.2, B2.5]). W taki sposób buduje się większość używanych obecnie sieci, chociaż wiele eksperymentów udowadnia, iż lepsze rezultaty nauki uzyskuje się dla topologii sieci częściowo połączonych, które są zazwyczaj bardziej oszczędne obliczeniowo i dają dodatkowo możliwość lepszego dostosowania architektury sieci do danego ciągu uczącego. Zagadnienia związane z automatyczną budową takich „oszczędnych" topologii (np. metody ontogeniczne - *ontogenic methods*) są jak na razie w fazie badań i brak sformalizowanych i udowodnionych przesłanek ich konstruowania [FIESLER 1997, B2.6].

Modele neuronów. W naukach biomedycznych rozróżnia się kilka podstawowych modeli neuronów w zależności od ich kształtu, wielkości, rodzaju i ilości ich połączeń z sąsiadującymi neuronami, rodzaju neuroprzekaźników (*neurotransmitters*), funkcji, jakie spełniają oraz ich położenia w hierarchii całej sieci neuronowej [KANDEL 1996]. Również w teorii sztucznych sieci neuronowych w przeciągu ostatnich kilkudziesięciu lat skonstruowano wiele różnych modeli neuronów (*artificial neuron*), stosowanych do budowy różnego rodzaju sieci biorąc jako priorytet ich biologiczną zgodność, bądź koncentrując się na ich

właściwościach obliczeniowych, często tylko luźno związanych z ich biologicznym odpowiednikiem.

Wzorce uczące. Zróżnicowanie i ilość wzorców uczących również ma niebagatelne znaczenie w procesie dostosowywania sieci neuronowej do zadanego ciągu uczącego. Można sobie łatwo wyobrazić, że jeśli sieć neuronowa zostanie postawiona przed problemem nauczenia się ciągu liczb losowych lub wzorców ze sobą sprzecznych, w naturalny sposób sobie z takim zadaniem nie poradzi, albowiem jej zadaniem jest odnajdywanie pewnych zależności i powiązań pomiędzy wzorcami uczącymi. Ilość wzorców oraz ich dobór jakościowy jest również nie mniej ważny. Jeśli bowiem zadamy sieci ciąg uczący, w którym zdecydowana większość wzorców reprezentować będzie pewną część modelowanego zagadnienia a tylko skromna część wzorców całą resztę, wtedy można się spodziewać, że sieć będzie podążała za głosem większości. Ciąg uczący powinien być reprezentatywny dla danego zagadnienia. Można sobie, co prawna, wyobrazić metodę uczącą, która na bazie jakiejś metryki traktuje bardziej odmienne wzorce uczące z wyższym priorytetem, jednak w takim przypadku zdolność do uogólniania tak nauczonej sieci ulegnie jakościowej zmianie, jak również inna będzie jej interpretacja i działanie. Ponadto w przypadku nie równorzędnego traktowania wzorców proces nauki byłby narażony na duży wpływ artefaktów pojawiających się czasami w danych doświadczalnych, które stają się często bez wstępnej filtracji zbiorem danych uczących.

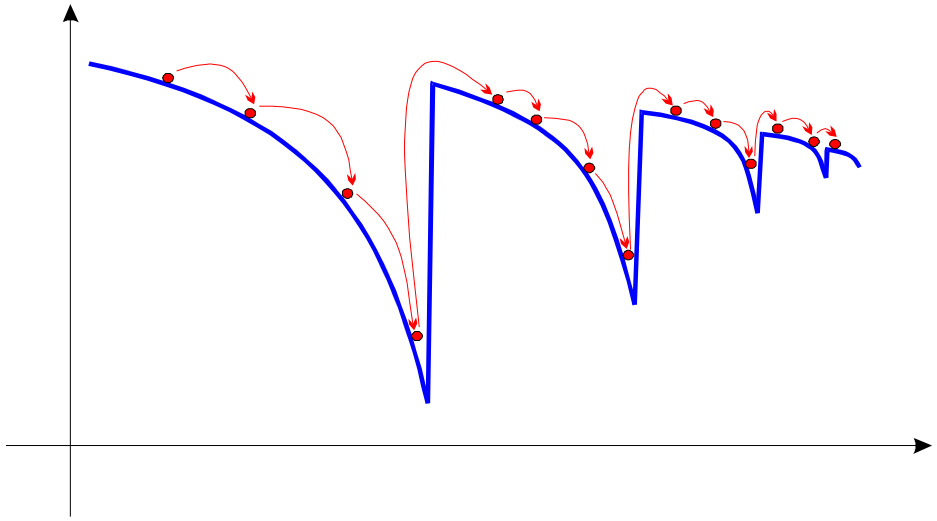
Reprezentacja danych. Wzorce uczące (a także dane używane potem podczas eksploatacji sieci) z powodów obliczeniowych muszą być reprezentowane liczbowo[1]. Jednak taka reprezentacja nie powinna być dokonywana arbitralnie, tylko tak, by intuicyjnie odzwierciedlała pewne podobieństwo rozważanych sygnałów. Jeśli na przykład litery alfabetu łacińskiego ponumerujemy zgodnie z ich porządkiem alfabetycznym (tzn. A – 1, B – 2, C – 3, itd.) i przyjmiemy taką ich reprezentację dla potrzeb ciągu uczącego, wtedy można się spodziewać kłopotów z dostosowaniem się sieci neuronowej do takiego ciągu uczącego, albowiem taka reprezentacja tych wzorców w żaden sposób nie odzwierciedla w sposób ilościowy informacji niesionej ani przez kształt tych liter ani przez ich fonetyczne brzmienie, nie wyraża też żadnej miary podobieństwa pomiędzy nimi. Warto podkreślić, iż sieć neuronowa traktuje wszystkie liczby w sposób ilościowy a nie w sensie porządku. Ze względu na ilościowe podejście sieci neuronowej do każdej informacji i biorąc pod uwagę ich

[1] Znane są oczywiście metody reprezentacji w sieciach neuronowych także danych typu jakościowego (symbolicznych, opisowych), jednak nie będziemy się nimi zajmowali w tej pracy.

numeryczną reprezentację, w procesie uczenia może dojść do sytuacji uprzywilejowania pewnych wzorców uczących (np. reprezentowanych większymi liczbami) na niekorzyść innych. Wskazane jest zatem, ażeby reprezentacja liczbowa wzorców uczących była związana w sposób ilościowy z informacjami, jakie poszczególne wzorce uczące niosą. [FIESLER B4.6]

Metoda uczenia. Sukces nauki nie jest zależny tylko od zastosowanej metody, ale równie ważny jest grunt, na jakim działa, i dane, które podlegają nauce. Dla danego ciągu uczącego i topologii sieci neuronowej (o ile jest dobierana arbitralnie) trzeba znaleźć taki zbiór wag i progów, żeby w jak najlepszy sposób (tj. obarczony jak najmniejszym błędem według zadanych kryteriów) odwzorowywał wejście sieci neuronowej w pożądane jej wyjście zgodnie z ciągiem uczącym. Za znajdywanie takiego stanu sieci odpowiedzialna jest metoda ucząca. Sieć neuronowa bez sprzężeń zwrotnych (np. MLP) jest swego rodzaju rozbudowanym przekształtnikiem funkcyjnym, na bazie którego można zdefiniować funkcję błędu sieci i dążyć do odnalezienia jej minimum globalnego. Funkcja błędu, w zależności od stopnia skomplikowania sieci może posiadać wiele minimów lokalnych. Z tego też powodu znalezienie minimum globalnego w oparciu o metody gradientowe może nie być proste, albowiem informacja płynąca z gradientu funkcji w danym punkcie wskazuje tylko na kierunek do pewnego minimum – zwykle lokalnego, które w praktyce rzadko okazuje się być zarazem minimum globalnym. Wielu autorów podejmuje próby omijania owych minimów lokalnych z pomocą różnych metod (do tego celu służy na przykład szeroko znana metoda momentum), gdyż w przypadku metod gradientowych są one punktem stagnacji procesu uczenia. Niestety nawet możliwość omijania minimów lokalnych nie zapewnia końcowego sukcesu procesu nauki, ponieważ nadal nie znany jest kierunek ani odległość poszukiwania minimum globalnego od aktualnego stanu sieci. Ponadto metoda momentum jest dla pewnych przypadków źle uwarunkowana i może prowadzić wręcz do wzrostu funkcji błędu sieci, jak to pokazano na rysunku 4.7. Problemy te związane są z tym, że informacja o gradiencie nie zawiera w sobie informacji o kierunku poszukiwania minimum globalnego, a jego znalezienie jest w takim przypadku czysto przypadkowe, zależne bardziej od stanu początkowego sieci niż od samej metody uczącej. Stanów początkowych sieci zaś jest nieskończona ilość, więc przebadanie ich wszystkich jest niemożliwe. Taka sytuacja wynika z faktu, iż w przypadku metod gradientowych korzysta się jedynie z lokalnej informacji o gradiencie funkcji błędu sieci neuronowej dla aktualnego jest stanu. Z tego też powodu metody gradientowe są z założenia nieefektywne. Z matematycznego punktu widzenia nie istnieją zaś żadne przesłanki,

wskazujące jak można by było przechodzić z jednego minimum lokalnego do innego ani gdzie lub jak daleko od aktualnego stanu sieci neuronowej inne minima się znajdują i czy są one lokalne czy globalne.



Rysunek 4.7. Przykład złego uwarunkowania metody momentum prowadzącego proces uczenia w niekorzystnym kierunku ze względu na zastosowaną bezwładność, która w pewnych sytuacjach może być korzystna a w pewnych niekorzystna.

4.2.4. Odchylenia od uzyskanego kompromisu jako globalny wyznacznik stanu procesu uczenia

Postawmy sobie teraz pytanie: Czy istnieją jakieś informacje, na podstawie których można by było wnioskować o stanie nauki sieci i efektywniej zmierzać do minimum globalnego funkcji błędu, tj. do optymalnego rozwiązania postawionego problemu zadanego ciągiem uczącym dla zadanej topologii sieci neuronowej? Otóż informacje takie istnieją! W metodach gradientowych nie uwzględnia się zazwyczaj tego, co się fizycznie w sieci neuronowej dzieje dla poszczególnych wzorców uczących w trakcie procesu uczenia. Dla każdego wzorca uczącego i jego sygnału uczącego obliczane są błędy i na ich podstawie przeprowadzana jest później korekta wag i progów. Korekta ta przebiega w różnych kierunkach i z różnym natężeniem. Finalna korekta jest zawsze wynikiem pewnego kompromisu zawartego pomiędzy wzorcami ciągu uczącego. **Kompromis** ten oparty jest zazwyczaj na pewnej średniej (np. arytmetycznej, ważonej, geometrycznej), choć można się pokusić również o inne sposoby jego wyznaczania. Niezależnie od sposobu wyznaczania nadmienionego kompromisu, kompromis ten zawsze w różny sposób satysfakcjonuje różne wzorce uczące,

które mogą z łatwością obliczyć odchylenie swojej postulowanej korekty błędu od obliczonego kompromisu (dalej nazywane krótko „**odchyleniem**"). Na tej podstawie każdy wzorzec uczący może na bieżąco analizować swoją siłę przebicia na tle całego ciągu uczącego. Co więcej każdy wzorzec może się przypatrywać kolejno wszystkim swoim odchyleniom, jakie miały miejsce w poszczególnych neuronach i synapsach, i na tej podstawie podejmować decyzję o bardziej efektywnym rozkładzie swojej korekty błędu na poszczególne elementy sieci neuronowej. W praktyce oznacza to udrożnienie jednych i blokowanie innych dróg korekty błędu. Takie postępowanie można nazwać **sterowaniem** sposobem korekty błędów w celu uzyskania jak najkorzystniejszego kompromisu z punktu widzenia całego ciągu uczącego. Stąd wypływa prosty wniosek, iż na podstawie wyżej opisanych odchyleń można wnioskować o stanie nauki sieci, albowiem wielkość tych odchyleń jest bezpośrednio związana z błędem sieci neuronowej i *vice versa*. Stan braku takowych odchyleń świadczy jednoznacznie o tym, że sieć neuronowa nauczyła się postawionego jej problemu, zadanego ciągiem uczącym. Rzadko się jednak zdarza, żeby wszystkie odchylenia od uzyskanego kompromisu miały wartość zerową. Wartościową informację stanowi więc tutaj bardziej malejąca wielkość tych odchyleń w procesie uczącym, niż same ich wartości. Ponadto, jeśli stan sieci neuronowej charakteryzuje się niezerowymi odchyleniami i zarazem bliskimi zeru kompromisowymi korektami wszystkich wag i progów, świadczy to o zbliżaniu się do minimum lokalnego. Zatem rozważana metoda pozwala minimalizować błędy występujące w sieci, zarazem dając możliwość wnioskowania o tym, czy to minimum ma szansę być globalnym czy też nie. Na podstawie odchyleń od uzyskanego kompromisu można skonstruować pewną miarę odległości od rozwiązania idealnego (tj. od takiego stanu, gdy wszystkie odchylenia są zerowe), a także można wnioskować na tej podstawie o tym, czy proces uczenia osiągnął już satysfakcjonujący poziom dokładności dla wszystkich wzorców ciągu uczącego i czy ewentualnie osiągnięte minimum kwalifikuje się jako minimum lokalne czy też globalne. Taką odpowiedź można oczywiście uzyskać dopiero po przeprowadzeniu dogłębnej statystyki na podstawie uzyskanych odchyleń dla wszystkich elementów składających się na sieć neuronową i dla wszystkich wzorców uczących. Dzięki temu zaś, że informacja o odchyleniach rozproszona jest po wszystkich synapsach i progach sieci neuronowej, daje to dodatkowo możliwość wnioskowania, w którym miejscu sieci, z jaką intensywnością i dla których wzorców uczących występują problemy z dostosowaniem się sieci neuronowej do ciągu uczącego. Pozwala to między innymi na ukierunkowanie procesu uczenia poprzez bardziej inteligentne **sterowanie** sposobem zawieranego **kompromisu** pomiędzy wszystkimi wzorcami ciągu uczącego w dalszych epokach uczących.

Ponadto informacje te mogą stanowić wartościowe źródło informacji dla technik ontogenicznych, dynamicznie rekonfigurujących topologię sieci neuronowej w trakcie procesu uczenia. Informacje płynące z opisanych odchyleń mają charakter lokalny, tak więc sam neuron może zadecydować na ich podstawie o swojej rekonfiguracji. Nie trzeba podkreślać, iż jest to dużą zaletą, albowiem analizowanie całej architektury sieci neuronowej globalnie jest skomplikowane obliczeniowo, a także nie posiada potwierdzenia w biologii.

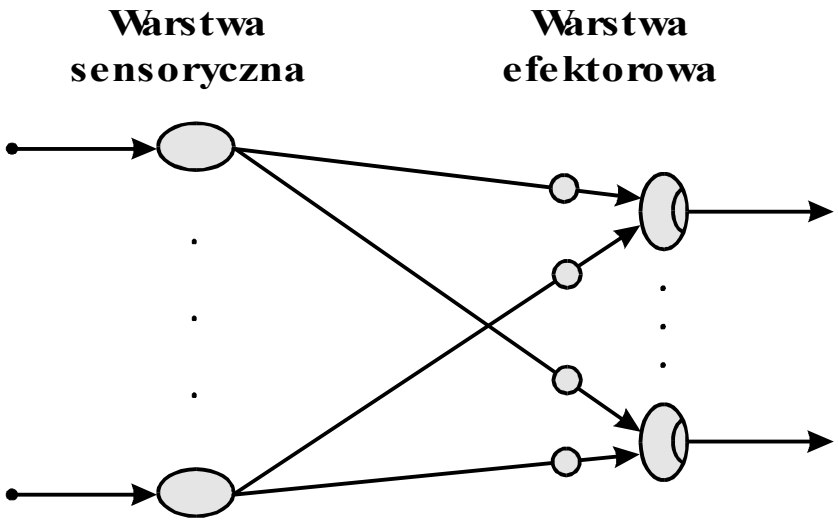
Matematyczne podstawy wprowadzonych w tym rozdziale koncepcji i pojęć zostaną opisane dokładniej w rozdziale 4.3.4. tej pracy. Przybliżona zostanie tam także możliwa do wykorzystania strategia **sterowania** procesem uczącym poprzez bardziej inteligentne obliczanie **kompromisu**.

4.3. Opis metody

Podstawowa wersja metody sterowanych kompromisów bazuje na dwóch warstwach sieci:

warstwie wejściowej, zwanej również sensoryczną,

warstwie wyjściowej, zwanej również efektorową.



Rysunek 4.8. Dwuwarstwowa sieć neuronowa z uwidocznioną warstwą sensoryczną, efektorową i łączącymi je połączeniami synaptycznymi.

Nauka sieci neuronowej odbywa się w kolejnych epokach uczących, w których rozważana jest korekta wszystkich wag i progów sieci. Każda **epoka ucząca** (krótko zwana dalej „epoką") składać się może z kilku etapów uczących. Ich ilość zależna jest od ilości warstw, wynikających z topologii wybranej sieci neuronowej. W każdym konkretnym etapie uczącym

zostaje przeprowadzona korekta pewnej wydzielonej części wag i progów. Dla sieci dwuwarstwowych, opisanych w tym rozdziale, każda epoka ucząca składa się tylko z jednego etapu uczącego. Pojęcie **etapu uczącego** (krótko zwanego dalej „etapem") dla sieci wielowarstwowej zostanie dokładniej wyjaśnione w jednym z następnych rozdziałów. Teraz wyjaśnione zostanie, na czym polega etap uczący dla sieci neuronowej o prostej topologii dwuwarstwowej, gdyż ułatwi to systematyczne poznanie proponowanej metody.

W każdym etapie uczącym dla każdego wzorca uczącego wykonywane są następujące dwie **fazy**:

1. Faza propagacji pobudzenia sieciowego,

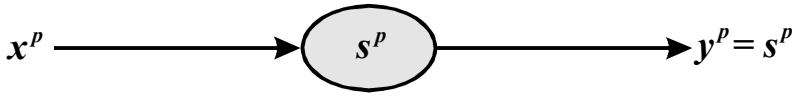
2. Faza wstecznej propagacji sygnału uczącego.

Faza druga połączona jest z obliczaniem nowych wartości wag i progów na podstawie informacji płynących zarówno od wejścia sieci (od sensorów) jak również od wyjścia sieci (od efektorów). Ten dwukierunkowy przepływ sygnałów rozważany jest kolejno dla każdego aktualnie uczonego wzorca uczącego. Korekta nie jest jednak przeprowadzana od razu, ale dopiero po wykonaniu powyżej wymienionych dwóch faz dla każdego wzorca uczącego, tzn. po przejrzeniu całego ciągu uczącego, przy czym kolejność wzorców uczących nie ma w tej metodzie żadnego znaczenia.

4.3.1. Faza propagacji pobudzenia sieciowego

W fazie pobudzenia sieciowego dany wzorzec propagowany jest od wejścia (sensorów) sieci do jej wyjścia (efektorów) w następujący sposób:

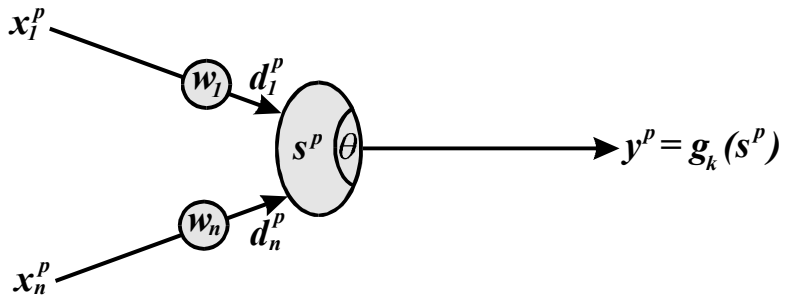
Najpierw pobudzane są kolejno wszystkie sensory (Rys. 4.9.) sygnałem podanym z wejścia sieci zgodnym z uczonym wzorcem uczącym. Dla sensorów wartość sygnału aktywacji równa jest wartości pobudzenia, a więc funkcja aktywacji sensorów jest funkcją identycznościową[2].



[2] Nie koniecznie funkcja aktywacji sensora musi być funkcją identycznościową. Można się pokusić o wyposażenie sensora w dodatkowe mechanizmy umożliwiające mu podczas nauki np. przeskalowanie danych wejściowych, bądź poddanie ich działaniu jakiejś nieliniowej funkcji aktywacji. Biologiczne sensory (receptory) też są wyposażone w różne mechanizmy umożliwiające im bardziej się wyczulić na bodźce lub na nie znieczulić.

Rysunek 4.9. Sensor (element wejściowy) sieci neuronowej. Funkcja aktywacji sensora jest funkcją identycznościową $y^p = s^p = x^p$.

Następnie pobudzeniu ulegają wszystkie efektory na podstawie sygnałów aktywacji sensorów, które zostają odpowiednio zważone w synapsach stojących na drodze sygnałów prowadzących od sensorów do efektorów. W praktyce wartość sygnału aktywacji dla efektorów obliczana jest podobnie jak dla neuronów, tzn. tak jak zostało to opisane w rozdziale 4.1.



Rysunek 4.10. Efektor (element wyjściowy) sieci neuronowej. Funkcja aktywacji efektora

$$y^p = g_k\left(s^p\right) = g_k\left(\sum_{i=1}^{n} d_i^p - \theta\right) = g_k\left(\sum_{i=1}^{n} x_i^p \cdot w_i - \theta\right).$$

W końcu otrzymane sygnały aktywacji efektorów są zarazem sygnałami wyjściowymi sieci neuronowej i odpowiedzią sieci na pobudzenie jej danym wzorcem uczącym. Wyjścia te w następnej fazie wstecznej propagacji sygnału uczącego będą służyły korekcie powstałych błędów sieci, jakie pojawiły się na jej wyjściu.

4.3.2. Faza wstecznej propagacji sygnału uczącego

Faza wstecznej propagacji sygnału uczącego rozpoczyna się od obliczenia pożądanej wartości stanu pobudzenia efektora S^p na podstawie pożądanej wartości sygnału wyjściowego Y^p dla ustalonego wzorca uczącego p pochodzącego od nauczyciela, tzn. z ciągu uczącego:

$$S^p = G_k\left(Y^p\right),$$

gdzie funkcja G_k dla opisanych w rozdziale 4.1. monotonicznych funkcji aktywacji przyjmuje postać funkcji odwrotnej:

$$G_k\left(Y^p\right) = g_k^{-1}\left(Y^p\right),$$

Jak się okaże w następnych rozdziałach, funkcja G_k nie zawsze musi być funkcją odwrotną do funkcji aktywacji g_k. Dzięki temu, że znana jest nie tylko wartość Y^p, lecz również wartość s^p, możliwe jest jednoznaczne (przy pewnych założeniach) obliczanie wartości S^p również dla funkcji niemonotonicznych, a nawet okresowych. Na razie jednak nasze rozważania ograniczone zostaną do funkcji opisanych w rozdziale 4.1., dla których funkcja G_k przyjmuje następującą postać:

$$S^p = G_1\left(Y^p\right) = g_1^{-1}\left(Y^p\right) = \frac{Y^p}{\beta}$$

$$S^p = G_2\left(Y^p\right) = g_2^{-1}\left(Y^p\right) = \frac{\left|Y^p\right| \cdot Y^p}{\beta}$$

$$S^p = G_3\left(Y^p\right) = g_3^{-1}\left(Y^p\right) = \frac{\operatorname{sgn}\left(Y^p\right) \cdot \sqrt{\left|Y^p\right|}}{\beta}$$

Po obliczeniu wartości pożądanej stanu pobudzenia efektora S^p, może zostać wyznaczona lokalnie wartość błędu aktualnego stanu pobudzenia s^p dla ustalonego wzorca uczącego *p* według wzoru:

$$\sigma_s^p = S^p - s^p.$$

Na powstanie błędu σ_s^p mają bezpośredni wpływ wartości postsynaptycznych pobudzeń dendrytów d_i^p ($i=1,...,N$) oraz wartość progu θ zgodnie ze wzorem 4.1.1., dlatego korekta powstałego błędu σ_s^p powinna zostać rozłożona (zgodnie z przyjętymi kryteriami) na poszczególne dendryty oraz powinna także spowodować ustalenie wartości korekty błędu progu. Kryteria rozkładu błędu na wyżej wymienione składowe mogą być różne. Można brać pod uwagę wartości postsynaptycznych pobudzeń dendrytycznych d_i^p, wartości presynaptycznych pobudzeń synaptycznych x_i^p, wartości aktualnych wag w_i^p itp. Ważne jest, aby przyjęte kryteria ukierunkowywały proces zmian wag i progów w kierunku minimalizacji opisanych wcześniej odchyleń od kompromisu, który zostanie uzyskany na ich podstawie, i prowadziły w związku z tym do rozwiązania zadanego ciągiem uczącym z jak największą

dokładnością. Minimalizacja wszystkich owych odchyleń jest zaś równoważna globalnej minimalizacji błędu sieci neuronowej, czyli znalezieniu minimum globalnego funkcji błędu.

W podstawowej wersji metody sterowanych kompromisów przyjęto uzależnienie rozkładu korekty błędu od wartości postsynaptycznych pobudzeń dendrytycznych d_i^p. W tym celu najpierw obliczana jest suma modułów wszystkich postsynaptycznych pobudzeń dendrytycznych d_i^p oraz wartości pobudzenia, jaki wnosi próg, według następującego wzoru:

$$M_s^p = \sum_{i=1}^{N} \left| d_i^p \right| + \left| -\theta \right| = \sum_{i=1}^{N} \left| d_i^p \right| + \left| \theta \right|.$$

Następnie dla każdego wzorca *p* każdy dendryt (jak również próg) obarczany jest taką częścią błędu σ_s^p, jaka wynika z jego udziału w powstałym stanie pobudzenia efektora s^p, tzn.:

dla progu:

$$\sigma_\theta^p = \sigma_s^p \cdot \frac{\left| -\theta^p \right|}{M_s^p} = \sigma_s^p \cdot \frac{\left| \theta^p \right|}{M_s^p},$$

a następnie obliczana jest nowa wartość progu dla ustalonego wzorca uczącego *p*:

$$\Xi^p = \theta^p + \sigma_\theta^p,$$

i dla dendrytów:

$$\sigma_{d_i}^p = \sigma_s^p \cdot \frac{\left| d_i^p \right|}{M_s^p},$$

a następnie obliczana jest nowa wartość wagi dla ustalonego wzorca uczącego *p*:

$$W_i^p = \frac{d_i^p + \sigma_{d_i}^p}{x_i^p},$$

która umożliwia wyznaczenie korekty błędu wagi dla ustalonego wzorca uczącego *p*:

$$\sigma_{w_i}^p = W_i^p - w_i^p.$$

4.3.3. Faza obliczania kompromisu

Korekty błędów wag i progów, obliczone kolejno dla wszystkich wzorców uczących, służą (po przeglądnięciu całego ciągu uczącego) do wyznaczenia wypadkowych poprawek,

które będą wprowadzone do sieci celem polepszenia (w kolejnej epoce uczenia) jej działania. Poprawki te nazywane będą **kompromisem** zawartym pomiędzy wzorcami uczącymi i mogą mieć w najprostszej postaci formę np. średniej arytmetycznej, stanowiącej podstawę wynikowej korekty wag i progów. W związku z tym poprawki wyznacza się następująco:

Średnia arytmetyczna proponowanych korekt błędów wag dla wszystkich wzorców uczących wynosi:

$$\sigma_{w_i} = \frac{\sum_{p=1}^{Q} \sigma_{w_i}^{p}}{Q},$$

więc nowa waga obliczona na bazie tej średniej:

$$W_i = w_i + \sigma_{w_i},$$

Średnia arytmetyczna proponowanych korekt błędów progów dla wszystkich wzorców uczących wynosi:

$$\sigma_{\theta} = \frac{\sum_{p=1}^{Q} \sigma_{\theta}^{p}}{Q},$$

i nowy próg obliczony na bazie tej średniej:

$$\Xi = \theta + \sigma_{\theta}.$$

Należy podkreślić, iż wynikowe kompromisowe korekty wag i progów nie muszą być obliczane na bazie akurat średniej arytmetycznej. Dodatkowo przy wyznaczaniu nowych wartości tych parametrów można wziąć pod uwagę na przykład:

- odchylenia od obliczonego kompromisu, tutaj średniej arytmetycznej (ale można próbować wykorzystać również średnią ważoną, geometryczną i inne),
- uniezależnianie wielkości korekt od reprezentacji liczbowej wzorców uczących,
- położenie nacisku na kierunek, w jakim chce podążać większość wzorców uczących,
- wzięcie za priorytet najbardziej poszkodowane wzorce w trakcie wyznaczania owego kompromisu,
- eliminację z ciągu uczącego artefaktów powodujących zakłócenia procesu uczenia związanego z obliczaniem poszczególnych wartości kompromisowych.

Te parametry mogą w bardziej zaawansowanych adaptacjach metody sterowanych kompromisów posłużyć do bardziej efektywnego sterowania procesem nauki poprzez bardziej inteligentne obliczanie wynikowych kompromisowych korekt błędów dla poszczególnych wag i progów przy uwzględnieniu proponowanych korekt tych błędów dla poszczególnych wzorców ciągu uczącego.

4.3.4. Faza wyznaczania odchyleń

Zanim przedstawiony zostanie sposób obliczania opisanych w rozdz. 4.2.4. odchyleń od uzyskanego kompromisu, przypomnijmy znaczenie używanych w tym rozdziale oznaczeń:

w_i – *i*-ta waga synaptyczna rozważanego neuronu lub efektora,

W_i^p – proponowana nowa *i*-ta waga synaptyczna rozważanego neuronu lub efektora, wyznaczona dla ustalonego wzorca uczącego *p*,

W_i – kompromisowa nowa *i*-ta waga synaptyczna rozważanego neuronu lub efektora dla całego ciągu uczącego (nazywana krótko kompromisem zawartym dla *i*-tej wagi),

θ – próg rozważanego neuronu lub efektora,

Ξ^p – proponowany nowy próg rozważanego neuronu lub efektora, wyznaczony dla ustalonego wzorca uczącego *p*,

Ξ – kompromisowy nowy próg rozważanego neuronu lub efektora dla całego ciągu uczącego (nazywany krótko kompromisem zawartym dla progu).

Na podstawie wyżej wymienionych oznaczeń zdefiniowane zostaną odchylenia, leżące u podstaw dalszych rozważań związanych z możliwością sterowania procesem nauki ukierunkowanym na bardziej efektywne zmierzanie do przyjętego końcowego celu, jakim jest globalna minimalizacja błędu sieci neuronowej dla zadanego ciągu uczącego:

$\hat{\delta}_{W_i}^p = W_i^p - W_i$ – odchylenie od uzyskanego kompromisu dla rozważanej *i*-tej wagi (W_i), wyznaczone dla ustalonego wzorca uczącego *p* (zwane krótko: **odchyleniem wagi dla wzorca uczącego**),

$\hat{\delta}_{\Xi}^{p} = \Xi^{p} - \Xi$ – odchylenie od uzyskanego kompromisu dla rozważanego progu (Ξ), wyznaczone dla ustalonego wzorca uczącego *p* (zwane krótko: **odchyleniem progu dla wzorca uczącego**),

$\delta_{W_i}^{p} = \left|\hat{\delta}_{W_i}^{p}\right|$ – bezwzględne odchylenie od uzyskanego kompromisu dla rozważanej *i*-tej wagi (W_i), wyznaczone dla ustalonego wzorca uczącego *p* (zwane krótko: **bezwzględnym odchyleniem wagi dla wzorca uczącego**),

$\delta_{\Xi}^{p} = \left|\hat{\delta}_{\Xi}^{p}\right|$ – bezwzględne odchylenie od uzyskanego kompromisu dla rozważanego progu (Ξ), wyznaczone dla ustalonego wzorca uczącego *p* (zwane krótko: **bezwzględnym odchyleniem progu dla wzorca uczącego**),

$$\delta_{W_i} = \frac{\sum_{p=1}^{Q} \delta_{W_i}^{p}}{Q}$$ – bezwzględne odchylenie od uzyskanego kompromisu dla rozważanej i-tej wagi synaptycznej (W_i) dla całego ciągu uczącego (zwane krótko: **bezwzględnym odchyleniem wagi dla ciągu uczącego**),

$$\delta_{\Xi} = \frac{\sum_{p=1}^{Q} \delta_{\Xi}^{p}}{Q}$$ – bezwzględne odchylenie od uzyskanego kompromisu dla rozważanego progu (Ξ) dla całego ciągu uczącego (zwane krótko: **bezwzględnym odchyleniem progu dla ciągu uczącego**),

$$\delta_{r}^{p} = \frac{\sum_{i_r}^{n_r} \left|W_{i_r}^{p} - W_{i_r}\right| + \left|\Xi_{r}^{p} - \Xi_{r}\right|}{n_r + 1}$$ - bezwzględne odchylenie od uzyskanego kompromisu dla rozważanego neuronu lub efektora *r* oraz dla ustalonego wzorca uczącego *p*, będące sumą bezwzględnych odchyleń od uzyskanego kompromisu dla wszystkich wag i progów całej sieci neuronowej (zwane krótko: **bezwzględnym odchyleniem neuronu dla wzorca uczącego**),

$\delta_r = \frac{\sum_{p=1}^{Q} \delta_r^p}{Q}$ – bezwzględne odchylenie od uzyskanego kompromisu dla rozważanego neuronu lub efektora *r* dla całego ciągu uczącego (zwane krótko: **bezwzględnym odchyleniem neuronu dla ciągu uczącego**),

$\delta^p = \frac{\sum_{r=1}^{R} \delta_r^p}{R}$ – bezwzględne odchylenie od uzyskanego kompromisu dla rozważanego wzorca uczącego *p*, będące sumą bezwzględnych odchyleń od uzyskanego kompromisu dla wszystkich wag i progów całej sieci neuronowej (R – ilość wszystkich neuronów i efektorów sieci) (zwane krótko: **bezwzględnym odchyleniem wzorca uczącego**),

Powyżej zdefiniowane odchylenia od uzyskanego kompromisu można wyznaczyć podczas jednego przeglądnięcia ciągu uczącego. Ponadto czynność tą można połączyć z pierwszym etapem uczącym kolejnej epoki uczącej. Dzięki temu przed aktualizacją dokonywaną dla pierwszego etapu danej epoki sieć dysponuje informacjami odnośnie wszystkich zdefiniowanych powyżej odchyleń. W takim przypadku przed dokonaniem aktualizacji sieć zawsze może się wesprzeć dodatkowymi informacjami płynącymi z wartości poszczególnych odchyleń i na tej podstawie bardziej inteligentnie sterować zawieranym w fazie aktualizacji kompromisem pomiędzy wzorcami uczącymi. Przez **kompromis** skrótowo rozumie się wszystkie kompromisy cząstkowe zawierane dla poszczególnych elementów sieci neuronowej – w szczególności dla wag i progów sieci. Zdefiniowane odchylenia informują sieć neuronową o wielu korzystnych jak również niekorzystnych sprawach odgrywających się w niej podczas procesu uczenia, mianowicie:

- Na podstawie wartości **bezwzględnego odchylenia wzorca uczącego** można się dowiedzieć, które wzorce uczące sprawiają sieci neuronowej najwięcej „kłopotów”.
- W przypadku kłopotów z dostosowaniem się sieci do określonego wzorca uczącego można na podstawie **bezwzględnego odchylenia neuronu** dla tego wzorca zlokalizować w sieci te neurony, gdzie odchylenia te są najbardziej kłopotliwe.
- Informacja o kłopotach z dostosowaniem się pewnego neuronu do danego wzorca uczącego lub całego ciągu uczącego może posłużyć do zlokalizowania wag i progów,

które charakteryzują się największymi **bezwzględnymi odchyleniami wag** i **bezwzględnymi odchyleniami progów** dla rozważanego wzorca uczącego lub całego ciągu uczącego.

- Informacje o kłopotach w dostosowaniu się różnych części sieci neuronowej do ciągu uczącego lub części wzorców uczących mogą służyć do podjęcia decyzji o zmianie strategii rozkładu korekty pozostałego jeszcze błędu na poszczególne parametry sieci neuronowej, bądź też mogą stać u podłoża rekonfiguracji pewnej części architektury sieci neuronowej, która stanowi źródło niezgody (tzn. występuje brak akceptowalnego kompromisu) pomiędzy wszystkimi lub częścią wzorców uczących.

Wychodząc z założenia, że zależy nam na uzyskaniu jak najlepszych wyników (tzn. jak najmniejszego błędu sieci przy jak najlepszej zdolności do uogólniania), można podjąć próby zmierzające do wykorzystania zdefiniowanych w tej pracy odchyleń do opracowania metod zmierzających do zoptymalizowania jak architektury sieci neuronowej tak samego procesu dostosowywania jej parametrów.

4.3.5. Sterowanie procesem uczenia

Przez sterowanie procesem uczenia rozumie się w praktyce sterowanie sposobem rozkładu błędu na poszczególne składniki sieci neuronowej (w szczególności wagi i progi) oraz sterowanie sposobem zawieranego kompromisu pomiędzy wzorcami uczącymi. Bardziej inteligentny sposób rozkładu błędu może doprowadzić do szybszej zbieżności poprzez wyeliminowanie pewnych punktów niezgody pomiędzy wzorcami, zaś sposób zawieranego kompromisu determinuje, w jakim stopniu poszczególne wzorce uczące mają wpływ na proces uczący. W świetle tej terminologii można zdefiniować, co oznacza minimum lokalne i globalne funkcji błędu. Minimum lokalne będzie utrzymującym się kompromisem zawartym pomiędzy wzorcami uczącymi, charakteryzujący się niezadowalająco dużymi odchyleniami dla poszczególnych wzorców uczących. Minimum globalne będzie również pewnym utrzymującym się kompromisem, który jednak będzie się w ogólności charakteryzował minimalnymi możliwymi odchyleniami dla poszczególnych wzorców. Kwestia tego, czy odchylenia te (a co za tym idzie również błąd sieci) będą zadowalająco małe, zależy już wtedy tylko od konkretnego problemu. W razie uzyskania niezadowalających wyników trzeba podjąć próbę rekonfiguracji architektury sieci, która w rozważanej postaci może nie pozwalać na lepsze jej dostosowanie do zadanego problemu. W przypadku konieczności rekonfiguracji

sieci neuronowej również bardzo przydatne są informacje o poszczególnych odchyleniach, które pozwalają ukierunkować proces jej rekonfiguracji.

4.3.6. Warunki zatrzymania procesu uczenia

W związku z procesem uczenia pojawia się pytanie, ilu epok potrzeba, by można było uznać, że sieć jest już wystarczająco nauczona – czyli potrzebne są warunki zatrzymania algorytmu. Warunki te mogą być uzależnione od wielu różnych parametrów w zależności od tego, jaką wagę przypisujemy poszczególnym z nich. Badać je należy oczywiście dopiero po ukończeniu całej epoki uczącej. Przedmiotem badania powinny być przede wszystkim:

- błąd, jaki powstaje na wyjściu sieci neuronowej dla wzorców uczących,
- błąd, jaki powstaje na wyjściu sieci neuronowej dla wzorców testujących,
- wartości odchyleń od uzyskanego kompromisu dla poszczególnych wzorców uczących,
- wartości odchyleń od uzyskanego kompromisu dla poszczególnych wag i progów.

4.4. Rozszerzenia metody

Metoda sterowanych kompromisów może zostać uogólniona i rozszerzona o nowe możliwości tak, by była w stanie sprostać również bardziej wymagającym zadaniom. Do użytecznych rozszerzeń zaliczyć można mianowicie:

- Uniezależnienie korekty błędów od reprezentacji liczbowej poszczególnych wzorców.
- Wzbogacenie podstawowych funkcji aktywacji neuronów o inne funkcje usprawniające bądź upraszczające naukę lub architekturę sieci neuronowej, bądź też poprawiające jakość uzyskanego uogólnienia.
- Uczenie sieci neuronowych zawierających neurony pośrednie lub całe warstwy ukryte.

Próby wzbogacania i uogólniania zaprezentowanej metody są aktualnie w stanie intensywnych badań. W tej pracy opisano tylko wybraną ich część. Opisane niżej rozszerzenia nie zawsze przedstawiają finalne rozwiązanie przedstawionego problemu, lecz bardziej wskazują na potencjał zaproponowanej metody i na różne kierunki poszukiwań, do których stanowi ona inspirację – podkreślając zarazem dalszą potrzebę takich właśnie badań.

4.4.1. Uniezależnienie od reprezentacji liczbowej wzorców uczących

Wzorce uczące reprezentowane są przez liczby, które z natury przyjmują różne wartości. W procesie uczącym byłoby jednak nie wskazane, gdyby np. wzorce reprezentowane przez duże liczby miały wyższy priorytet uczenia niż te reprezentowane liczbami małymi. Przy obliczaniu wynikowej kompromisowej korekty wag i progów bierze się pod uwagę korekty obliczone dla poszczególnych wzorców ciągu uczącego, a więc nie uwzględnia się stosunku wielkości korekty do wartości, których ta korekta dotyczy. W takim układzie do wzorców reprezentowanych większymi liczbami odnosi się zazwyczaj większa korekta i *vice versa*. Taka sytuacja sprawia, że przy obliczaniu wynikowej kompromisowej korekty wzorce nie są traktowane równorzędnie. Ażeby uniezależnić naukę od reprezentacji liczbowej konkretnych wzorców uczących, trzeba znormalizować poszczególne korekty błędów przez wprowadzenie współczynnika odzwierciedlającego wielkość reprezentacji wzorców dla danego neuronu sieci. Współczynnikiem takim, który określa bezwzględną wielkość pobudzeń lub hamowań neuronu, jest używany już w rozdziale 4.3.2. współczynnik M_s^p, który przyjmuje odpowiednio większe jak i mniejsze wartości w zależności od bezwzględnej wartości reprezentacji poszczególnych wzorców. Wartość M_s^p dla danego wzorca p można więc przyjąć jako współczynnik normalizujący. Wzory służące do obliczania wynikowych kompromisowych korekt z rozdziału 4.3.3. opartych na średniej arytmetycznej przyjmą teraz następującą postać:

dla wag:

$$\sigma_{w_i} = \frac{\sum_{p=1}^{Q} \frac{\sigma_{w_i}^p}{M_s^p}}{\sum_{p=1}^{Q} \frac{1}{M_s^p}},$$

dla progów:

$$\sigma_{\theta} = \frac{\sum_{p=1}^{Q} \frac{\sigma_{\theta}^p}{M_s^p}}{\sum_{p=1}^{Q} \frac{1}{M_s^p}},$$

W wyniku opisanej powyżej modyfikacji uzyskujemy względne zrównoważenie wielkości korekt dla poszczególnych wzorców niezależnie od ich reprezentacji liczbowej, co powinno w większości przypadków pozytywnie wpływać na przebieg procesu uczenia i na jego wyniki.

4.4.2. Niestandardowe funkcje aktywacji neuronów

Funkcje aktywacji neuronów w istotny sposób decydują o możliwościach dostosowania całej sieci neuronowej do problemu zadanego ciągiem uczącym, jak również wpływają one na możliwości interpolacyjne i ekstrapolacyjne, jakie nauczona sieć będzie posiadała. Sztywne trzymanie się w tym zakresie biologicznych modeli jest tylko po części uzasadnione i nie zawsze bywa związane z postawionym celem obliczeniowym, dla którego tworzona jest rozważana sieć. Jeśli zatem mamy na uwadze przesłanki badawcze związane z biologicznymi tajnikami rzeczywistych neuronów i badaniami modelowymi biologicznego mózgu, wtedy zaiste słuszne okaże się modelowanie neuronów ściśle wedle danych dostarczonych przez neurofizjologię i neuroanatomię. W tych przypadkach z pomocą specjalnie na ten cel ukierunkowanych badań poszukuje się funkcji przejścia modelowanych neuronów, wybierając je w oparciu o kryteria związane z ich biologiczną odpowiedniością[3]. Jeśli jednak ważne są dla nas obliczeniowe możliwości sieci neuronowych dla jak najszerszej gamy rzeczywistych problemów, wtedy bardziej rozsądne wydaje się wyposażenie sztucznych neuronów w takie możliwości, które umożliwiłyby osiągnięcie tego celu – bez względu na to, czy są to właściwości biologicznie prawdopodobne, czy też nie.

Udowodnione zostało, iż zastosowanie liniowych funkcji aktywacji neuronów nie pozwala rozwiązywać ogólnych zadań decyzyjnych, a tylko problemy liniowo separowalne (M. Minsky i S. Papert, 1969). W podobny sposób można by było pokazać, iż sieć neuronowa oparta na nieliniowych monotonicznych funkcjach aktywacji nie będzie się nadawała do ekstrapolacji problemów natury periodycznej. Co prawda znane są twierdzenia, iż sieć neuronowa z sigmoidalnymi (a więc monotonicznymi) funkcjami przejścia może modelować dowolną funkcję, jednak dowód tego twierdzenia przeprowadza się przy założeniu, że sieć neuronowa ma potencjalnie nieskończoną ilość neuronów, co jednak jest założeniem w oczywisty sposób nie realizowalnym w przypadku praktycznych obliczeń komputerowych, a nawet niepoprawnym w odniesieniu do biologicznego mózgu, którego liczba neuronów jest wprawdzie bardzo duża ($\sim 10^{11}$), ale jednak skończona.

Jednym ze skutków ograniczonej wydolności sieci zbudowanych z neuronów o monotonicznych charakterystykach jest ich ograniczona przydatność w problemach

[3] Porównaj tylko temu zagadnieniu poświęcone prace, na przykład: Tadeusiewicz R., Lazarewicz M.T.: Nowy paradygmat neurobiologii obliczeniowej: „*experiment in computo*". Materiały V Krajowej Konferencji „Modelowanie Cybernetyczne Systemów Biologicznych", Zakład Biocybernetyki Collegium Medicum UJ, Kraków 2000 ss. 13-40 **lub** Tadeusiewicz R., Lazarewicz M.T.: Postępy i sukcesy w biocybernetycznych pracach u podstaw sztucznej inteligencji. IV Krajowa Konferencja Naukowa nt. Sztuczna Inteligencja, Akademia Podlaska, Siedlce 2000, ss. 9-34

przewidywania przyszłości. Nawet dla człowieka ten problem jest wyjątkowo trudny, mimo iż mózg ludzki ma do dyspozycji niewyobrażalnie wielką liczbę rzędu stu miliardów neuronów. Być może trudność trafnego przewidywania przyszłości pojawia się w naszym zachowaniu właśnie dlatego, że znakomita większość zjawisk w przyrodzie ma naturę quasi-periodyczną. Jeśli więc przy tworzeniu neurokomputerów nacisk zostanie położony na obliczeniowe możliwości tworzonych sieci neuronowych, wtedy można rozszerzyć funkcje aktywacji neuronów o funkcje niemonotoniczne. Takie postępowanie jest jednak możliwe wyłącznie przy założeniu, iż metoda ucząca pozwala na uczenie sieci neuronowych zbudowanych z neuronów o takich funkcjach aktywacji. Wiele klasycznych metod uczenia sieci wykazuje w tym zakresie istotne ograniczenia, co jest jednym z powodów unikania w problematyce sztucznych sieci niemonotonicznych funkcji aktywacji (z potwierdzającym tę regułę wyjątkiem sieci RBF, które jednak są uczone w sposób nie klasyczny).

Metoda sterowanych kompromisów może zostać z powodzeniem zmodyfikowana w taki sposób, by nadawała się do uczenia sieci opartych również na niemonotonicznych funkcjach aktywacji. Modyfikacji takiej można dokonać przy założeniu, że zdefiniowana zostanie funkcja $G_k\left(Y^p\right)$ służąca do obliczania pożądanej wartości stanu pobudzenia efektora S^p dla wzorca uczącego *p* na podstawie wartości sygnału uczącego Y^p. Jest sprawą oczywistą, że zdefiniowanie funkcji (a właściwie odwzorowania) G_k nie jest w tym przypadku takie proste jak dla funkcji monotonicznych, gdzie funkcja G_k przyjmuje postać funkcji odwrotnej do g_k (por. rozdział IV.3.2). Dla niemonotonicznych funkcji aktywacji funkcja g_k^{-1} nie istnieje i trzeba się sporo natrudzić, by jednoznacznie zdefiniować, jak w takim przypadku obliczać wartość S^p dla różnych wartości Y^p. Jak wiadomo dla periodycznych funkcji aktywacji istnieje nieskończona ilość wartości S^p, dla których $g_k\left(S^p\right)=Y^p$. Na szczęście z pomocą przychodzi tu znana z fazy propagacji pobudzenia sieciowego wartość stanu pobudzenia efektora (neuronu) s^p, która może się przysłużyć do jednoznacznego określenia wartości S^p dla wzorca uczącego *p*. Funkcja G_k staje się wtedy funkcją dwóch zmiennych:

$$S^p = G_k\left(Y^p, s^p\right).$$

Funkcję G_k dla funkcji periodycznych można zdefiniować w różnoraki sposób, jednak ze względu na stabilność nauki wartość S^p powinna być jak najbliższa wartości s^p spośród wszystkich możliwych. Można więc przyjąć następującą definicję funkcji G_k:

$$G_k\left(Y^p, s^p\right) = \min\left\{S^p : \left|S^p - s^p\right| = \min\left\{\left|\hat{S}^p - s^p\right| : g_k\left(\hat{S}^p\right) = Y^p\right\}\right\}$$

lub

$$G_k\left(Y^p, s^p\right) = \max\left\{S^p : \left|S^p - s^p\right| = \min\left\{\left|\hat{S}^p - s^p\right| : g_k\left(\hat{S}^p\right) = Y^p\right\}\right\}.$$

Ze względu na ogólność powyższej definicji można używać różnych funkcji jako funkcji aktywacji, jednak dla potrzeb tej pracy zawężono się do następujących okresowych funkcji aktywacji:

$y^p = g_4\left(s^p\right) = \tan\left(\beta \cdot s^p\right)$ - funkcja tangensoidalna (Rys. 11.)

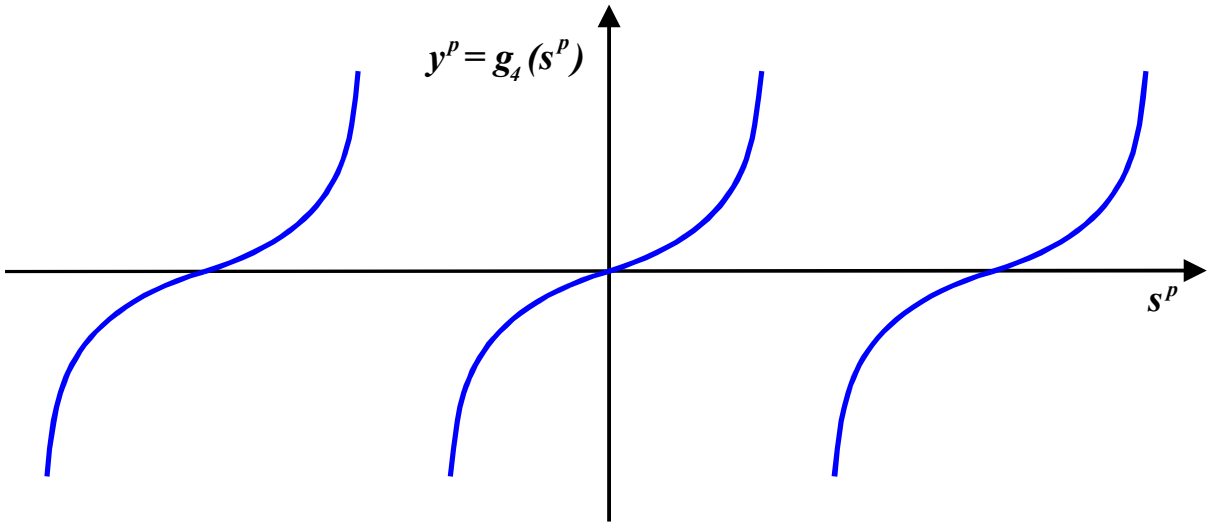
$y^p = g_5\left(s^p\right) = \tan\left(\beta \cdot s^p\right) \cdot h_{\pm}\left(\beta \cdot s^p\right)$ - hybrydowa funkcja tangensoidalna (Rys. 12.)

$y^p = g_6\left(s^p\right) = \sin\left(\beta \cdot s^p\right) \cdot h_{\pm}\left(\beta \cdot s^p\right)$ - hybrydowa funkcja sinusoidalna (Rys. 13.)

$y^p = g_7\left(s^p\right) = \sin\left(\beta \cdot s^p\right)$ - funkcja sinusoidalna (Rys. 14.)

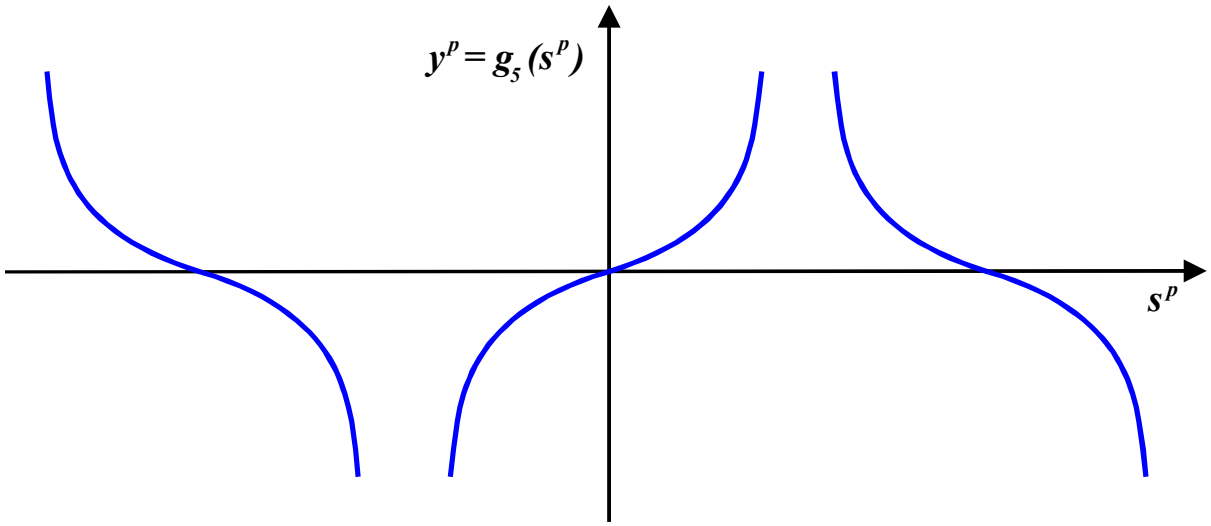
gdzie:

$$h_{\pm}(x) = \begin{cases} +1 & gdy\, x \in \bigcup_{k=-\infty}^{+\infty}\left(\frac{\pi(4k-1)}{2}, \frac{\pi(4k+1)}{2}\right] \\ -1 & gdy\, x \in \bigcup_{k=-\infty}^{+\infty}\left(\frac{\pi(4k+1)}{2}, \frac{\pi(4k+3)}{2}\right] \end{cases}.$$



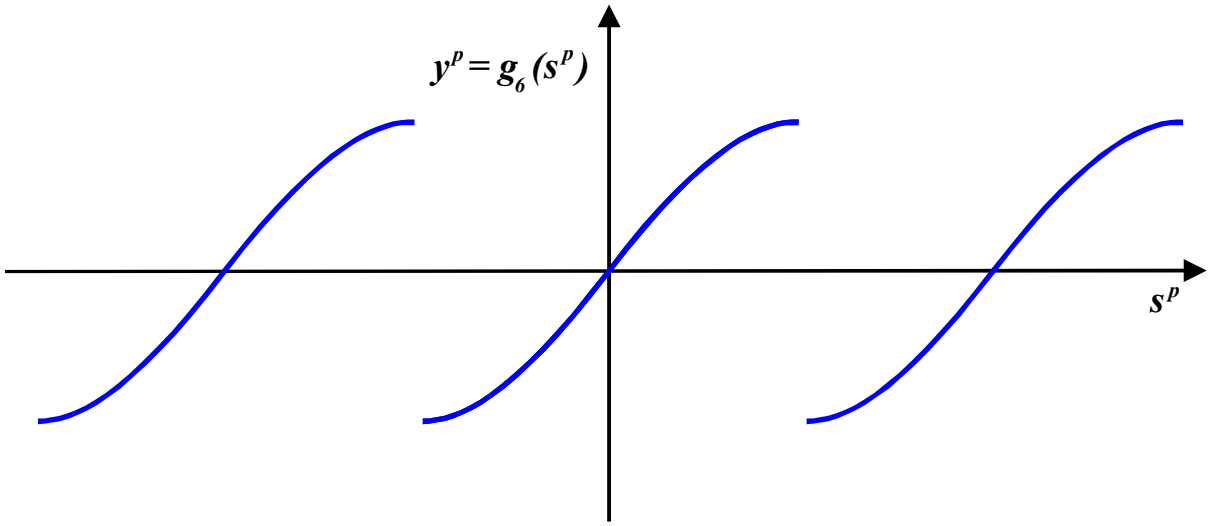
Rysunek 4.11. Okresowa funkcja aktywacji - tangensoidalna:

$g_4(s^p) = (\tan \beta \cdot s^p)$ taka, że $g_4 : R \to R$



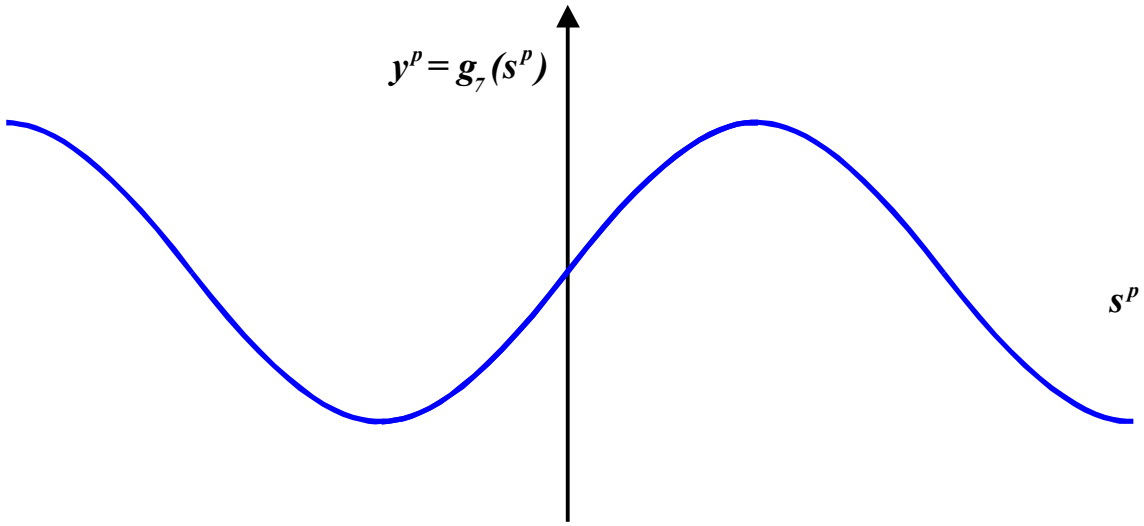
Rysunek 4.12. Okresowa funkcja aktywacji – hybrydowa tangensoidalna:

$g_5(s^p) = \tan(\beta \cdot s^p) \cdot h_{\pm}(\beta \cdot s^p)$ taka, że $g_5 : R \to R$



Rysunek 4.13. Okresowa funkcja aktywacji – hybrydowa sinusoidalna:

$$g_6\left(s^p\right) = \sin\left(\beta \cdot s^p\right) \cdot h_{\pm}\left(\beta \cdot s^p\right) \text{ taka, że } g_6 : R \to [-1,+1]$$



Rysunek 4.14. Okresowa funkcja aktywacji – sinusoidalna:

$$g_7\left(s^p\right) = \sin\left(\beta \cdot s^p\right) \text{ taka, że } g_7 : R \to [-1,+1]$$

Dla powyższych przykładowych okresowych funkcji aktywacji g_k wyznaczamy funkcje G_k następująco:

$$S^p = G_4\left(Y^p, s^p\right) = \frac{\arctan\left(Y^p\right)}{\beta} + K\left(s^p\right) \cdot \pi ,$$

$$S^p = G_5\left(Y^p, s^p\right) = \frac{\arctan\left(Y^p \cdot h_{\pm}\left(\beta \cdot s^p\right)\right)}{\beta} + K\left(s^p\right) \cdot \pi ,$$

$$S^p = G_6\left(Y^p, s^p\right) = \frac{\operatorname{arcsin}\left(Y^p \cdot h_{\pm}\left(\beta \cdot s^p\right)\right)}{\beta} + K\left(s^p\right) \cdot \pi ,$$

$$S^p = G_7\left(Y^p, s^p\right) = \frac{\operatorname{arcsin}\left(Y^p\right)}{\beta} + K\left(s^p\right) \cdot \pi ,$$

gdzie

$$K(s) = \left\lfloor \frac{\left\lfloor \frac{2\beta \cdot s}{\pi} \right\rfloor + \operatorname{sgn}(s)}{2} \right\rfloor ,$$

przy czym $\lfloor x \rfloor$ oznacza całkowitoliczbową część wartości x powstałą przez obcięcie części ułamkowej.

Postawmy sobie pytanie: Czy – widząc komplikacje, jakie temu towarzyszą - mimo wszystko warto stosować periodyczne funkcje aktywacji? Czy z powodzeniem nie można by było ograniczyć się do stosowanych obecnie funkcji aktywacji, które znajdują pewne biologiczne uzasadnienie? Otóż, jak zostanie pokazane w rozdziale V., zastosowanie okresowych funkcji aktywacji umożliwia rozwiązanie pewnych zadań, których rozwiązanie klasycznymi metodami neuronowymi z funkcjami aktywacji w dotychczas stosowanej postaci okazało się niemożliwe albo bardzo trudne. Dotyczy to np. klasycznego problemu dwóch spiral (opisanego dokładniej w rozdziale V). W przypadku użycia niemonotonicznej funkcji aktywacji rozwiązanie tego problemu okazało się możliwe (z poprawną interpolacją i ekstrapolacją wyników uczenia!) przy użyciu sieci neuronowej składającej się z pojedynczego efektora! Jest to duży sukces, gdyż - jak wiadomo - problem ten bardzo trudno rozwiązuje się nawet w bardzo dużych sieciach - przy założeniu użycia typowych sieci neuronowych, np. klasy MLP.

W analogiczny sposób, jak to pokazano wyżej, może zostać zdefiniowana funkcja G_k dla wielu różnych funkcji aktywacji g_k, dlatego użytkownik może wybierać spośród bardzo wielu potencjalnie dostępnych funkcji aktywacji, używając ich w zależności od potrzeb. Należy zwrócić uwagę, że użycie niemonotonicznych funkcji aktywacji daje nowe możliwości, ale rodzi także nowe problemy. Związane są one głównie z efektywnością

procesu uczenia, gdyż w związku z zastosowaniem periodycznych (czyli zarazem nie monotonicznych) funkcji aktywacji już dla sieci dwuwarstwowych pojawia się problem minimów lokalnych. Ogromnie trudno jest tak prowadzić proces uczenia, by nie blokować go w punktach odpowiadających lokalnym minimom funkcji błędów, których jest dodatkowo bardzo dużo – dla funkcji periodycznych używanych jako funkcje przejścia jest ich nieskończona ilość! Zastosowanie sinusoidalnej funkcji aktywacji, dla której $g_k : R \to [-1,+1]$, wymusza zatem automatycznie kolejne zmiany algorytmu uczenia sieci neuronowej, w związku z zawężeniem zakresu zbioru wartości funkcji. Zastosowanie funkcji sinusoidalnych tylko w warstwie efektorowej upraszcza ten problem i pozwala ograniczyć uwagę badacza wyłącznie do skalowania sygnałów pochodzących od nauczyciela (z ciągu uczącego), tak aby należały one do zakresu $[-1,+1]$.

4.4.3. Uczenie sieci neuronowych zawierających neurony ukryte

Dwuwarstwowe sieci neuronowe często nie wystarczają do efektywnego rozwiązywania bardziej skomplikowanych zadań, albo nie dostarczają rozwiązań z wymaganą dokładnością, dlatego na podobieństwo struktur neuronów zawartych w korze mózgowej - stworzono topologie sieci neuronowych, w których neurony uporządkowane są w kilku warstwach. Przez dłuższy okres czasu nie znany był jednak algorytm, który by umożliwiał uczenie takich architektur sieciowych. Pierwszą metodą, która pozwoliła na uczenie topologii zawierających dodatkowe warstwy (nazywane „ukrytymi"), była Metoda Propagacji Wstecznej (BP – *Back Propagation*), która jest często skuteczna, jednak ma tę ważną cechę, że korzysta z bazy koncepcyjnej metod optymalizacyjnych, a w szczególności z metod gradientowych. Jak wynika z dyskusji przeprowadzonej na wstępie tej pracy, metody takie nie gwarantują jednak efektywnego zmierzania do minimum globalnego funkcji błędu, lecz tylko poszukują minimum lokalnego w kierunku, na jaki wskazuje gradient rozważanej funkcji. Jest to istotna i trudna do uniknięcia wada tych metod.

Metoda sterowanych kompromisów, wprowadzona w tej pracy, ma jednak także i tę zaletę, że może zostać uogólniona również dla potrzeb uczenia sieci wielowarstwowych. Co więcej, metoda ta da się zastosować również dla tych sieci, które nie bazują na jawnych warstwach, ale posiadają pewną ilość niecyklicznie połączonych neuronów pośrednich (zwanych też „ukrytymi") pomiędzy sensorami (tj. warstwą wejściową) i efektorami (tj. warstwą wyjściową). Niezależnie od tego, czy neurony „ukryte" są z założenia uszeregowane w warstwy czy nie, zawsze można je pogrupować we wspomniane w rozdziale IV.3.4.

„wirtualne warstwy uczenia", wyznaczone dynamicznie w danej epoce dla danego etapu uczącego na podstawie aktualnych (w razie potrzeby aktywizowanych lub dezaktywizowanych) połączeń synaptycznych pomiędzy poszczególnymi sensorami, neuronami i efektorami sieci neuronowej. Ponadto w opisanej metodzie proces uczenia daje się dobrze zorganizować w sensie obliczeniowym, ponieważ wszystkie te elementy sieci można zawsze jednoznacznie uszeregować tak, by było możliwe przeprowadzenie sekwencyjnego ich przetwarzania. Fakt ten umożliwia sprawne przeprowadzenie symulacji sieci (opartnych na metodzie sterowanych kompromisów) na komputerach jednoprocesorowych. Jakkolwiek aktualnie większość sieci neuronowych realizowana jest w postaci programów symulacyjnych na klasycznych komputerach, to przy ocenie właściwości metody sterowanych kompromisów ważne jest również to, że obliczenia dla tej metody można dokonywać również równolegle dla wszystkich elementów danej wirtualnej warstwy uczącej, co pozwala na jej realizację w strukturach systemów współbieżnych.

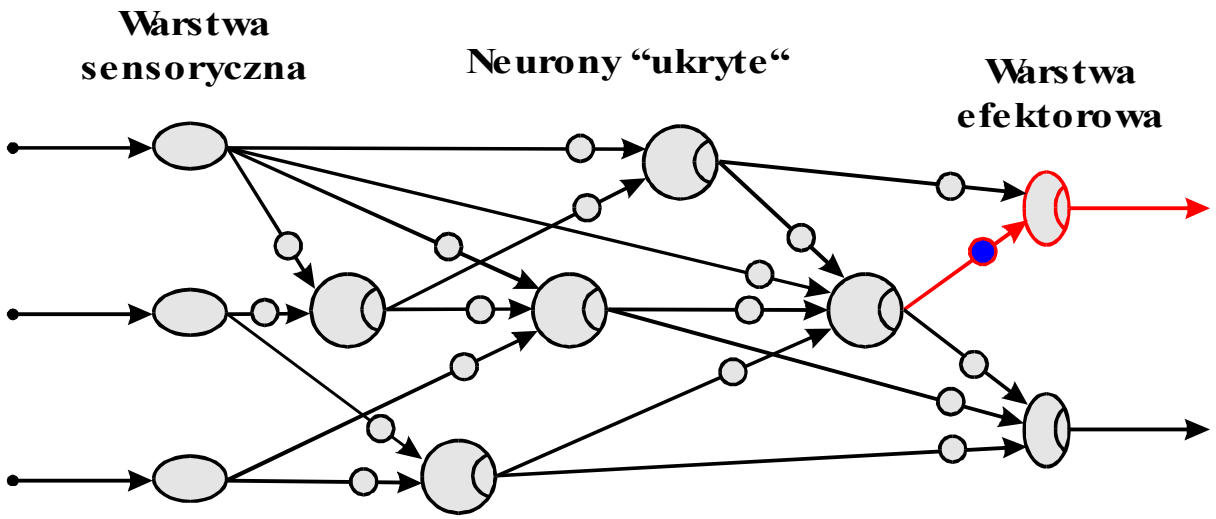
4.4.3.1. Idea uogólnienia metody dla sieci zawierających kilka warstw

Uogólnienie metody sterowanych kompromisów dla potrzeb uczenia sieci neuronowych zawierających neurony (warstwy) pośrednie („ukryte") pociąga za sobą konieczność zdefiniowania tego, jak powstały błąd, który może zostać skonfrontowany z sygnałem uczącym pochodzącym od nauczyciela tylko na wyjściu sieci, ma zostać rozłożony na parametry wolne całej sieci neuronowej. W przypadku zaprezentowanej tu metody uczącej będzie chodzić o określenie tego, jak sygnał uczący ma dotrzeć do neuronów warstw pośrednich, by tam wywołać odpowiednie zmiany, prowadzące w efekcie całą sieć neuronową do wyznaczonego celu.

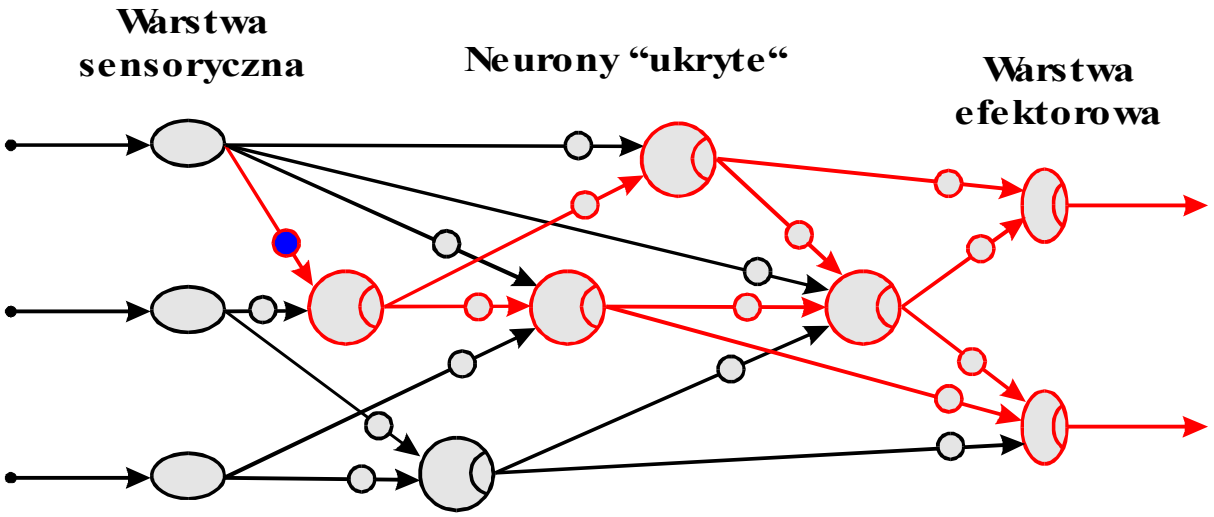
Dla poprawnego opisania algorytmu uczenia sieci zawierających kilka warstw celowe jest przypomnienie sobie, że celem uczenia jest globalna minimalizacja błędu, jaki sieć neuronowa daje na swoim wyjściu, dla wszystkich wzorców ciągu uczącego. Żeby odpowiednio opracować strategię korekt parametrów wolnych sieci, należy najpierw rozważyć, jaki wpływ na działanie sieci będą miały poszczególne korekty. Jeżeli skorygowana zostanie waga lub próg związany z neuronem lub efektorem blisko wyjścia sieci neuronowej (jak to pokazano na rys. IV.15), wtedy wpływ tej korekty na proces przetwarzania informacji w sieci jest niewielki. Jeśli zaś korekcie ulegnie waga lub próg neuronu znajdującego się w pobliżu wejścia sieci (rys. IV.16), wtedy fundamentalnej zmianie ulegnie duża część przetwarzanych w sieci informacji. Wpływ powyżej opisanych zmian na

proces przetwarzania informacji przez sieć można by było porównać odpowiednio do zmian jakie wprowadza w przedsiębiorstwie jakiś szeregowy pracownik niskiego szczebla (np. kucharz - rys. IV.15) i dyrektor (rys. IV.16).

Przenosząc teraz nasze rozważania na grunt sieci neuronowych, możemy powiedzieć, że w razie wystąpienia błędu na wyjściu sieci warto spróbować najpierw skorygować powstały błąd tam, gdzie prawdopodobieństwo zaburzenia całego procesu przetwarzania informacji jest mniejsze, a dopiero kiedy zmiany te okażą się niewystarczające, stopniowo przechodzić do restrukturyzacji całego systemu przetwarzania informacji w sieci.



Rysunek 4.15. Korekta wagi synaptycznej w pobliżu wyjścia sieci ma mały wpływ na proces przetwarzania informacji w całej sieci (analogia zmian jadłospisu przez kucharza w przedsiębiorstwie).



Rysunek 4.16. Korekta wagi synaptycznej w pobliżu wejścia sieci ma mały duży wpływ na proces przetwarzania informacji w całej sieci (analogia zmian strategii firmy na rynku wprowadzonej przez dyrektora w przedsiębiorstwie).

Zauważmy dalej, że dane wejściowe – stanowią zawsze pewien konkret, zaś im dalej w głąb sieci, tym bardziej ogólne (bardziej abstrakcyjne) cechy są wydobywane z tych konkretnych sygnałów, i że sygnał wyjściowy sieci powstaje właśnie w oparciu o ten najbardziej abstrakcyjny opis danego problemu. Znaczy to tyle, że jeśli do wytworzenia odpowiedniego sygnału wyjściowego nie wystarcza sam sygnał wejściowy (jak to ma miejsce w przypadku sieci dwuwarstwowych), to dodaje się kolejne neurony lub całe warstwy neuronów pośrednich, które ten sygnał odpowiednio preparują (wyłuskując pewne podobieństwa i wtórne cechy prezentowanych danych). Czyni się to w tym celu, żeby w nowych przestrzeniach cech wtórnych (obliczanych na bazie sygnału wejściowego) można było łatwiej podjąć końcową decyzję, czyli obliczyć odpowiedni sygnał wyjściowy. Wniosek z tego jest taki, że zastosowanie większej liczby warstw ukrytych teoretycznie powinno umożliwić bardziej płynne przejście od szczegółów do ogółów i dać możliwość hierarchicznego rozwikłania nawet najbardziej skomplikowanych problemów zadanych ciągiem uczącym. Podsumowując te rozważania i przekładając je na „filozofię" sposobu przeprowadzania korekt parametrów wolnych sieci, można powiedzieć, że najpierw należy spróbować skorygować otrzymaną przez sieć abstrakcję, a jeśli to nie da odpowiedniego rezultatu, przechodzić należy w kierunku korekty tych czynników, które wpłynęły na powstanie tej niewłaściwej (bo wymagającej korekty) abstrakcji.

Takie postępowanie znajduje również pewne odzwierciedlenie w sposobie rozumowania człowieka, który często woli najpierw zmienić interpretację (pewną abstrakcję) dostrzeganych faktów oraz sposób ich wykorzystywania w celu osiągnięcia zamierzonego celu, niż gruntownie zmieniać całe swoje postępowanie (dokonując na przykład konkrety w sferze wartości), co wymaga o wiele więcej pracy i wysiłku. Ze względu na stabilność procesu uczenia można taki oportunistyczny sposób postępowania uzasadnić zdrowym pragmatyzmem: po prostu zmiany mniej ingerujące w proces przetwarzania informacji wydają się być również bardziej wskazane niż te, które drastycznie zmieniają większość obliczeń zachodzących w sieci. Oczywiście nie można się kierować wyłącznie oportunizmem, więc fundamentalne zmiany mogą być czasami także konieczne, np. w przypadku osiągnięcia minimum lokalnego lub na początku uczenia sieci. Zwłaszcza na początku procesu uczenia, konieczne jest często sięganie do zmian zasadniczych (w sferze najbardziej podstawowych pojęć i „wartości"), ale pamiętajmy, że dotyczy to sieci „młodej", w której wtedy zazwyczaj występuje zupełny chaos informacyjny, wymagający podstawowego ukierunkowania nauki. Proces nauki sieci powinien być jednak generalnie tak uwarunkowany, aby zmiany toczyły się przede wszystkim na poziomie jak najmniejszej ingerencji w stabilność nauki, zarazem jednak umożliwiając sieci w każdym momencie przeprowadzenie istotnych fundamentalnych zmian, o ile wystąpi taka potrzeba. Zmiany fundamentalne powinny być jednak zawsze poprzedzone próbami osiągnięcia celu tymi poprawkami parametrycznymi, możliwie delikatnymi i mało zakłócającymi dla całego procesu, w celu finalnego osiągnięcia zbieżności sieci do rozwiązania zadanego ciągiem uczącym.

Realizacja wyżej opisanych postulatów w opisywanej w pracy metodzie sterowanych kompromisów odbywa się w następujący sposób:

W każdej epoce uczącej przeprowadzanych jest kilka etapów. Ich ilość zależy od ilości wirtualnych warstw uczenia, zależnych (jak wiadomo) od aktualnej architektury sieci, a w szczególności od sposobu połączeń poszczególnych sensorów, neuronów i efektorów.

Podczas każdego etapu uczącego, w którym przeglądany jest cały ciąg uczący, dokonywana jest propagacja wsteczna pożądanego sygnału Y^p dla ustalonego wzorca uczącego p do aktualnie dostrajanej warstwy uczącej. Dostrajanie warstw uczących przebiega w takiej kolejności, by kolejność ta odzwierciedlała ideę korekty powstałych błędów w kierunku od tych najmniej ingerujących w proces przetwarzania informacji w sieci, do tych najbardziej go modyfikujących.

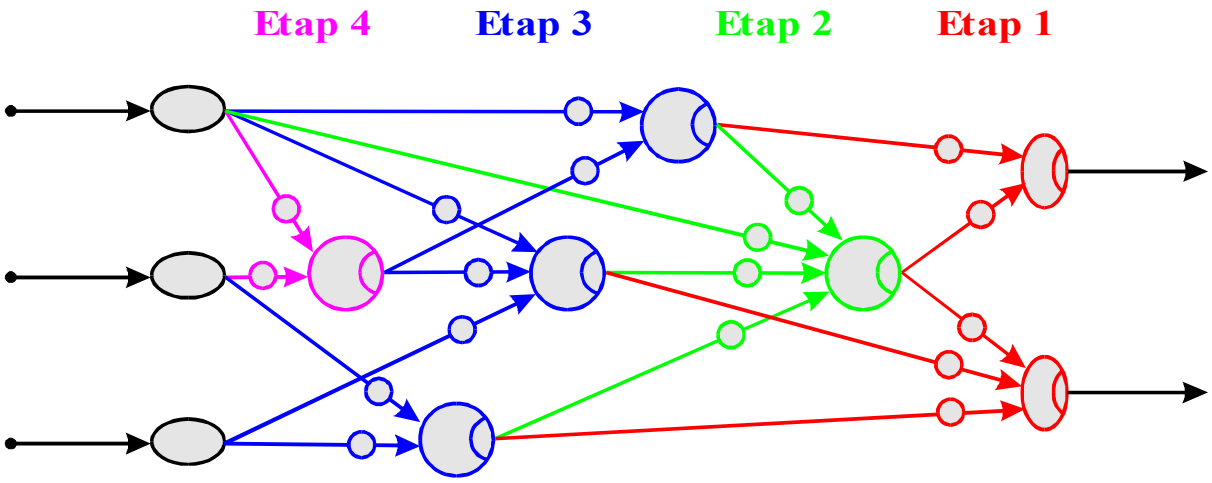
W danym etapie uczącym obliczane są tylko korekty wag i progów neuronów i efektorów warstwy uczącej związanej z rozważanym etapem. Na końcu takiego etapu obliczany jest kompromis i dokonywana jest aktualizacja wag i progów elementów sieci na podstawie zebranych propozycji wszystkich korekt, sugerowanych przez poszczególne wzorce uczące.

Następnie sprawdza się, na ile przeprowadzona aktualizacja wag i progów wpłynęła pozytywnie na globalną minimalizację błędu sieci dla wszystkich wzorców. O ile błąd nadal nie został w pełni skorygowany i istnieje jeszcze jakaś poprzednia warstwa ucząca, rozpoczyna się kolejny etap uczący dla tej warstwy.

Jeśli zaś brak warstwy poprzedniej, epoka ucząca jest kończona i rozpoczyna się następna, w której neurony są znowu etapami korygowane w kierunku od wyjścia do wejścia sieci.

Na rysunku 4.17. zilustrowano proces etapowej korekty i aktualizacji sieci neuronowej. Trzeba zauważyć, że dla sieci neuronowych o „luźnych" topologiach, w których nie każdy z każdym neuronem jest połączony, warstwa sieci neuronowej zaprojektowana przez konstruktora nie musi odpowiadać warstwie uczącej neuronów aktualizowanych w danym etapie. Dla sieci neuronowych o dynamicznie zmiennej architekturze (w przypadku zastosowania metod ontogenicznych) ilość warstw uczących (a więc i etapów uczących) może się dynamicznie zmieniać. Dynamiczne wyznaczanie wirtualnych warstw uczących jest ściśle związane z procesem propagacji wstecznej sygnału uczącego i nie stanowi żadnego dodatkowego obciążenia natury obliczeniowej.

Dla pełności opisu należy dodać, że metoda sterowanych kompromisów teoretycznie pozwala na rozpoczęcie procesu korygowania parametrów sieci od dowolnej wirtualnej warstwy uczenia i kontynuowanie tego procesu w dowolnym porządku, tzn. np. można rozpocząć aktualizację wag i progów sieci od warstwy uczenia usytuowanej zaraz przy wejściu sieci i później kolejno korygować następne warstwy w kierunku do jej wyjścia. Ze względu na szeroki zakres zagadnienia z powodów opisanych w tym rozdziale w tej pracy ograniczono się do rozważania procesu korygowania parametrów sieci w kierunku od wyjścia sieci do jej wejścia dla poszczególnych wirtualnych warstw uczenia.



Rysunek 4.17. Wizualizacja epoki uczącej z zaznaczeniem poszczególnych etapów uczących (a zarazem wirtualnych warstw uczących) dla przykładowej sieci neuronowej.

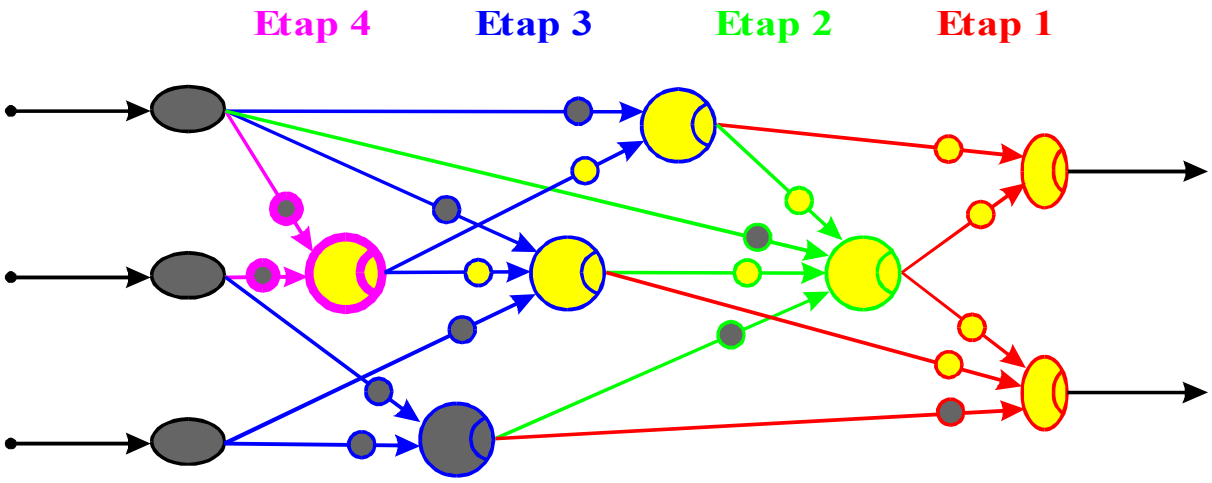
4.4.3.2. Opis algorytmu uogólnionej metody sterowanych kompromisów dla wielu warstw

4.4.3.2.1. Wyznaczanie ścieżek uczenia

Uogólniona wersja metody sterowanych kompromisów dla wielu warstw używa specjalnych znaczników przypisanych do wszystkich neuronów, efektorów i synaps sieci neuronowej. Znaczniki te służą do wyznaczenia ścieżek, na których możliwa jest wsteczna propagacja sygnału uczącego dla danego etapu uczącego. Znaczniki te są automatycznie propagowane w kierunku od wejścia sieci neuronowej do jej wyjścia w fazie propagacji sygnału pobudzenia sieciowego. Dzięki temu nawet przy zastosowaniu dynamicznie zmiennej architektury sieci neuronowej, możliwe jest każdorazowo dynamiczne wyznaczanie ścieżek uczących i prawidłowa propagacja wsteczna sygnału uczącego, zgodnie z założeniami opisanej metody. Używane są dwa rodzaje znaczników:

- **znaczniki korekty** (przypisane wszystkim neuronom i efektorom),
- **znaczniki ścieżki korekty** (przypisane wszystkim neuronom, efektorom i synapsom).

Znaczniki korekty niosą informację o tym, czy dany neuron lub efektor został już w którymś z poprzednich etapów danej epoki skorygowany i zaktualizowany, czy nie. Przez skorygowanie i aktualizację danego neuronu lub efektora rozumie się korektę wszystkich jego wag i progu. W danej epoce uczącej każdy neuron i efektor jest korygowany tylko raz. Oznacza to tyle, że jeżeli dany neuron (efektor) zostanie skorygowany w danym etapie uczącym, dalszej korekcie nie podlegają już jego wagi ani próg, lecz sygnał uczący jest w następnym etapie uczącym propagowany do poprzedniej warstwy uczącej, a więc do neuronów, które nie zostały jeszcze w danej epoce uczącej skorygowane. Rozkład sygnału uczącego na poszczególne dendryty odbywa się na podstawie **znaczników ścieżek korekty** przypisanych do poszczególnych synaps. Znaczniki te informują dany neuron lub efektor o tym, gdzie można jeszcze propagować sygnał uczący w celu korekty pozostałego jeszcze błędu sieciowego. Epoka ucząca trwa tak długo, dopóki wskaźniki korekty wszystkich biorących udział w nauce neuronów i efektorów nie przyjmą wartości określającej, iż związane z nimi elementy sieci zostały już skorygowane. Rysunek 4.18. uwidacznia opisaną wyżej strategię wstecznej propagacji sygnału uczącego dla jednej z epok uczących z uwzględnieniem znaczników uczących.



Rysunek 4.18. Strategia wstecznej propagacji sygnału uczącego w 4. etapie uczącym z uwidocznieniem opisanych w tekście znaczników: kolorem żółtym oznaczone są wskaźniki otwartej ścieżki korekty, zaś kolorem szarym wskaźniki zamkniętej ścieżki korekty. Sygnał uczący jest rozkładany na te elementy sieci neuronowej, dla których wskaźniki ścieżki korekty są otwarte. Pogrubione zostały te elementy sieci, które w 4. etapie uczącym są korygowane.

Neurony i efektory podczas fazy propagacji wstecznej sygnału uczącego zachowują się odmiennie w zależności od tego, w jakim stanie uczącym danej epoki się znajdują. Sensory wcale nie biorą udziału w nauce. Rozróżniamy następujące stany przypisywane dynamicznie elementom sieci w trakcie uczenia:

- **nie korygowany** – nie biorący udział w danym etapie uczącym,
- **aktualizowany** – aktualizowany zgodnie z wzorami opisanymi w rozdziale 4.3.2.-3.,
- **skorygowany** – biorący udział w nauce w roli pośredników (gdy ich wskaźnik ścieżki korekty jest otwarty) czyli propagujących odpowiednio rozłożony sygnał uczący poprzez te swoje dendryty, których ścieżka korekty jest otwarta.

Do neuronów **nie korygowanych** nie dochodzi pełna informacja umożliwiająca ich korektę. Neurony te więc nie są aktualizowane w danym etapie uczącym, lecz czekają na swoją kolejność w którymś z następnych etapów uczących danej epoki uczącej. Neurony **aktualizowane**, do których w danym etapie dotarła pełna informacja umożliwiająca ich korektę, tworzą wyżej wspomnianą wirtualną warstwę uczącą i podlegają korektom i aktualizacji. Neurony **skorygowane** w którymś z poprzednich etapów uczących danej epoki, których wskaźnik ścieżki jest nadal otwarty, nie podlegają już następnym korektom w danej

epoce uczącej, lecz przesyłają – w sposób opisany poniżej – odpowiednio rozłożony na poszczególne swoje dendryty (o statusie otwartej ścieżki korekty) sygnał uczący, umożliwiając w ten sposób neuronom z wcześniejszych warstw korektę tego, czego im nie udało się skorygować.

4.4.3.2.2. Wsteczna propagacja sygnału uczącego dla danego etapu uczącego sieci wielowarstwowych

Każda epoka ucząca składa się z jednego lub kilku etapów uczących w zależności od stopnia skomplikowania architektury sieci neuronowej. Gdy w sieci występuje więcej niż jeden etap uczący, konieczna jest wsteczna propagacja sygnału uczącego przez sieć do wcześniejszych warstw uczących. Sygnał uczący musi być wtedy wstecznie przekształcany i rozkładany na poszczególne dendryty efektorów i tych neuronów, które już były wcześniej skorygowane. Proces przekazywania informacji o sygnale uczącym wygląda następująco:

Najpierw obliczana jest wartość stanu pobudzenia efektora S^p na podstawie wartości sygnału uczącego Y^p dla ustalonego wzorca uczącego p analogicznie jak dla efektorów opisanych w rozdziałach 4.3.2. i 4.4.2.:

$$S^p = G_k\left(Y^p\right) \qquad \text{lub} \qquad S^p = G_k\left(Y^p, s^p\right).$$

Następnie obliczana jest korekta tego stanu:

$$\sigma_s^p = S^p - s^p.$$

oraz współczynnik $\hat{M}_s^p$ jego rozkładu na poszczególne dendryty tak, jak zostało to opisane w rozdz. 4.3.2. z tą różnicą, że przy jego obliczaniu brane są pod uwagę tylko te dendryty, których znaczniki ścieżki korekty są otwarte:

$$\hat{M}_s^p = \sum_{i\in\{1,\ldots N:\text{ścieżka}(i)=\text{otwarta}\}} \left|d_i^p\right|.$$

W dalszej kolejności obliczane są korekty postsynaptycznych pobudzeń dendrytycznych dla poszczególnych dendrytów charakteryzujących się otwartą ścieżką korekty następująco:

$$\sigma_{d_i}^p = \sigma_s^p \cdot \frac{\left|d_i^p\right|}{\hat{M}_s^p},$$

W końcu zamiast obliczać nową proponowaną wagę dla ustalonego wzorca uczącego p obliczana jest nowa proponowana dla tego wzorca wartość presynaptycznego pobudzenia synapsy w następujący sposób:

$$X_i^p = \frac{d_i^p + \sigma_{d_i}^p}{W_i^p}.$$

Na podstawie tej wartości obliczana jest korekta presynaptycznego pobudzenia rozważanej synapsy dla ustalonego wzorca uczącego *p*:

$$\sigma_{X_i}^p = X_i^p - x_i^p.$$

Obliczone w ten sposób korekty dla poszczególnych wzorców ciągu uczącego służą następnie do obliczenia wynikowej kompromisowej korekty sygnału presynaptycznego pobudzenia synapsy i zarazem korekty aksonalnego sygnału uczącego Y^p neuronu presynaptycznego dla rozważanej synapsy. Liczymy się przy tym z faktem, że drzewko aksonalne tego neuronu może być rozbudowane, tzn. jego wyjście może być połączone z wieloma różnymi neuronami i efektorami poprzez połączenia synaptyczne. Trzeba więc w procesie uczenia uwzględniać korekty docierające do neuronu presynaptycznego różnymi drogami. W najprostszej wersji metody można się tu ograniczyć do liczenia średniej arytmetycznej[4] z sugerowanych propozycji korekt dla poszczególnych synaps. Odpowiednia formuła korekty sygnału wyjściowego neuronu presynaptycznego wygląda wtedy następująco:

$$\sigma_Y^p = \frac{\sum_{m=1}^{M} \sigma_{X_m}^p}{M},$$

a stąd już wyznaczana jest wartość sygnału uczącego neuronu presynaptycznego dla ustalonego wzorca uczącego *p*:

$$Y^p = y^p + \sigma_Y^p.$$

W ten sposób w każdym etapie uczącym dochodzi do korekty pewnej części błędu, jakim obarczona jest sieć neuronowa. Korekta przeprowadzana jest etapami, od najwyższego poziomu abstrakcji do najniższego. Poszczególne etapy się wzajemnie dopełniają na zasadzie: Czego nie uda się skorygować w jednym etapie, próbuje się korygować w następnym i tak aż do skutku. Proces uczenia przeprowadzany jest hierarchicznie, co jest zgodne z intuicją i z

[4] Średnia arytmetyczna w ogólności niezbyt dobrze nadaje się do obliczania kompromisowych korekt i może być powodem występowania różnych problemów zbieżności nauki opisanych w następnym rozdziale. Obliczanie kompromisowej korekty może być jakkolwiek oparte na tej średniej, ale powinno być przeprowadzane bardziej inteligentnie ze względu na ominięcie mogących się pojawić trudności natury numerycznej.

doświadczeniami związanymi z nauką. Zawsze prościej dostosować szczegóły niż zasadniczo zmieniać od podstaw cały system postępowania.

Taka hierarchia nauki idzie również w parze z odległością sygnału uczącego od poszczególnych warstw uczących, zdefiniowanych w tej pracy. Im warstwa znajduje się dalej od wyjścia sieci neuronowej, tym bardziej jej stan jest chroniony przed zmianami poprzez wcześniejsze korekty warstw znajdujących się bliżej wyjścia sieci neuronowej. Takie postępowanie powinno zapewnić stabilność procesu uczenia sieci neuronowej w odniesieniu do hierarchii wpływu zmian poszczególnych parametrów sieci na proces przetwarzania informacji w całej sieci neuronowej.

4.4.3.2.3. Problemy związane z uczeniem sieci wielowarstwowych

Uczenie wielowarstwowych sieci neuronowych narażone jest na wiele trudności. Do podstawowych należą:

- **Problem minimów lokalnych** i zapewnienia zbieżności metody do minimum globalnego.
- **Problem wstępnej inicjacji sieci**, która w pewien sposób determinuje kierunek poszukiwań minimum globalnego.
- **Problem małych i dużych liczb** związany z obliczaniem kompromisu na bazie wyliczania wartości średniej.

Problem minimów lokalnych może zostać rozwiązany dzięki dobrodziejstwu opisanych w tej pracy odchyleń umożliwiającym przeprowadzenie wnioskowania i ukierunkowania procesu uczenia niezależnie od gradientu funkcji błędu sieci. Dzięki temu można zmierzać w kierunku minimum globalnego i zapewnić odpowiednią zbieżność nauki na bazie opisanych odchyleń, które są równoważne globalnej minimalizacji błędu sieci.

Wstępna inicjacja sieci neuronowej decyduje o sposobie propagowania pobudzenia przez sieć na początku procesu nauki i może w niekorzystny sposób utrudnić znalezienie minimum globalnego. Z metod gradientowych opartych na bazie metod optymalizacyjnych wynika, że problem ten jest niebanalny. Potrzeba przeprowadzenia wielu prób i konieczność stosowania różnych technik omijania minimów lokalnych może zniechęcić do stosowania sieci neuronowych nie jednego badacza. W przypadku metody sterowanych kompromisów problem wstępnej inicjacji sieci można sprowadzić do wcześniej opisanego problemu omijania minimów lokalnych i rozwiązać go znowu dzięki dobrodziejstwu płynącemu z

informacji o odchyleniach. W praktyce wygląda to tak, że dla sieci niekorzystnie zainicjowanej, proces uczący będzie wymuszał w trakcie etapu uczącego dogłębne korekty we wszystkich warstwach uczących sieci próbując zminimalizować opisane odchylenia.

Problem małych i dużych liczb jest specyficznym problemem związanym z zaprezentowaną w tej pracy metodą i nie występuje tak dogłębnie w przypadku innych metod np. gradientowych. Ze względu na to że metoda sterowanych kompromisów próbuje powstały dla poszczególnych wzorców uczących błąd poprawić od razu, zmiany te mogą mieć bardzo radykalny charakter. Proponowane korekty dla poszczególnych wzorców stają się podstawą obliczanego kompromisu, a ten powinien w pewien sposób odzwierciedlać propozycje stawiane przez wszystkie wzorce uczące. Dla rozwiązania tego problemu intuicyjnie nasuwa się rozwiązanie poprzez średnią (np. arytmetyczną), która z natury uwzględnia wszystkie wzorce. Jednak proponowane korekty mogą być jak dodatnie tak ujemne, mogą się wzajemnie znosić – co powoduje stagnację wynikowych kompromisowych korekt i w praktyce prowadzi do minimów zazwyczaj lokalnych. Żeby tego nie było mało, taka wynikowa kompromisowa korekta może spowodować, że nowa obliczona wartość na jej podstawie akurat trafiła w wartość bliską zeru (**małą liczbę**), która w dalszych obliczeniach tej metody jest często wykorzystywana w roli dzielnika ilorazu powodującego powstanie **dużej liczby**. Takie małe i duże liczby zaś w następnej epoce uczącej bardzo destabilizują proces nauki i powodują niekorzystne fluktuacje. Należy podkreślić, że zazwyczaj obliczenie takich małych liczb nie było proponowane przez żaden wzorzec uczący, lecz było wynikiem obliczania średniej. Z tego powodu należy szukać bardziej inteligentnych metod obliczania kompromisu, które nie dopuszczają do powstawania takich niechcianych anomalii.

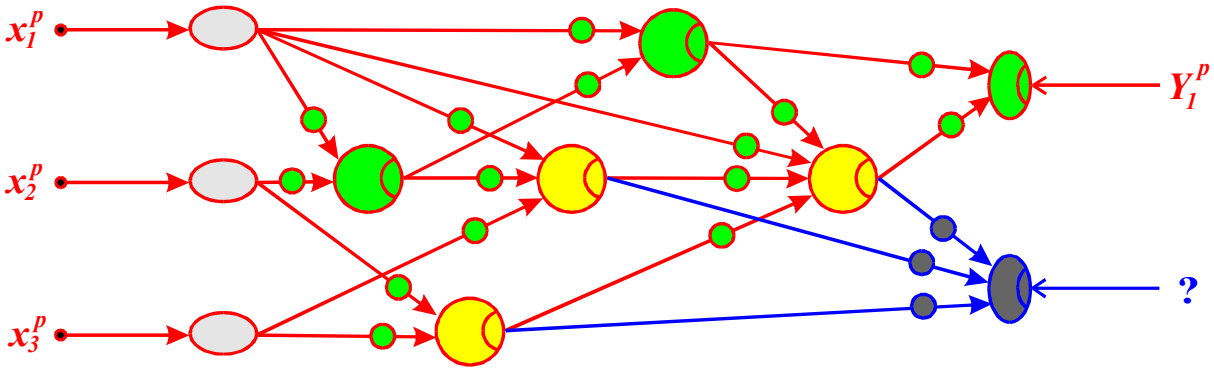
4.4.4. Uczenie niekompletnych wzorców uczących

Metoda sterowanych kompromisów pozwala również wykorzystać niekompletne wzorce uczące do skutecznej nauki sieci neuronowej. Oznacza to, że jeśli w wyniku jakiegoś doświadczenia (dla wybranego zbioru parametrów charakteryzujących go) uzyskamy pewne wyniki, które mają stać się zbiorem uczącym, ale są one niekompletne i z ważnych powodów (na przykład natury finansowej lub deontologicznej) uzyskanie kompletnych danych jest niemożliwe bądź trudne do osiągnięcia, to w metodzie sterowanych kompromisów nadal możemy ten zbiór danych wykorzystać jako zbiór uczący. Sieć neuronowa będzie wtedy w procesie nauki zdobywać mniej wiedzy, gdyż będzie dostosowywana do tego niekompletnego zbioru danych, ale dane takie nie będą dla procesu nauki całkowicie stracone. W wyniku

przeprowadzonej nauki uzyskane zostanie pewne uogólnienie, które umożliwi nie tylko klasyczne zadawanie zapytań z poza zbioru uczącego, ale również zadawanie zapytań dla niekompletnych wzorców. W wyniku takiego zapytywania sieci istnieje możliwość, że sieć na podstawie uzyskanego uogólnienia dopełni brakujące dane wyjściowe i w ten sposób da odpowiedź na to, na co nie udało się uzyskać pełnej odpowiedzi w trakcie przeprowadzanego doświadczenia.

Realizacja wyżej opisanego postulatu wygląda następująco:

W przypadku, gdy dla danego wzorca uczącego dla określonych danych wejściowych brak kompletnego sygnału uczącego, wtedy cały proces przebiega analogicznie jak dla wzorców kompletnych z tą różnicą, że brakująca część sygnału uczącego nie jest wstecznie propagowana przez sieć. W zależności od stopnia niekompletności wzorca i od architektury sieci neuronowej niektóre wagi i progi sieci dla rozważanego wzorca mogą nie być w związku z tym korygowane. W przypadku wielowarstwowej architektury sieci podczas obliczania kompromisowej korekty sygnału aksonalnego neuronu presynaptycznego brane są pod uwagę tylko te odgałęzienia (kolaterale) drzewka aksonalnego, dla których możliwe jest obliczenie stosownej korekty. Gdy dane odgałęzienie drzewka aksonalnego nie jest w stanie przesłać neuronowi presynaptycznemu proponowanej korekty jego pobudzenia aksonalnego, wtedy informuje o tym ten neuron, żeby ten mógł zrealizować swoją korektę na podstawie tej części informacji, która do niego dotarła od innych kolaterali. Na rysunku 4.19. uwidoczniona jest sytuacja opisana powyżej.



Rysunek 4.19. Uwidocznienie postulatu korekty na podstawie nie kompletnego sygnału uczącego. Czerwonym kolorem oznaczony został znany sygnał uczący dla ustalonego wzorca uczącego *p* pochodzący od nauczyciela Y_1^p oraz te, które zostały wstecznie propagowane przez sieć neuronową na jego podstawie, zaś niebieskim, te które dla danego wzorca uczącego *p* nie są znane. Synapsy oznaczone kolorem niebieskim przekazują (w przynależnym im etapie uczącym) neuronom wypełnionym kolorem żółtym tylko informację o tym, że nie są im zdolne przesłać propozycji korekty, żeby te nie czekały na nią myśląc, że nie nadeszła jeszcze pora ich korekty w danym etapie uczącym. Neurony wypełnione kolorem żółtym są więc korygowane na podstawie propozycji korekt pozostałych (oznaczonych kolorem czerwonym) kolaterali drzewka aksonalnego. Neurony, efektory i synapsy wypełnione kolorem zielonym są korygowane w zwykły sposób określony we wcześniejszych rozdziałach (rozdz. 4.3.2. i rozdz. 4.4.3.2.2.). Efektor i wagi wypełnione kolorem szarym nie są korygowane dla rozważanego niekompletnego wzorca uczącego *p*.

4.5. Podsumowanie i kierunki rozwoju metody

Metoda sterowanych kompromisów opisana w tej pracy stanowi całkowicie nowe podejście do problemu szukania minimum globalnego funkcji błędu. Proces minimalizacji funkcji błędu udało się przy tym znacząco uprościć poprzez jego przeniesienie z konieczności przeszukiwania nieskończonej dziedziny tej funkcji i nieznanej jej charakterystyki, na problem minimalizacji skończonej ilości odchyleń od uzyskanego kompromisu, zdefiniowanych w tej pracy. Metoda ta wprowadza też pojęcie lokalności w odniesieniu do obliczania błędu i jego korekty dla poszczególnych elementów sieci neuronowej.

Następnym atutem tej metody jest stosunkowo szeroki zakres możliwości wyboru funkcji aktywacji neuronów i efektorów. Opisana metoda umożliwia używania takich funkcji aktywacji, których zbiór wartości jest nieograniczony. To z kolei eliminuje konieczność

sztucznego skalowania, a przede wszystkim umożliwia prawidłową ekstrapolację sieciom zbudowanym na bazie takich neuronów i efektorów. Zaprezentowana metoda nie ogranicza się jedynie do funkcji monotonicznych, ale daje możliwość stosowania różnorodnych funkcji aktywacji, z periodycznymi włącznie. Zastosowanie tak nietypowych funkcji aktywacji otwiera przed sieciami neuronowymi drogę do rozwiązania bardziej skomplikowanych zadań i uzyskania lepszej sprawności obliczeniowej dla problemów, których natura może być w łatwiejszy sposób określona właśnie dzięki takim funkcjom. Dodatkowo istnieje możliwość prawie dowolnej kombinacji różnych funkcji aktywacji w jednej sieci neuronowej.

Możliwości uogólniania opisanej metody są duże. W tej pracy zaprezentowano podstawową ideę metody sterowanych kompromisów, jak również wprowadzono pewne jej rozszerzenia. Nie wszystkie problemy związane z tą metodą zostały jednak jeszcze finalnie rozwiązane. Część problemów rozwiązano a część tylko zasygnalizowano w celu określenia dalszych możliwych kierunków rozwoju zaprezentowanej metody. Głównymi problemami, jakie pozostają otwarte, są: konieczność bardziej inteligentnego wyznaczania kompromisu oraz opracowanie odpowiedniej strategii postępowania na podstawie obliczonych odchyleń od wyznaczonego kompromisu. Rozwiązanie tych dwóch problemów umożliwi zaproponowanej tu metodzie stać się pełnowartościowym narzędziem rozwiązywania nawet bardzo skomplikowanych zadań. Można oczekiwać, że po pokonaniu wskazanych ograniczeń metoda sterowanych kompromisów będzie mogła zostać znacząco spopularyzowana i stanowić będzie dużą konkurencję dla istniejących metod uczenia sieci neuronowych.

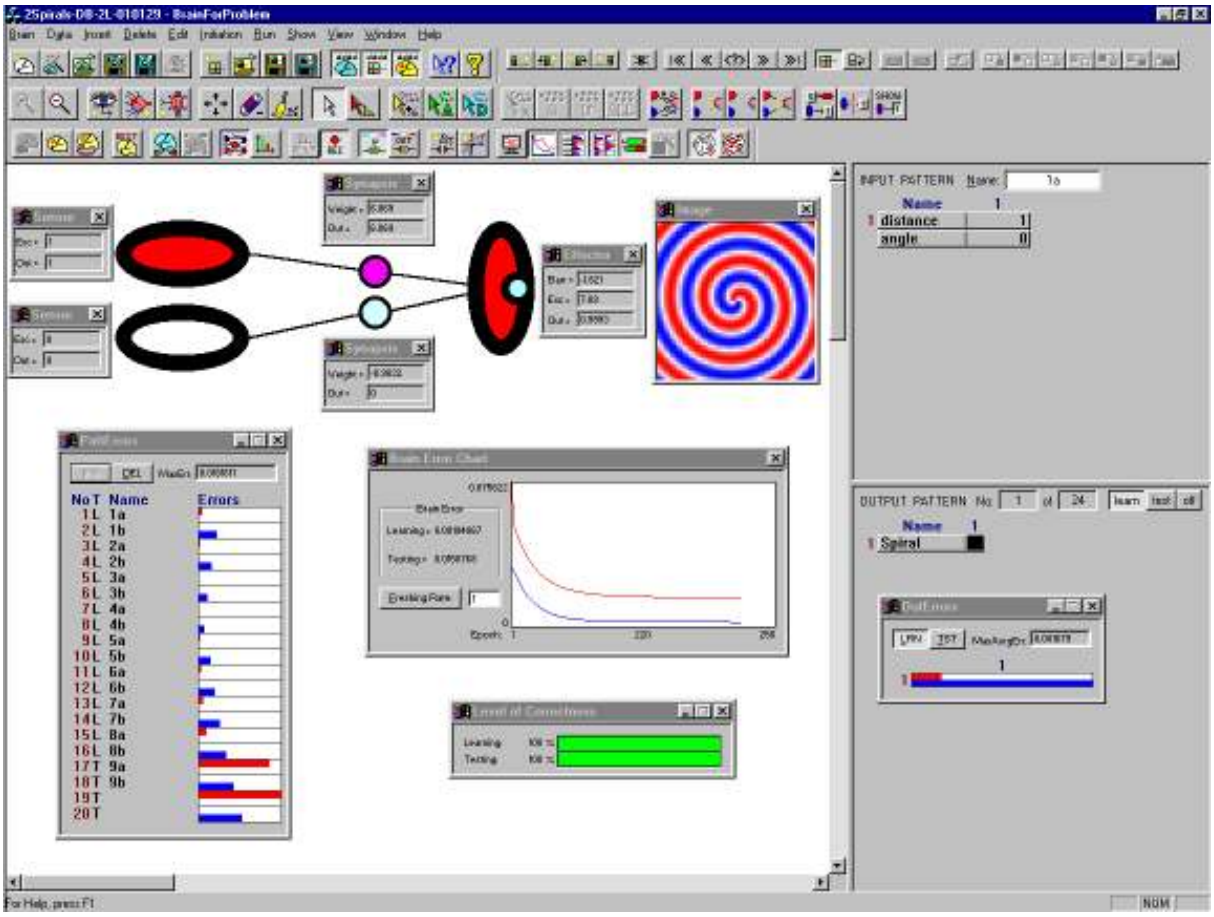
Metoda ta jest cały czas intensywnie rozwijana, jednak ze względu na konieczność przeprowadzania licznych badań porównawczych, symulacji, wyprowadzania i weryfikowania różnych formuł matematycznych, osiągnięcie następnych wartościowych wyników jest kwestią czasu.

4.6. Aplikacja „Brain for Problem”

Dla zilustrowania funkcjonowania zaproponowanej w tej pracy metody uczenia, a także dla praktycznego zbadania jej właściwości, została przez autora pracy opracowana i oprogramowana aplikacja nazwana „**Brain for Problem**” (Rysunek 4.19.). Aplikacja ta została zaprojektowana z myślą o przetestowaniu zaproponowanych wyżej algorytmów uczących sieci neuronowe. Program „***Brain for Problem***” jest załączony do niniejszej pracy, co pozwala Czytelnikowi ocenić w praktyce opracowaną metodę sterowanych kompromisów. Z myślą o użytkownikach, którzy chcieli by poznać i samodzielnie zbadać zaproponowaną

metodę uczenia zbudowana przez autora aplikacja została wyposażona w szereg udogodnień, stwarzających możliwości różnych badań. Do najważniejszych udogodnień wbudowanych w stworzony dla tej pracy program można zaliczyć:

- edytor ciągów uczących (graficzny i liczbowy) z możliwością zapisu na dysk,
- graficzny edytor sieci neuronowych z możliwością zapisu na dysk,
- kreator ułatwiający tworzenie podstawowych architektur sieci neuronowych dla zadanego ciągu uczącego,
- wyświetlanie wielu okien diagnostycznych umożliwiających podgląd i modyfikację parametrów sieci neuronowej oraz procesu uczenia,
- dynamiczny podgląd nauki sieci neuronowej w specjalnie zaprojektowanym interfejsie graficznym,
- w pełni kontrolowane i obserwowane uczenie zaprojektowanych sieci neuronowych metodą sterowanych kompromisów.



Rysunek 4.19. Widok głównego okna aplikacji „Brain for Problem” wraz z okienkami umożliwiającymi dynamiczną obserwację parametrów sieci neuronowej podczas jej uczenia.

4.6.1. Opis podstawowych funkcji zbudowanej aplikacji

Aplikacja „Brain for Problem” może być sterowana poprzez przejrzyste menu (Rysunek 4.20.), które udostępnia użytkownikowi wszystkie możliwe opcje, ustawienia i funkcje zaimplementowane w programie. Menu podzielone jest na kilka podstawowych kategorii przedstawionych na rysunku 4.21.

Rysunek 4.21. Menu aplikacji „Brain for Problem”.

Omówimy je teraz kolejno:

Brain – umożliwia dokonywanie operacji zapisu lub odczytu z dysku sieci neuronowej oraz daje możliwość wykorzystania *kreatora* ułatwiającego tworzenie podstawowej architektury sieci neuronowej dla zadanego ciągu uczącego.

Data – odnosi się do operacji zapisu lub odczytu z dysku ciągu uczącego,

Insert – opcja ta umożliwia dodawanie do sieci neuronowej sensorów, efektorów, neuronów, synaps, połączeń z ciągiem uczącym oraz dodawanie nowych wzorców do ciągu uczącego,

Delete – w tej opcji mamy do czynienia z czynnościami odwrotnymi do operacji znajdujących się w opcji **Insert**. Przy użyciu tej opcji można obsługiwać też funkcję optymalizacji architektury sieci neuronowej (**Clean**) polegającą na usunięciu z niej wszystkich funkcjonalnie niepotrzebnych składników,

Edit – opcja ta związana jest z funkcjami edytorskimi odnoszącymi się zarówno do sieci neuronowej, jak i do ciągu uczącego,

Initiation –opcja ta pozwala zainicjować wszystkie, bądź wybrane parametry wolne sieci neuronowej (wagi, progi, parametry stromości funkcji aktywacji, funkcje aktywacji neuronów i efektorów). Inicjacja może być dokonywana liczbami pseudolosowymi z podanych zakresów liczbowych. Dodatkowo można wydzielić pewną grupę podlegających inicjacji elementów sieci neuronowej,

Run – daje możliwość uczenia sieci neuronowej metodą sterowanych kompromisów, kontynuacji jej nauki, testowania oraz zapamiętywania jej najlepszego stanu uzyskanego w trakcie procesu uczenia,

Show – udostępnia pewne opcje związane z wyglądem sieci neuronowej oraz daje dostęp do poszczególnych wzorców ciągu uczącego, oferując różne możliwości wyszukiwawcze,

View – zarządza widokami pasków narzędziowych, paska stanu oraz wielkością widoku sieci neuronowej i rozmiarem widoku ciągu uczącego,

Window – umożliwia otwarcie okien dynamicznego podglądu wielkości i kształtowania się błędów w trakcie procesu uczenia. Można obserwować zmiany błędów: sieci neuronowej, poszczególnych wzorców, poszczególnych wyjść sieci oraz graficzną interpretację wyników,

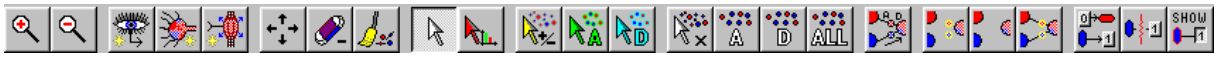
Help – służy pomocą oraz daje informacje o programie.

Bardzo przydatne przy edycji sieci neuronowej i ciągu uczącego, jak również w trakcie prowadzenia doświadczeń, są paski narzędziowe, które dają natychmiastowy dostęp – bez ciągłego przeglądania menu – do prawie wszystkich funkcji i opcji zbudowanej aplikacji. Przyciski zostały pogrupowane tworząc następujące paski narzędziowe:

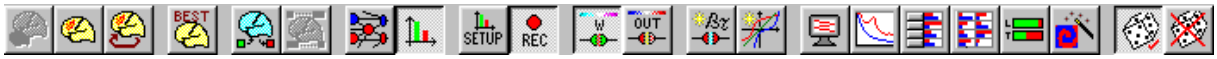
- pasek operacji dyskowych (Rysunek 4.22.),
- pasek edycji ciągu uczącego (Rysunek 4.23.),
- pasek edycji sieci neuronowej (Rysunek 4.24.),
- pasek nauki i podglądu jej przebiegu (Rysunek 4.25.).

Rysunek 4.22. Pasek narzędzi operacji dyskowych, kreowania nowych sieci neuronowych i ciągów uczących.

Rysunek 4.23. Pasek narzędzi służący do przeglądania i modyfikacji ciągu uczącego.



Rysunek 4.24. Pasek narzędzi ułatwiający edycję sieci neuronowych.



Rysunek 4.25. Pasek narzędzi związany z inicjacją, nauką i dynamicznym podglądem sieci neuronowej.

Wszystkie przyciski opatrzone są „dymkami" z opisem ich funkcji. Opisy te pojawiają się po umieszczeniu wskaźnika myszki nad nimi, a na pasku stanu programu pojawia się przy tym bardziej treściwy opis funkcji zaznaczonego przycisku.

4.6.2. Zastosowana symbolika

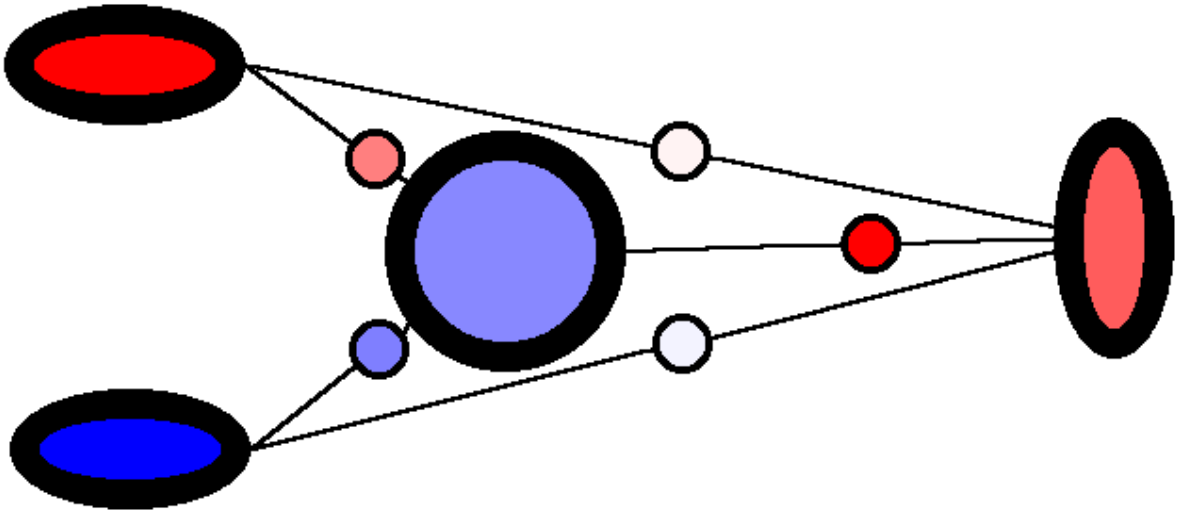
Rysunki 4.26. – 4.31. przedstawiają zastosowaną w aplikacji „Brain for Problem" symbolikę graficzną, odnoszącą się do poszczególnych elementów sieci neuronowej.

Rysunek 4.26. Symbol sensora (*input neuron*). Kolor czerwony oznacza stan dodatniego pobudzenia sensora.

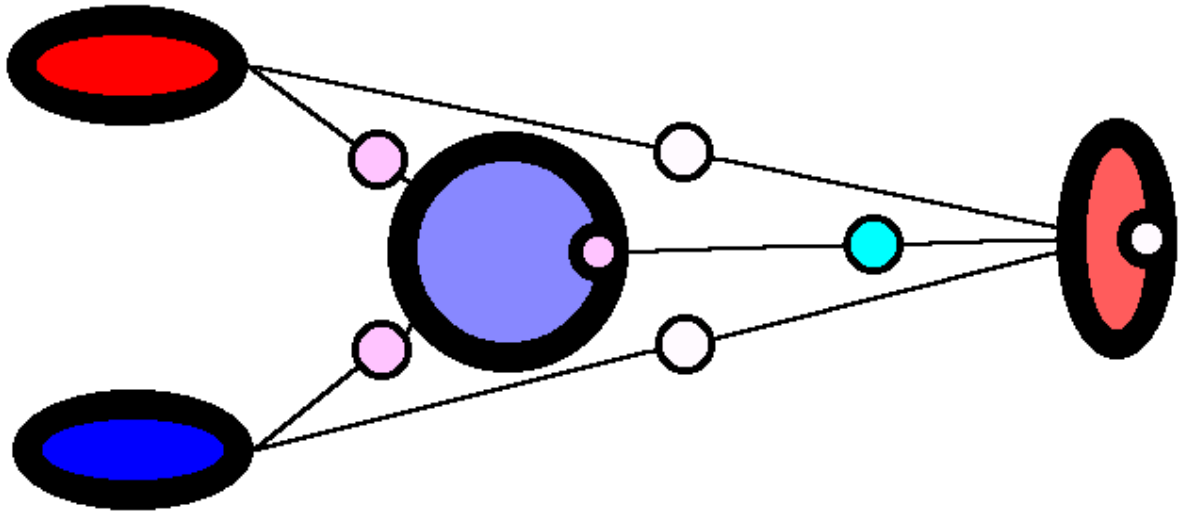
Rysunek 4.27. Symbol wewnętrznego neuronu sieci (*hidden neuron*) z zaznaczonym progiem. Kolor niebieski oznacza stan ujemnego pobudzenia neuronu (hamowanie). Kolor purpurowy wypełniający wnętrze małego wewnętrznego kółka odnosi się do progu neuronu i oznacza jego dodatnią wartość. Jeżeli wnętrze progu byłoby wypełnione kolorem błękitnym, oznaczałoby to jego wartość ujemną.

Rysunek 4.28. Symbol efektora (*output neuron*) z progiem. Kolor czerwony oznacza stan dodatniego pobudzenia efektora. Kolor biały wypełniający wnętrze progu oznacza, że jego wartość jest bliska zeru.

Rysunek 4.29. Symbole synapsy. Kolor błękitny oznacza wartość ujemną wagi, zaś kolor purpurowy jej wartość dodatnią. Nasycenie kolorem mówi o wielkości danej wagi względem pozostałych wag i progów sieci neuronowej. Kolor czerwony wyraża pobudzenie elementu postsynaptycznego, zaś kolor niebieski jego hamowanie. Nasycenie kolorem mówi o wielkości pobudzenia bądź hamowania względem pozostałych pobudzeń i hamowań sieci neuronowej.



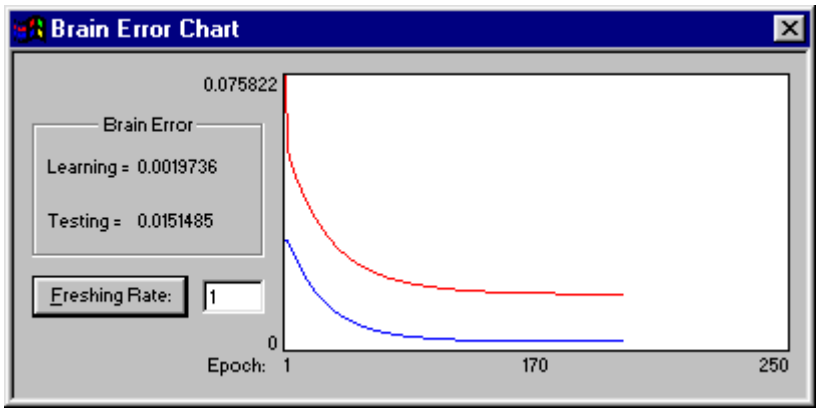
Rysunek 4.30. Prosta sieć neuronowa składająca się z dwóch sensorów (z lewej), jednego efektora (z prawej), neuronu „ukrytego” (w środku) oraz pięciu połączeń synaptycznych. Kolorystyka symbolizuje stan pobudzenia poszczególnych elementów: kolor czerwony – dodatni stan pobudzenia, kolor niebieski – ujemny stan pobudzenia (hamowanie). Intensywność koloru wyraża stopień pobudzenia danego elementu.



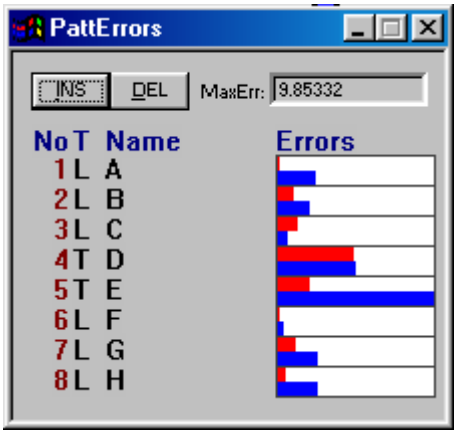
Rysunek 4.31. Sieć neuronowa z rysunku 4.30. z widokiem wartości wag i progów zamiast widoku ich pobudzenia bądź hamowania postsynaptycznego.

4.6.3. Opis okienek umożliwiających dynamiczny podgląd procesu uczenia

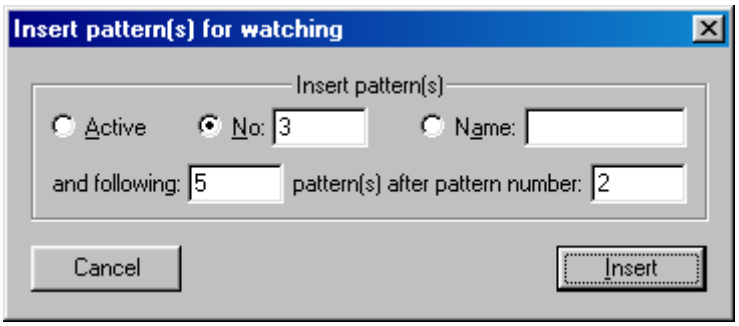
Aplikacja „Brain for Problem” umożliwia dynamiczne wyświetlanie wielu użytecznych parametrów w trakcie procesu uczenia sieci neuronowej. Daje to możliwość obserwowania zmian zachodzących w sieci neuronowej pod wpływem zastosowanego algorytmu uczącego i wyciągania stosownych wniosków, które mogą następnie posłużyć przeprowadzeniu rewizji i usprawnienia algorytmu uczącego. Wybranie graficznej formy prezentacji wyników spowodowane jest chęcią uwolnienia się od ograniczeń percepcyjnych człowieka, które uniemożliwiają mu śledzenie dużej ilości danych w formie liczbowej. Następujące rysunki (4.32. – 4.51.) prezentują najważniejsze i najczęściej używane okienka aplikacji wraz z ich opisem.



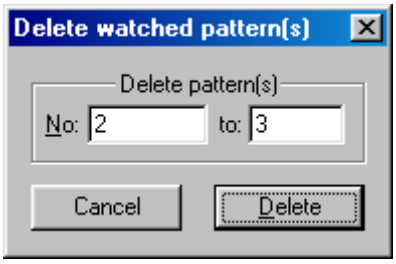
Rysunek 4.32. Okienko wykresów średniego błędu sieci neuronowej dla całego ciągu uczącego. Niebieskim kolorem oznaczony jest błąd sieci dla wzorców uczących, zaś czerwonym błąd sieci dla wzorców testujących**.** W środku pod wykresem znajduje się licznik podający numer aktualnie wykonanej epoki uczącej (tutaj 170 epoka).



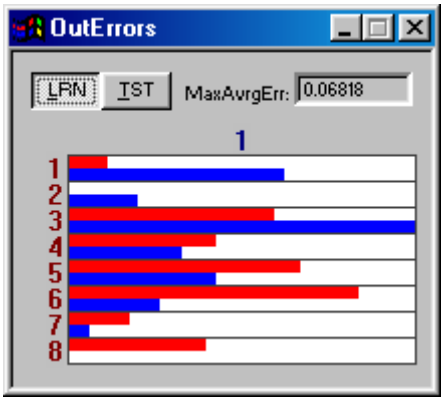
Rysunek 4.33. Okienko porównawcze błędów dla poszczególnych wzorców uczących, z wyszczególnieniem wartości liczbowej maksymalnego z wyświetlanych błędów. Kolor czerwony oznacza błąd dodatni, zaś kolor niebieski błąd ujemny. Oprócz błędu wzorca wyświetlany jest jego numer porządkowy (No), status (L – wzorzec uczący, T – wzorzec testujący, O – wzorzec nieaktywny) oraz jego nazwę. Dostępne są okienka umożliwiające dodawanie (Rysunek 4.34.) i usuwanie (Rysunek 4.35.) wybranych wzorców uczących z podglądu.



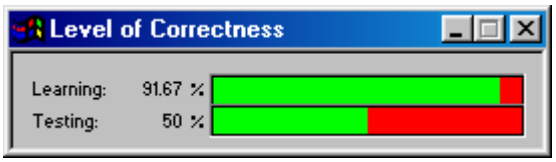
Rysunek 4.34. Okienko umożliwiające dodanie nowych wzorców do widoku z rysunku 4.33. na wybranej pozycji.



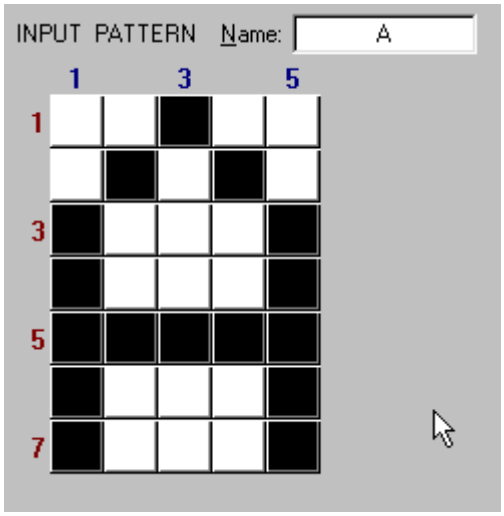
Rysunek 4.35. Okienko umożliwiające usuwanie wybranych wzorców z widoku z rysunku 4.33.



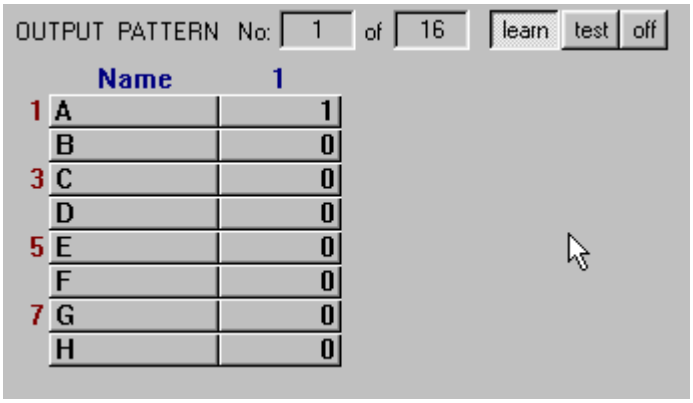
Rysunek 4.36. Okienko porównawcze średnich błędów wszystkich wyjść sieci neuronowej z wyszczególnieniem wartości liczbowej maksymalnego błędu. Kolor czerwony oznacza błąd dodatni, zaś kolor niebieski błąd ujemny. Okienko umożliwia wyświetlanie błędu tylko dla wzorców uczących (jak wyżej), tylko dla wzorców testujących, bądź sumarycznego błędu dla wszystkich wzorców uczących i testujących.



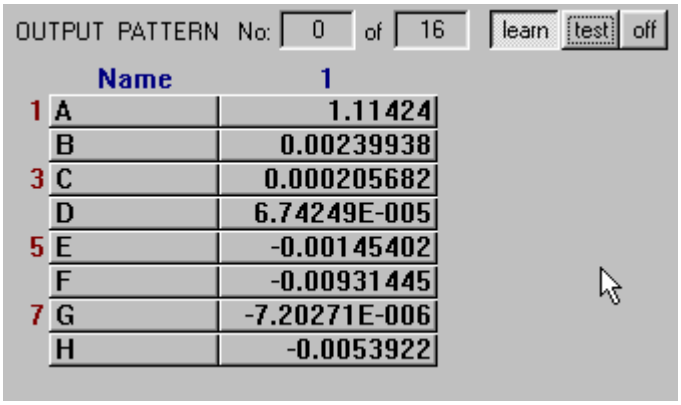
Rysunek 4.37. Okienko przedstawia stan procesu uczenia z wyszczególnieniem poprawnej kwalifikacji wzorców uczących (w 91,67%) i testujących (w 50%). Okienko to może być użyte dla tych wzorców, których sygnał uczący przyjmuje wartości binarne bądź całkowitoliczbowe. Informacja o procencie poprawnych i błędnych klasyfikacji jest również uwidoczniona w formie graficznej odpowiednio kolorem zielonym i czerwonym.



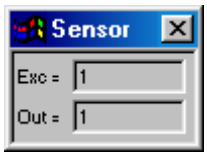
Rysunek 4.38. Widok ten pokazuje wzorzec uczący wraz z jego nazwą. Możliwe jest zadanie wzorca w formie graficznej (dla wzorców binarnych), bądź w formie liczbowej dla wzorców zadanych liczbami całkowitymi bądź wymiernymi. Możliwe jest też skalowanie wielkości pól edycyjnych.



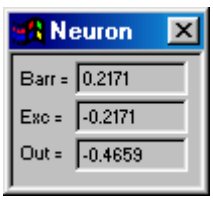
Rysunek 4.39. Widok ten pokazuje sygnał uczący dla ustalonego wzorca uczącego przedstawionego na rysunku 4.38. wraz z jego numerem porządkowym (w polu No) oraz jego statusem (uczący – testujący – wyłączony). Dodatkowo widok zawiera informację o ilości wszystkich wzorców ciągu uczącego.



Rysunek 4.40. Widok ten pokazuje sygnał wyjściowy częściowo nauczonej już sieci neuronowej dla ustalonego wzorca uczącego z rysunku 4.38. i sygnału uczącego z rysunku 4.39.

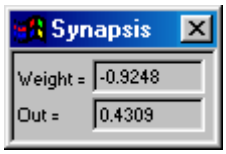


Rysunek 4.41. Okienko dynamicznego podglądu stanu pobudzenia sensora oraz jego wartości wyjściowej, otwierane przez naciśnięcie prawego przycisku myszki, gdy jej wskaźnik znajduje się nad wybranym sensorem.

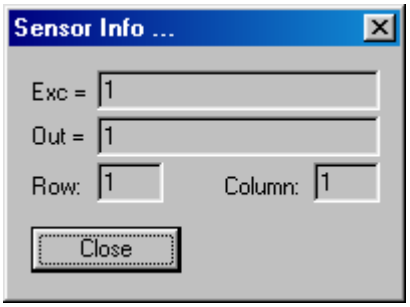


Rysunek 4.42. Okienko dynamicznego podglądu stanu pobudzenia neuronu jego wartości wyjściowej oraz wartości progu, otwierane przez naciśnięcie prawego przycisku myszki, gdy jej wskaźnik znajduje się nad wybranym neuronem.

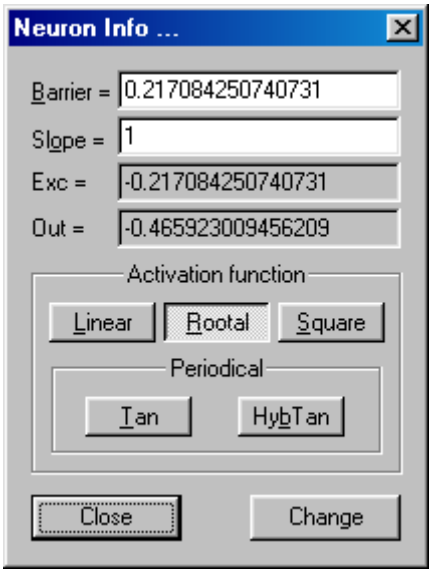
Rysunek 4.43. Okienko dynamicznego podglądu stanu pobudzenia efektora jego wartości wyjściowej oraz wartości progu, otwierane przez naciśnięcie prawego przycisku myszki, gdy jej wskaźnik znajduje się nad wybranym efektorem.



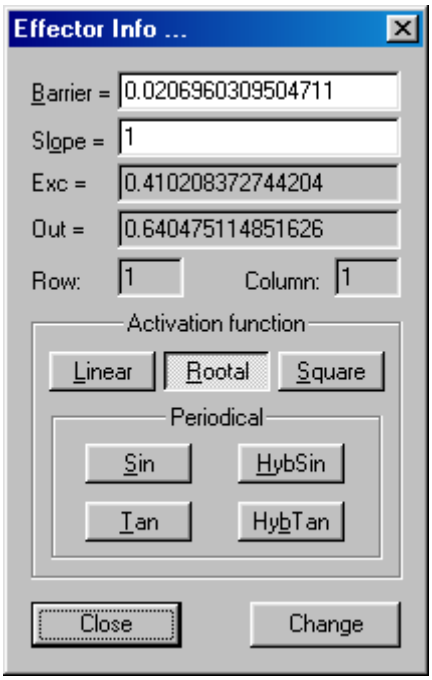
Rysunek 4.44. Okienko dynamicznego podglądu synapsy wraz z jej wartością wagi i wartością pobudzenia elementu postsynaptycznego, otwierane przez naciśnięcie prawego przycisku myszki, gdy jej wskaźnik znajduje się nad wybraną synapsą.



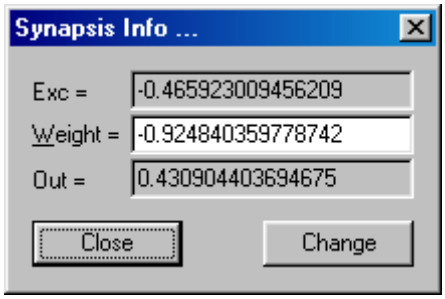
Rysunek 4.45. Statyczne okienko podglądu parametrów sensora wzbogacone o informację o jego podłączeniu do miejsca macierzy wzorca uczącego, otwierane przez naciśnięcie lewego przycisku myszki, gdy jej wskaźnik znajduje się nad wybranym sensorem.



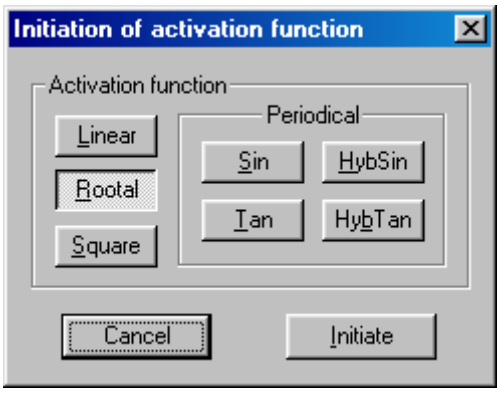
Rysunek 4.46. Statyczne okienko podglądu parametrów neuronu wzbogacone o informacje dotyczące jego funkcji aktywacji oraz parametru stromości z nią związanego. Okienko to umożliwia również zmianę części parametrów, otwierane przez naciśnięcie lewego przycisku myszki, gdy jej wskaźnik znajduje się nad wybranym neuronem.



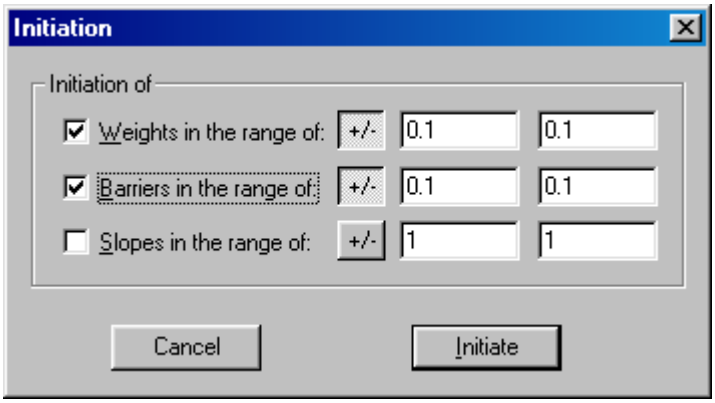
Rysunek 4.47. Statyczne okienko podglądu parametrów efektora wzbogacone o informacje dotyczące jego funkcji aktywacji, parametru stromości z nią związanego oraz o jego podłączeniu do miejsca macierzy sygnału uczącego. Okienko to umożliwia również zmianę części parametrów, otwierane przez naciśnięcie lewego przycisku myszki, gdy jej wskaźnik znajduje się nad wybranym efektorem.



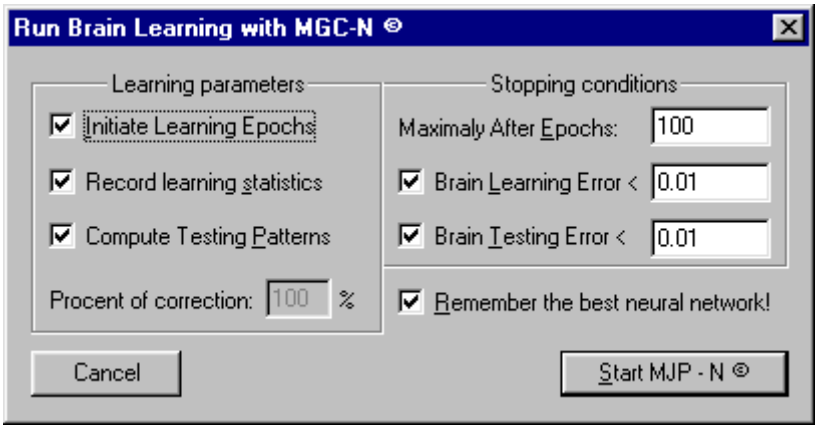
Rysunek 4.48. Statyczne okienko podglądu parametrów synapsy. Okienko to umożliwia również zmianę części parametrów, otwierane przez naciśnięcie lewego przycisku myszki, gdy jej wskaźnik znajduje się nad wybraną synapsą.



Rysunek 4.49. Okienko związane ze zmianą funkcji aktywacji wszystkich bądź wybranych neuronów i efektorów sieci neuronowej.



Rysunek 4.50. Okienko związane z inicjacją wag, progów i parametrów stromości funkcji aktywacji wszystkich bądź wybranych synaps, neuronów i efektorów sieci neuronowej.



Rysunek 4.51. Okienko umożliwia ustawienie parametrów stopu i innych opcji procesu uczenia metodą sterowanych kompromisów.

4.6.4. Podgląd procesu zmian parametrów sieci neuronowej

Aplikacja „Brain for Problem" umożliwia również zapisać całą historię zmian wszystkich istotnych (z punktu widzenia nauki sieci) parametrów sieci neuronowej, w celu ich późniejszego przeanalizowania lub dokonania statystyk na ich podstawie. Zapis danych o procesie zmian parametrów dokonywany jest poprzez uaktywnienie odpowiedniej opcji w okienku umożliwiającym określenie parametrów procesu uczenia (rysunek 4.50) lub przez naciśnięcie przycisku 

. Dane te są zapisywane na dysku ze względu na swoją obszerność. Przełączanie widoku sieci neuronowej i podglądu wybranych parametrów procesu uczenia dokonywany jest przy użyciu przycisków  . Istnieje również możliwość wyboru parametrów (rysunek 4.52.), które są następnie zestawiane w widoku ich podglądu (rysunek 4.53.). Wybór owych parametrów możliwy jest po naciśnięciu przycisku 

. Możliwy jest podgląd następujących parametrów procesu uczenia:

- wartości pobudzenia (*input*) danego sensora, neuronu, efektora lub synapsy (EXC Is),
- wartości przetworzonej (*output*) przez dany sensor, neuron, efektor lub synapsę (OUT Is),
- wartości sygnału uczącego dla danego parametru rozważanego sensora, neuronu, efektora lub synapsy (Lrn),
- wartości korekty, jaka jest związana z danym parametrem rozważanego sensora, neuronu, efektora lub synapsy (Corr),
- wartości błędu dla danego parametru i wszystkich wzorców uczących (Err), również w rozbiciu na jego część dodatnią (ErrP) i ujemną (ErrN),
- wartości, na podstawie których obliczany jest rozkład błędu na poszczególne wagi i próg podczas fazy korekty błędów (DD i DC).

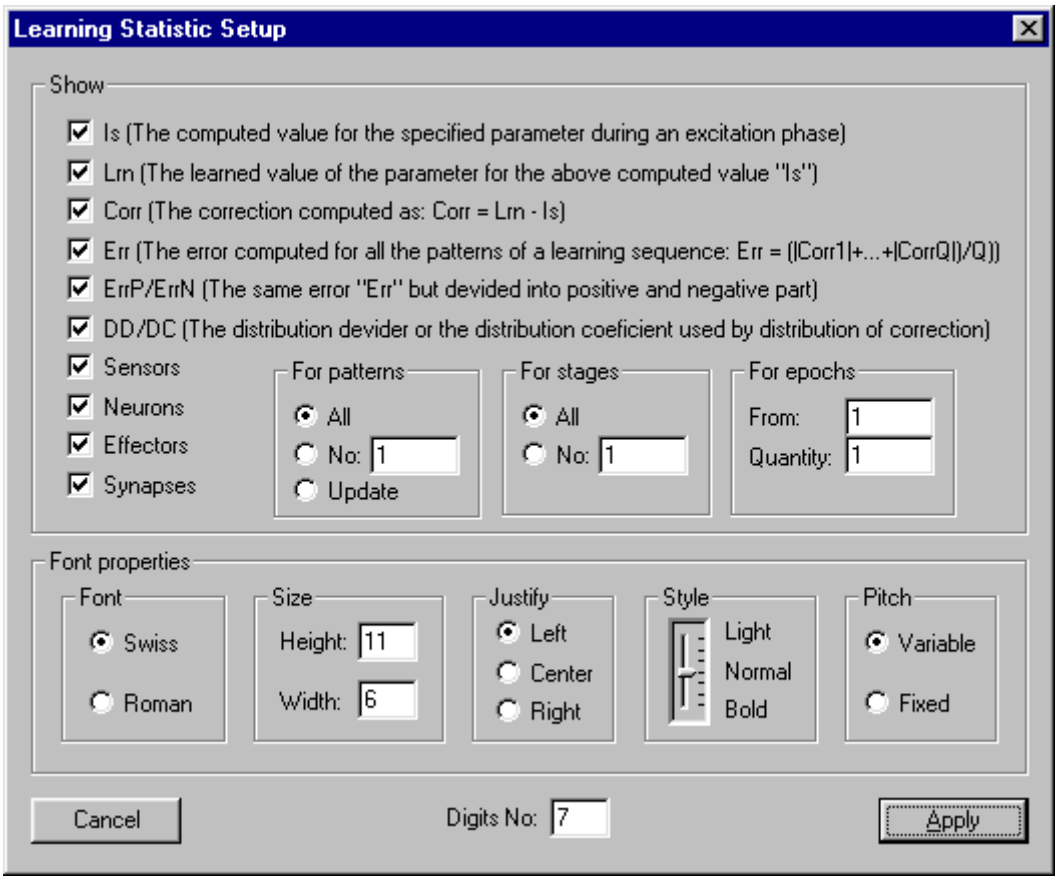
Można również dokonywać podglądu dla wybranej grupy interesujących węzłów sieci:

- sensorów,
- neuronów,
- efektorów,
- synaps.

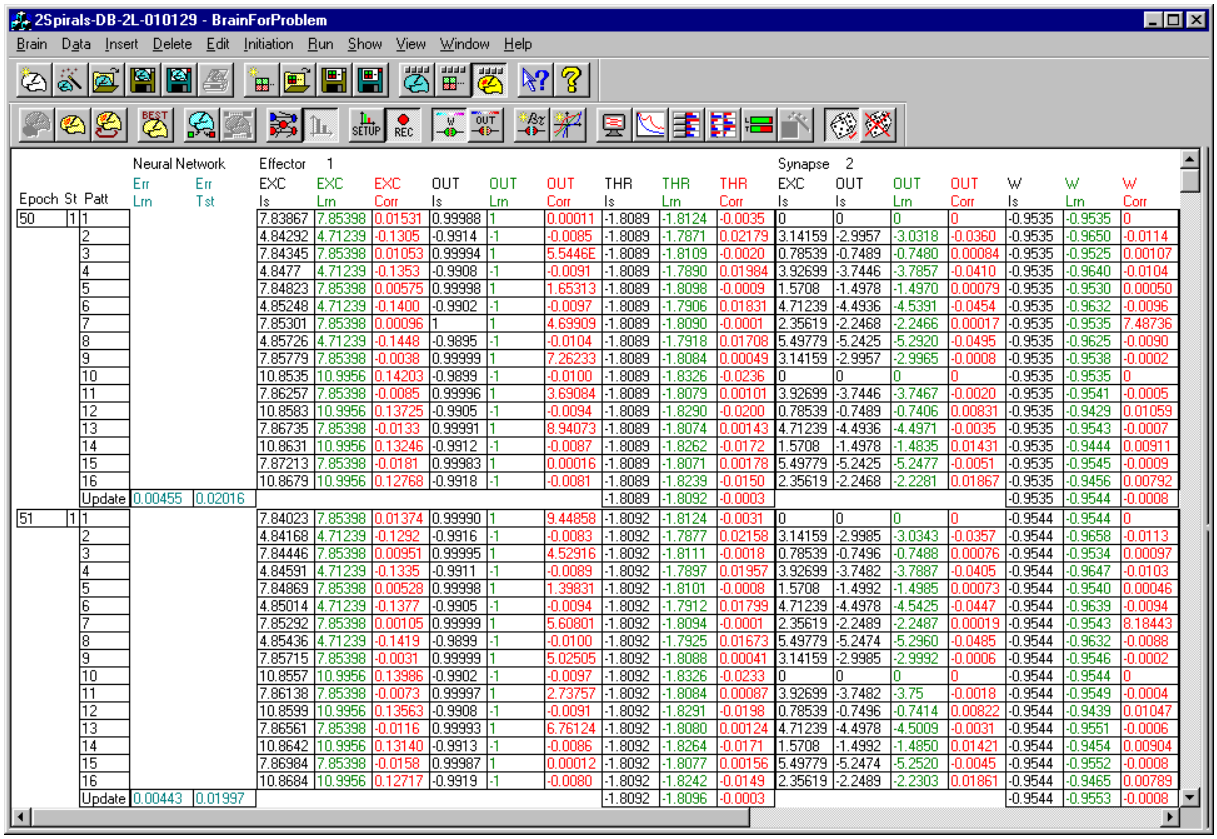
Parametry te mogą być prezentowane dla:

- całego ciągu uczącego (wszystkich wzorców uczących),
- wybranego wzorca uczącego,
- fazy aktualizacji wag i progów (tj. podgląd zawartego kompromisu) (rysunek 4.54.),
- wszystkich etapów uczących,
- wybranego etapu uczącego,
- wybranego zakresu epok uczących.

Dla udogodnienia i optymalizacji widoku możliwy jest również wybór różnych opcji w odniesieniu do czcionki, wielkości i grubości znaków, sposobu wyrównywania tekstu w komórkach tabeli.



Rysunek 4.52. Okienko umożliwiające wyboru parametrów procesu uczenia sieci neuronowej, jakie będą zestawione w głównym widoku (rysunek 4.53.).



			Neural Network		Effector 1								Synapse 2						
			Err	Err	EXC	EXC	EXC	OUT	OUT	OUT	THR	THR	THR	EXC	OUT	OUT	OUT	W	W	W
Epoch	St	Patt	Lrn	Tst	Is	Lrn	Corr	Is	Lrn	Corr	Is	Lrn	Corr	Is	Is	Lrn	Corr	Is	Lrn	Corr
50	1	1			7.83867	7.85398	0.01531	0.99988	1	0.00011	-1.8089	-1.8124	-0.0035	0	0	0	0	-0.9535	-0.9535	0
		2			4.84292	4.71239	-0.1305	-0.9914	-1	-0.0085	-1.8089	-1.7871	0.02179	3.14159	-2.9957	-3.0318	-0.0360	-0.9535	-0.9650	-0.0114
		3			7.84345	7.85398	0.01053	0.99994	1	5.5446E	-1.8089	-1.8109	-0.0020	0.78539	-0.7489	-0.7480	0.00084	-0.9535	-0.9525	0.00107
		4			4.8477	4.71239	-0.1353	-0.9908	-1	-0.0091	-1.8089	-1.7890	0.01984	3.92699	-3.7446	-3.7857	-0.0410	-0.9535	-0.9640	-0.0104
		5			7.84823	7.85398	0.00575	0.99998	1	1.65313	-1.8089	-1.8098	-0.0009	1.5708	-1.4978	-1.4970	0.00079	-0.9535	-0.9530	0.00050
		6			4.85248	4.71239	-0.1400	-0.9902	-1	-0.0097	-1.8089	-1.7906	0.01831	4.71239	-4.4936	-4.5391	-0.0454	-0.9535	-0.9632	-0.0096
		7			7.85301	7.85398	0.00096	1	1	4.69909	-1.8089	-1.8090	-0.0001	2.35619	-2.2468	-2.2466	0.00017	-0.9535	-0.9535	7.48736
		8			4.85726	4.71239	-0.1448	-0.9895	-1	-0.0104	-1.8089	-1.7918	0.01708	5.49779	-5.2425	-5.2920	-0.0495	-0.9535	-0.9625	-0.0090
		9			7.85779	7.85398	-0.0038	0.99999	1	7.26233	-1.8089	-1.8084	0.00049	3.14159	-2.9957	-2.9965	-0.0008	-0.9535	-0.9538	-0.0002
		10			10.8535	10.9956	0.14203	-0.9899	-1	-0.0100	-1.8089	-1.8326	-0.0236	0	0	0	0	-0.9535	-0.9535	0
		11			7.86257	7.85398	-0.0085	0.99996	1	3.69084	-1.8089	-1.8079	0.00101	3.92699	-3.7446	-3.7467	-0.0020	-0.9535	-0.9541	-0.0005
		12			10.8583	10.9956	0.13725	-0.9905	-1	-0.0094	-1.8089	-1.8290	-0.0200	0.78539	-0.7489	-0.7406	0.00831	-0.9535	-0.9429	0.01059
		13			7.86735	7.85398	-0.0133	0.99991	1	8.94073	-1.8089	-1.8074	0.00143	4.71239	-4.4936	-4.4971	-0.0035	-0.9535	-0.9543	-0.0007
		14			10.8631	10.9956	0.13246	-0.9912	-1	-0.0087	-1.8089	-1.8262	-0.0172	1.5708	-1.4978	-1.4835	0.01431	-0.9535	-0.9444	0.00911
		15			7.87213	7.85398	-0.0181	0.99983	1	0.00016	-1.8089	-1.8071	0.00178	5.49779	-5.2425	-5.2477	-0.0051	-0.9535	-0.9545	-0.0009
		16			10.8679	10.9956	0.12768	-0.9918	-1	-0.0081	-1.8089	-1.8239	-0.0150	2.35619	-2.2468	-2.2281	0.01867	-0.9535	-0.9456	0.00792
		Update	0.00455	0.02016							-1.8089	-1.8092	-0.0003					-0.9535	-0.9544	-0.0008
51	1	1			7.84023	7.85398	0.01374	0.99990	1	9.44858	-1.8092	-1.8124	-0.0031	0	0	0	0	-0.9544	-0.9544	0
		2			4.84168	4.71239	-0.1292	-0.9916	-1	-0.0083	-1.8092	-1.7877	0.02158	3.14159	-2.9985	-3.0343	-0.0357	-0.9544	-0.9658	-0.0113
		3			7.84446	7.85398	0.00951	0.99995	1	4.52916	-1.8092	-1.8111	-0.0018	0.78539	-0.7496	-0.7488	0.00076	-0.9544	-0.9534	0.00097
		4			4.84591	4.71239	-0.1335	-0.9911	-1	-0.0089	-1.8092	-1.7897	0.01957	3.92699	-3.7482	-3.7887	-0.0405	-0.9544	-0.9647	-0.0103
		5			7.84869	7.85398	0.00528	0.99998	1	1.39831	-1.8092	-1.8101	-0.0008	1.5708	-1.4992	-1.4985	0.00073	-0.9544	-0.9540	0.00046
		6			4.85014	4.71239	-0.1377	-0.9905	-1	-0.0094	-1.8092	-1.7912	0.01799	4.71239	-4.4978	-4.5425	-0.0447	-0.9544	-0.9639	-0.0094
		7			7.85292	7.85398	0.00105	0.99999	1	5.60801	-1.8092	-1.8094	-0.0001	2.35619	-2.2489	-2.2487	0.00019	-0.9544	-0.9543	8.18443
		8			4.85436	4.71239	-0.1419	-0.9899	-1	-0.0100	-1.8092	-1.7925	0.01673	5.49779	-5.2474	-5.2960	-0.0485	-0.9544	-0.9632	-0.0088
		9			7.85715	7.85398	-0.0031	0.99999	1	5.02505	-1.8092	-1.8088	0.00041	3.14159	-2.9985	-2.9992	-0.0006	-0.9544	-0.9546	-0.0002
		10			10.8557	10.9956	0.13986	-0.9902	-1	-0.0097	-1.8092	-1.8326	-0.0233	0	0	0	0	-0.9544	-0.9544	0
		11			7.86138	7.85398	-0.0073	0.99997	1	2.73757	-1.8092	-1.8084	0.00087	3.92699	-3.7482	-3.75	-0.0018	-0.9544	-0.9549	-0.0004
		12			10.8599	10.9956	0.13563	-0.9908	-1	-0.0091	-1.8092	-1.8291	-0.0198	0.78539	-0.7496	-0.7414	0.00822	-0.9544	-0.9439	0.01047
		13			7.86561	7.85398	-0.0116	0.99993	1	6.75124	-1.8092	-1.8080	0.00124	4.71239	-4.4978	-4.5009	-0.0031	-0.9544	-0.9551	-0.0006
		14			10.8642	10.9956	0.13140	-0.9913	-1	-0.0086	-1.8092	-1.8264	-0.0171	1.5708	-1.4992	-1.4850	0.01421	-0.9544	-0.9454	0.00904
		15			7.86384	7.85398	-0.0158	0.99987	1	0.00012	-1.8092	-1.8077	0.00156	5.49779	-5.2474	-5.2520	-0.0045	-0.9544	-0.9552	-0.0008
		16			10.8684	10.9956	0.12717	-0.9919	-1	-0.0080	-1.8092	-1.8242	-0.0149	2.35619	-2.2489	-2.2303	0.01861	-0.9544	-0.9465	0.00789
		Update	0.00443	0.01997							-1.8092	-1.8096	-0.0003					-0.9544	-0.9553	-0.0008

Rysunek 4.53. Główny widok aplikacji w włączonym przykładowym zestawieniem parametrów procesu uczenia.

			Neural Network		Effector 1			Synapse 2			Synapse 3		
			Err	Err	THR	THR	THR	W	W	W	W	W	W
Epoch	St	Patt	Lrn	Tst	Is	Lrn	Corr	Is	Lrn	Corr	Is	Lrn	Corr
1	1	Update	0.03015	0.07582	-1.5	-1.7626	-0.2625	-1	-0.8632	0.13673	5	5.87532	0.87532
2	1	Update	0.02998	0.05491	-1.7626	-1.7719	-0.0093	-0.8632	-0.8635	-0.0002	5.87532	5.90639	0.03107
3	1	Update	0.02865	0.05280	-1.7719	-1.7734	-0.0015	-0.8635	-0.8668	-0.0032	5.90639	5.91158	0.00519
4	1	Update	0.02732	0.05112	-1.7734	-1.7748	-0.0013	-0.8668	-0.8701	-0.0032	5.91158	5.9161	0.00451
5	1	Update	0.02606	0.04952	-1.7748	-1.7761	-0.0013	-0.8701	-0.8732	-0.0031	5.9161	5.92048	0.00438
6	1	Update	0.02487	0.04800	-1.7761	-1.7774	-0.0012	-0.8732	-0.8763	-0.0031	5.92048	5.92475	0.00426
7	1	Update	0.02373	0.04654	-1.7774	-1.7786	-0.0012	-0.8763	-0.8794	-0.0030	5.92475	5.9289	0.00415
8	1	Update	0.02264	0.04515	-1.7786	-1.7798	-0.0012	-0.8794	-0.8823	-0.0029	5.9289	5.93294	0.00404
9	1	Update	0.02161	0.04382	-1.7798	-1.7810	-0.0011	-0.8823	-0.8852	-0.0028	5.93294	5.93688	0.00393
10	1	Update	0.02063	0.04256	-1.7810	-1.7822	-0.0011	-0.8852	-0.8880	-0.0027	5.93688	5.94071	0.00382

Rysunek 4.54. Widok zestawienia kompromisowych wag i progów obliczonych dla wszystkich wzorców uczących dla kolejnych 10 epok.

5. BADANIA EFEKTYWNOŚCI I UŻYTECZNOŚCI OPRACOWANYCH METOD

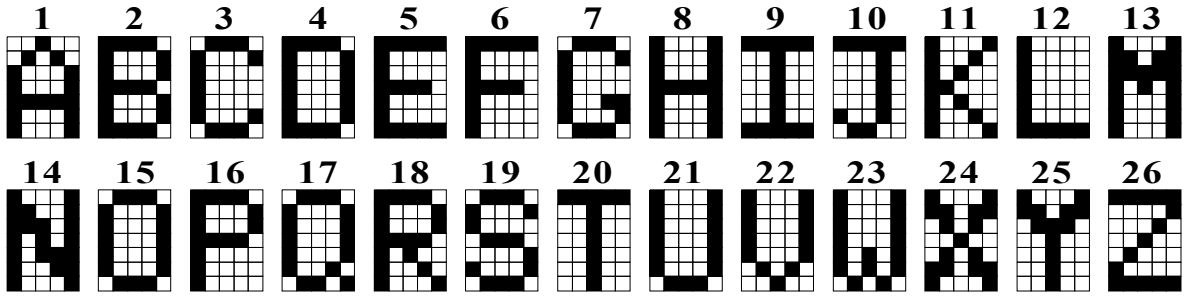
Badanie efektywności opracowanych metod ma na celu praktyczne zweryfikowanie użyteczności proponowanych w pracy metod uczenia. Oczywiście badanie takie nigdy nie może być w pełni wyczerpujące, dlatego w pracy przedstawimy raczej ilustrację działania opisanych metod niż ich wyczerpujące badanie. Jednak na podstawie kilku wybranych i zaprezentowanych dalej przykładów można zauważyć kilka ważnych cech opracowanych metod, stąd przytoczeniu tutaj wyników tych badań wydaje się wysoce celowe. Obserwacji podlegają przede wszystkim te parametry zaproponowanych metod, które są kluczowe dla ich działania, jak również te, które były celem ich usprawnienia. Zebrane wyniki mogą posłużyć do porównania wyników działania zaprezentowanej metody z wynikami zebranymi na podstawie innych metod uczących. Takie obserwacje mogą ujawnić zalety opisanych w pracy metod i mogą zdecydować o wyborze przez użytkownika sieci neuronowej konkretnej metody uczącej dla rozwiązania podejmowanego przez niego problemu.

W poniższych przykładach wykorzystano obydwie opisane w tej pracy metody. W przypadku pierwszej opisanej metody automatycznej konfiguracji sieci neuronowych, skupiono się nad uwidocznieniem możliwości konfiguracji sieci neuronowej połączonej z procesem automatycznej redukcji synaps. W przypadku drugiej zaproponowanej metody sterowanych kompromisów, kryjącej w sobie dużo większy potencjał i możliwości, pokazano dla wybranych przykładów efekty jej działania. Do badań jak również prezentacji wyników użyto aplikacji „**Optimal Recognition**” oraz „**Brain for Problem**”, które zostały napisane specjalnie dla potrzeb tej pracy.

5.1. Zastosowanie metody automatycznej konfiguracji sieci neuronowej do rozpoznawania obrazów.

Problem rozpoznawania obrazów jest zazwyczaj związany ze wspomnianym w rozdz. 3.3. problemem przesunięć, rotacji i skalowania, które to przekształcenia obrazu niekorzystnie wpływają także na działanie algorytmu opisanego w tej pracy. Jednak dzięki zastosowaniu stosowanych metod normalizacji obrazu przed jego wprowadzeniem na wejście sieci, a także na skutek użycia grubego rastra do reprezentacji obrazu można pewne drobne deformacje wejściowego obrazu skutecznie zamortyzować.

Badanie efektywności i użyteczności opracowanej metody automatycznej konfiguracji sieci neuronowej zostało przeprowadzone w oparciu o przykład 26-literowego alfabetu (Rysunek 5.1.), który był już wcześniej używany do zademonstrowania działania programu w rozdziale 3.4.



Rysunek 5.1. 26-literowy alfabet użyty do pomiarów efektywności działania zaproponowanej metody

W ramach opisywanych tu badań algorytmu wykonano szereg doświadczeń, których celem było sprawdzenie zdolności redukcyjnych i konfiguracyjnych zaproponowanej metody w odniesieniu do przytoczonego przykładu rozpoznawania liter. Celem tych doświadczeń było zweryfikowanie, jak wartości charakterystyki *FNP* wpływają na zdolności redukcyjne metody względem synaps i jak taka redukcja wpływa na jakość rozpoznawania i w szczególności jakość uogólniania wzorców. Biorąc pod uwagę, że parametry *FNP* są praktycznie jedynymi parametrami, na które konstruktor sieci może wpływać, zadanie budowy sieci neuronowej, konfiguracji jej architektury i parametrów dla danego ciągu uczącego staje się bardzo proste w porównaniu do innych metod uczenia sieci neuronowych.

Tabela 5.1. Podsumowanie badania wpływu charakterystyki *FNP* na stopień redukcji synaps oraz na jakość rozpoznawania i uogólniania wzorców

Charakte-rystyka	Ilość synaps	Zredukowane synapsy		Jakość uogólniania	Jakość rozpoznawania	Rozpoznanie 					
FNP	**SN**	**Ilość**	**%**	**Q_G%**	**Q_R%**	**jako**	**w %**	**jako**	**w %**	**jako**	**w %**
100|100|100	910	0	0	100,0	100,0	A	100	A	-100	A	77,6
70|60|90	854	56	6	96,1	**104,2**	A	100	A	-100	A	77,1
70|40|80	791	119	13	86,5	**104,2**	A	100	A	-100	A	83,7
60|40|70	690	220	24	74,7	38,7	A	100	A	-100	A	83,0
60|30|60	578	332	36	68,1	38,7	A	100	A	-100	A	83,0
55|30|55	461	449	49	50,2	40,9	A	100	A	-100	A	87,4
52|28|47	417	493	54	48,2	**40,9**	A	100	A	-100	A	87,4
45|25|40	321	589	65	31,4	**0,0**	A	100	A	-100	A	92,6
35|23|30	305	605	66	20,3	0,0	A	100	A	-100	A	100,0
30|20|30	253	657	72	14,3	0,0	A	100	A	-100	A	100,0
20|20|20	234	676	74	9,1	0,0	A	100	A	-100	A	100,0
15|13|15	166	744	82	5,4	0,0	A	100	A	-100	**A,J**	100,0
12|10|12	136	774	85	5,7	0,0	A	100	A	-100	**A,J,O,Z**	100,0
8|8|8	107	803	88	**0,0**	0,0	**A**	100	**A**	-100	**A,J,M,N,O,Z**	100,0
						I,T,W,Y	-100	**I,T,W,Y**	100	**I,T,W**	-100,0

Z powyższej tabeli wynika, iż w badanej metodzie redukcja synaps nie musi za sobą pociągać od razu brak rozpoznania wzorców uczących, a nawet przykładowy wzorzec testujący długo jest poprawnie klasyfikowany. W przypadku wzorców zaburzonych poprawna klasyfikacja zależy oczywiście od tego, jak bardzo dany wzorzec odstaje od wzorca uczącego i czy najważniejsze jego cechy binarne są zaburzone, czy nie. Można dostrzec, że przy bardzo dużej redukcji synaps rzędu 82% zaczyna dochodzić do niejednoznacznego rozpoznania zaburzonego wzorca, a przy redukcji synaps rzędu 88% dochodzi nawet do niejednoznaczności w przypadku rozpoznawania wzorca uczącego. Trudno by było przetestować wszystkie możliwe kombinacje, szczególnie dla wzorców zaburzonych. Zapewne można by było znaleźć takie zaburzone wzorce, które już dla mniejszej redukcji synaps będą niejednoznacznie bądź niepoprawnie rozpoznane. Niezerowa jakość rozpoznawania gwarantuje nam bowiem tylko poprawne i jednoznaczne rozpoznanie wzorców uczących, a jak widać z tabeli Q_R utrzymuje się na poziomie dodatnim jeszcze tylko przy 54% redukcji synaps, co nie jest wcale mało! W przypadku jakości rozpoznawania może czasami dochodzić do sytuacji jej poprawy (104,2%), jak to widać przy 6% i 13% redukcji synaps. Poprawa taka wynika jednak tylko z faktu odrzucenia części mało istotnych cech

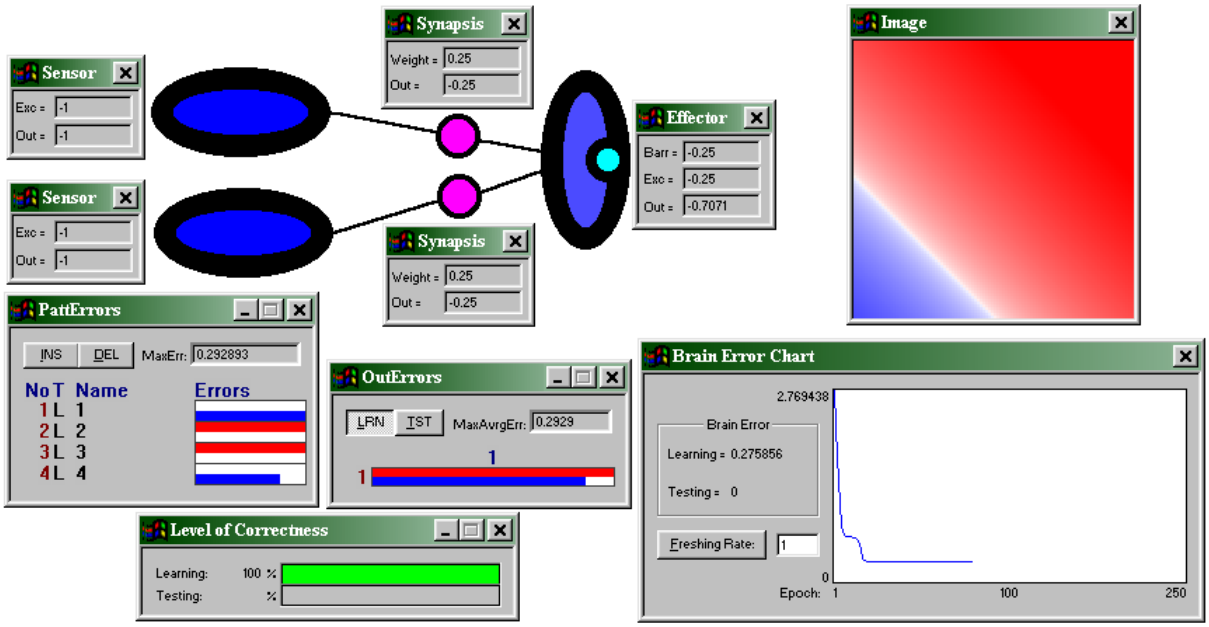
wzorców, co zarazem powoduje zwiększenie minimalnej odległości wzorców uczących względem siebie.

5.2. Uczenie funkcji logicznych.

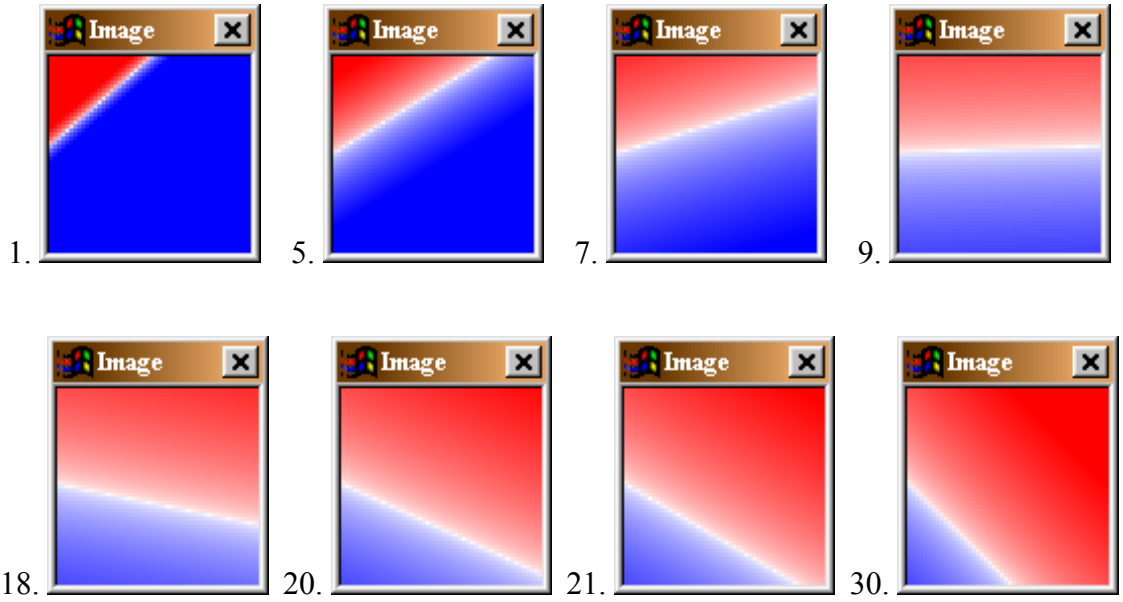
Funkcje logiczne są jednym z prostszych przykładów zastosowania opisanej metody sterowanych kompromisów, jednak dzięki powszechnej znajomości tych funkcji, można w łatwy sposób zademonstrować działanie metody, jak również zweryfikować jakość oraz szybkość jej działania. Naukę sieci neuronowej zademonstrowano na przykładzie funkcji: OR (funkcja liniowo separowalna) oraz XOR (funkcji, która nie jest liniowo separowalna). W przypadku funkcji XOR zastosowano dwie różne architektury sieci neuronowej:

- dwuwarstwową – składającą się z sensorów i pojedynczego efektora, w którym zastosowano periodyczną sinusoidalną funkcję aktywacji, co pozwoliło na uproszczenie architektury sieci neuronowej.
- trójwarstwową – składającą się z sensorów, pojedynczego neuronu ukrytego oraz pojedynczego efektora. W tym przypadku zastosowano jak dla neuronu tak dla efektora pierwiastkowe funkcje aktywacji.

Problem OR – jest klasycznym przykładem prostej liniowo-separowalnej funkcji logicznej. Dla tego problemu nauka przebiega szybko i bez zakłóceń. Rozwiązanie optymalne znajdywane jest w procesie uczenia sieci neuronowej metodą sterowanych kompromisów w przeciągu kilkunastu-kilkudziesięciu epok uczących. Na rysunkach 5.2-3. zilustrowano graficznie proces uczenia się sieci w postaci graficznych obrazów oraz funkcji błędu.



Rysunek 5.2. Na powyższym rysunku widoczna jest zastosowana dwuwarstwowa architektura sieci neuronowej oraz liczne okienka przedstawiające stan poszczególnych parametrów sieci w 100 epoce uczącej. Z wykresu błędu sieci widać, że efektywna nauka sieci trwała ok. 30 epok, po których w efekcie ustalił się pewien stabilny stan sieci. W okienku przedstawiającym graficznie obraz nauczonej sieci czerwonym kolorem oznaczono wartości logicznej prawdy, zaś kolorem niebieskim wartości logicznego fałszu. Jak widać wygląd tego obrazu jest w pełni zgodny z intuicją i definicją funkcji logicznej OR. Inne okienka umożliwiają dokładny podgląd wag i progu oraz błędów związanych z poszczególnymi wzorcami uczącymi z separacją na błędy o wartościach dodatnich (oznaczone kolorem czerwonym) oraz błędy o wartościach ujemnych (oznaczone kolorem niebieskim).



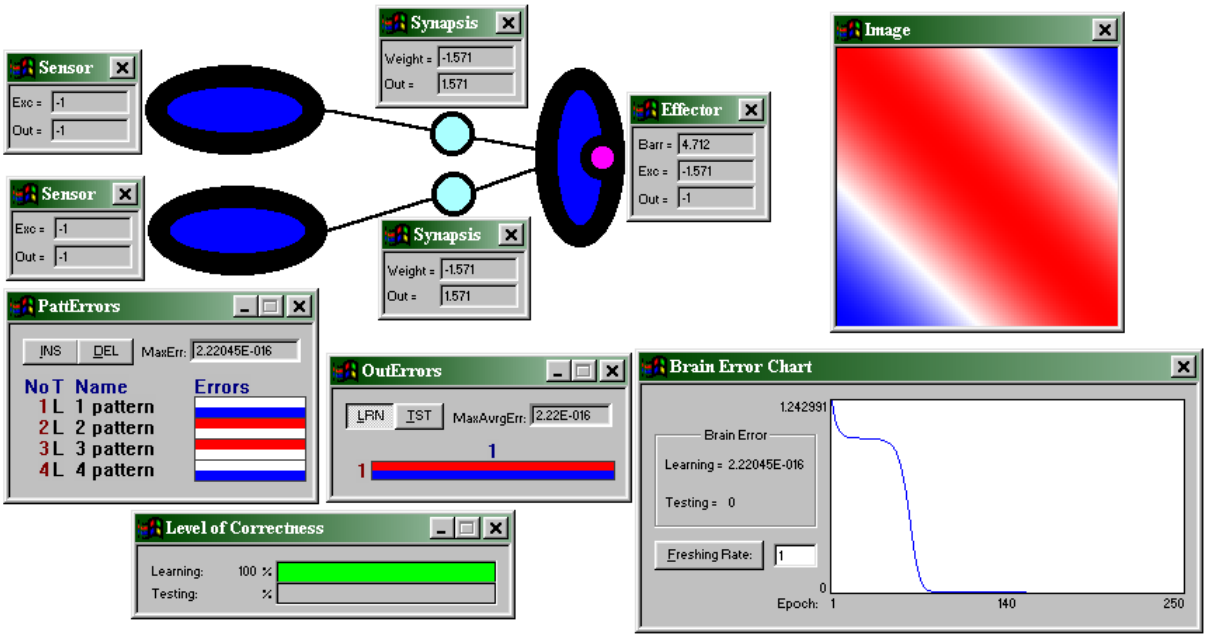
Rysunek 5.3. Na rysunku zobrazowano kolejne obrazy stanu sieci neuronowej podczas jej nauki dla opisanego problemu OR dla wybranych epok uczących: 1, 5, 7, 9, 18, 20, 21 i 30. Dzięki takim obrazom można w prosty sposób obserwować, w jaki sposób przebiega nauka sieci oraz w jakim kierunku zmierza.

Tabela 5.2. W poniższej tabeli zobrazowano proces zmian progu efektora (THR) oraz wag (W) w kolejnych epokach uczących (Epoch) w powiązaniu z malejącym błędem sieci neuronowej (Err Lrn) dla wzorców uczących. Można porównać, jak aktualne wartości wag i progu (Is) są korygowane na podstawie kompromisowej ich korekty (Corr).

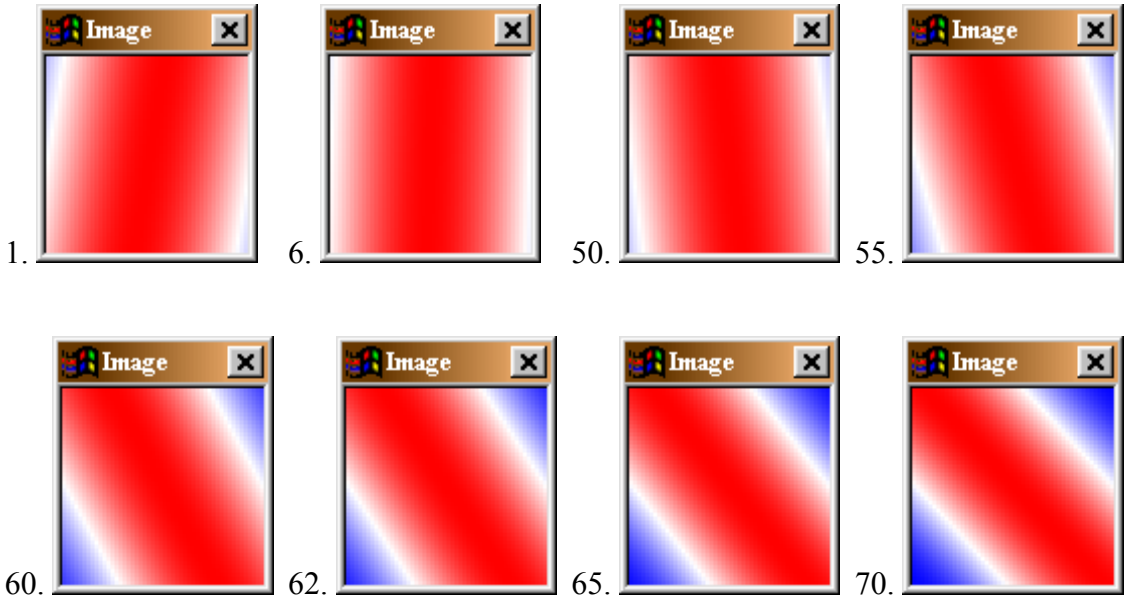
			Neural Network		Effector 1		Synapse 2		Synapse 3	
Epoch	St	Patt	Err Lrn	Err Tst	THR Is	THR Corr	W Is	W Corr	W Is	W Corr
1	1	Update	2.76944	0	4.47981	-1.69407	-3.73302	1.18878	4.29472	-1.38883
2	1	Update	2.24305	0	2.78574	-1.02682	-2.54424	0.863205	2.90588	-0.937082
3	1	Update	1.80838	0	1.75892	-0.653296	-1.68104	0.600166	1.9688	-0.625648
4	1	Update	1.44596	0	1.10562	-0.424633	-1.08087	0.407549	1.34315	-0.415983
5	1	Update	1.13537	0	0.680989	-0.277886	-0.673324	0.272497	0.927171	-0.275195
6	1	Update	0.793372	0	0.403103	-0.180827	-0.400828	0.17918	0.651976	-0.180011
7	1	Update	0.711307	0	0.222275	-0.114616	-0.221647	0.11414	0.471965	-0.114381
8	1	Update	0.672501	0	0.10766	-0.067229	-0.107507	0.0671053	0.357585	-0.067168
9	1	Update	0.652517	0	0.0404306	-0.0316291	-0.040402	0.0316037	0.290417	-0.0316166
10	1	Update	0.646842	0	0.00880149	-0.0082411	-0.00879838	0.00823809	0.2588	-0.00823962
11	1	Update	0.645652	0	0.00056039	-0.00112078	-0.000560291	0.00112058	0.25056	-0.000557846
12	1	Update	0.644867	0	-0.00056039	-0.000556633	0.000560291	0.000556535	0.250002	-2.48394E-006
13	1	Update	0.643308	0	-0.00111702	-0.00110218	0.00111683	0.00110199	0.25	-1.10361E-008
14	1	Update	0.640251	0	-0.00221921	-0.00216114	0.00221881	0.00216076	0.25	-9.68918E-011
15	1	Update	0.634366	0	-0.00438035	-0.00415791	0.00437958	0.00415719	0.25	-1.66178E-012
16	1	Update	0.623408	0	-0.00853825	-0.00771941	0.00853677	0.00771811	0.25	-5.4512E-014
17	1	Update	0.604131	0	-0.0162577	-0.0134511	0.0162549	0.013449	0.25	-3.27516E-015
18	1	Update	0.572917	0	-0.0297088	-0.0211516	0.0297038	0.0211486	0.25	-3.33067E-016
19	1	Update	0.52673	0	-0.0508604	-0.0287971	0.0508524	0.0287938	0.25	-5.55112E-017
20	1	Update	0.457844	0	-0.0796575	-0.0331516	0.0796462	0.0331491	0.25	0
21	1	Update	0.335106	0	-0.112809	-0.0325405	0.112795	0.0325397	0.25	0
22	1	Update	0.309525	0	-0.14535	-0.0281327	0.145335	0.0281338	0.25	0
23	1	Update	0.296644	0	-0.173482	-0.0222371	0.173469	0.0222393	0.25	-5.55112E-017
24	1	Update	0.289111	0	-0.195719	-0.0165627	0.195708	0.0165652	0.25	0
25	1	Update	0.284451	0	-0.212282	-0.0118698	0.212273	0.0118721	0.25	0
26	1	Update	0.281486	0	-0.224152	-0.0082972	0.224145	0.00829904	0.25	0
27	1	Update	0.279566	0	-0.232449	-0.00570673	0.232444	0.00570813	0.25	0
28	1	Update	0.278311	0	-0.238156	-0.00388371	0.238153	0.00388472	0.25	0
29	1	Update	0.277484	0	-0.242039	-0.00262474	0.242037	0.00262545	0.25	0
30	1	Update	0.276938	0	-0.244664	-0.00176576	0.244663	0.00176625	0.25	0

Problem XOR – jest klasycznym przykładem funkcji, która nie jest liniowo-separowalna i zarazem jednym z prostszych benchmarków służących do badania, czy dana metoda radzi sobie z uczeniem danych, które nie można liniowo rozseparować. Fakt ten pociąga za sobą konieczność zastosowania nieco innej architektury sieci neuronowej niż w przypadku problemu OR. Stosując metodę sterowanych kompromisów problem ten można rozwiązać na dwa różne sposoby:

a) użyć funkcji periodycznej (np. sinusoidalnej – por. rozdz. 4.4.2.) jako funkcji aktywacji efektora:



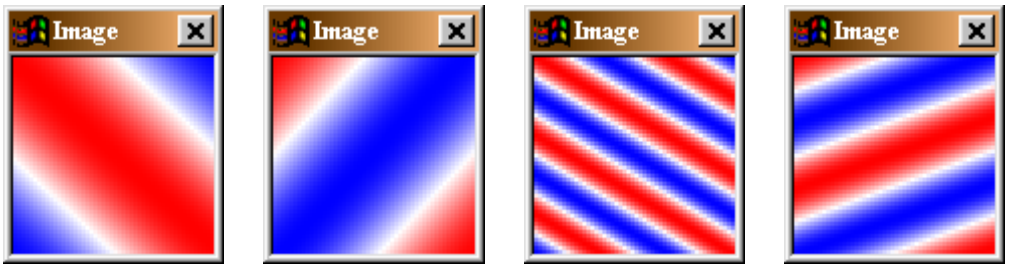
Rysunek 5.4. Powyższy rysunek uwidacznia przebieg nauki dla problemu XOR po zastosowaniu funkcji sinusoidalnej jako funkcji aktywacji efektora. Jak widać z przedstawionego wykresu błędu, po około 70 epokach proces uczenia jest uwieńczony sukcesem i błąd sieci neuronowej jest bliski zeru.



Rysunek 5.5. Na rysunku zobrazowano kolejne obrazy stanu sieci neuronowej podczas jej nauki dla opisanego problemu XOR dla wybranych epok uczących: 1, 6, 50, 55, 60, 62, 65 i 70.

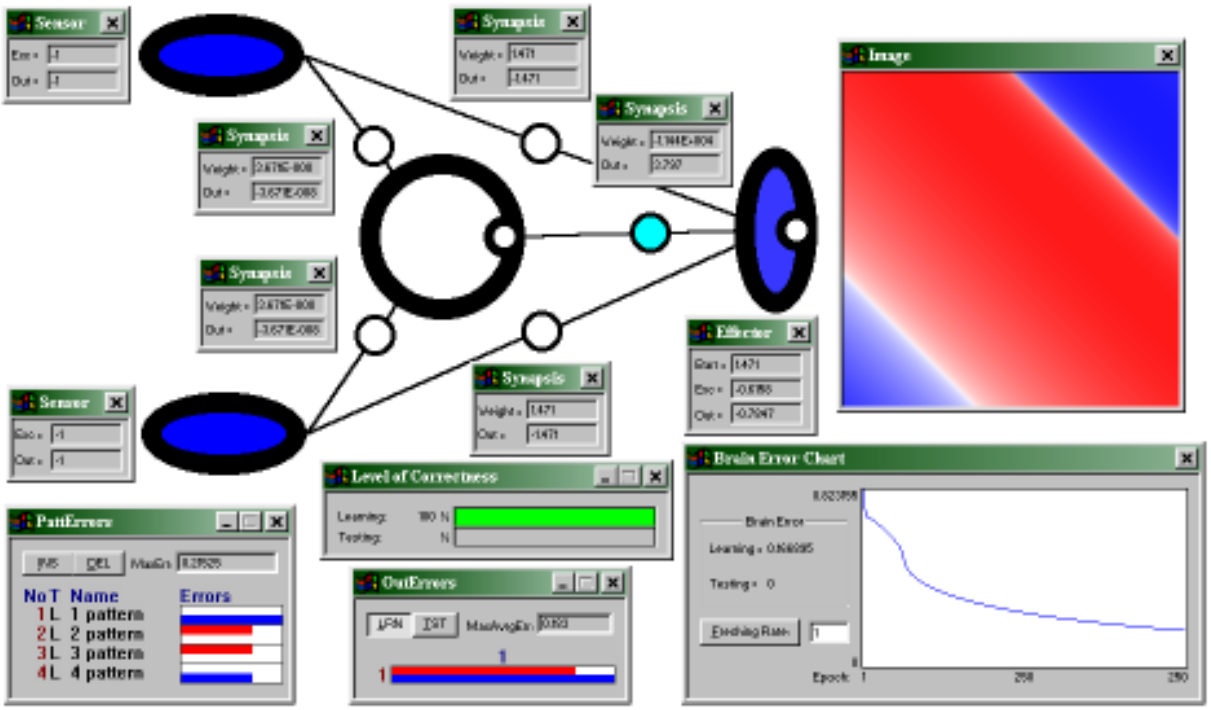
Tabela 5.3. W poniższej tabeli zobrazowano proces zmian progu efektora (THR) oraz wag (W) w kolejnych epokach uczących (Epoch) w powiązaniu z malejącym błędem sieci neuronowej (Err Lrn) dla wzorców uczących.

			Neural Network		Effector 1		Synapse 2		Synapse 3	
Epoch	St	Patt	Err Lrn	Err Tst	THR Is	THR Corr	W Is	W Corr	W Is	W Corr
1	1	Update	1.24299	0	4.52559	0.132241	-1.51524	-0.0131674	0.351878	-0.105831
2	1	Update	1.17552	0	4.65783	0.0395076	-1.52841	-0.0100713	0.246047	-0.0694977
3	1	Update	1.12805	0	4.69734	0.0110254	-1.53848	-0.00775282	0.17655	-0.0481091
4	1	Update	1.09409	0	4.70836	0.00296933	-1.54624	-0.00594964	0.128441	-0.0341923
5	1	Update	1.0695	0	4.71133	0.000782635	-1.55219	-0.00454373	0.0942482	-0.0246828
6	1	Update	1.05154	0	4.71212	0.000203327	-1.55673	-0.003455	0.0695654	-0.0180033
7	1	Update	1.03832	0	4.71232	5.22898E-005	-1.56018	-0.00261815	0.0515621	-0.0132276
8	1	Update	1.02856	0	4.71237	1.3349E-005	-1.5628	-0.00197885	0.0383345	-0.00977035
9	1	Update	1.02132	0	4.71238	3.38955E-006	-1.56478	-0.00149274	0.0285641	-0.00724489
10	1	Update	1.01593	0	4.71239	8.5726E-007	-1.56627	-0.00112441	0.0213192	-0.00538775
11	1	Update	1.01192	0	4.71239	2.16176E-007	-1.5674	-0.000846045	0.0159315	-0.00401526
12	1	Update	1.00892	0	4.71239	5.4394E-008	-1.56824	-0.000636074	0.0119162	-0.00299719
13	1	Update	1.00668	0	4.71239	1.36643E-008	-1.56888	-0.000477923	0.00891904	-0.00223992
14	1	Update	1.005	0	4.71239	3.42846E-009	-1.56936	-0.000358931	0.00667912	-0.00167548
15	1	Update	1.00375	0	4.71239	8.59437E-010	-1.56972	-0.000269473	0.00500364	-0.00125411
16	1	Update	1.00281	0	4.71239	2.15294E-010	-1.56999	-0.00020226	0.00374953	-0.00093918
17	1	Update	1.00211	0	4.71239	5.39053E-011	-1.57019	-0.000151782	0.00281035	-0.000703597
18	1	Update	1.00158	0	4.71239	1.34923E-011	-1.57034	-0.000113885	0.00210675	-0.000527255
19	1	Update	1.00118	0	4.71239	3.37597E-012	-1.57045	-8.54415E-005	0.00157949	-0.000395193
20	1	Update	1.00089	0	4.71239	8.43769E-013	-1.57054	-6.40966E-005	0.0011843	-0.000296255
21	1	Update	1.00067	0	4.71239	2.11386E-013	-1.5706	-4.80812E-005	0.000888046	-0.000222113
22	1	Update	1.0005	0	4.71239	5.32907E-014	-1.57065	-3.60658E-005	0.000665934	-0.00016654
23	1	Update	0.999501	0	4.71239	1.33227E-014	-1.57069	-2.70521E-005	0.000499394	-0.000998787
24	1	Update	0.999376	0	4.71239	3.55271E-015	-1.57072	-2.02902E-005	-0.000499394	-0.0001248
25	1	Update	0.99922	0	4.71239	8.88178E-016	-1.57074	-1.52181E-005	-0.000624194	-0.000155973
26	1	Update	0.999025	0	4.71239	0	-1.57075	-1.14139E-005	-0.000780167	-0.000194922
27	1	Update	0.998781	0	4.71239	0	-1.57076	-8.5606E-006	-0.000975089	-0.000243584
28	1	Update	0.998477	0	4.71239	0	-1.57077	-6.4206E-006	-0.00121867	-0.000304374
29	1	Update	0.998097	0	4.71239	0	-1.57078	-4.81556E-006	-0.00152305	-0.000380302
30	1	Update	0.997622	0	4.71239	0	-1.57078	-3.61176E-006	-0.00190335	-0.000475118
31	1	Update	0.997028	0	4.71239	0	-1.57079	-2.7089E-006	-0.00237847	-0.000593493
32	1	Update	0.996287	0	4.71239	0	-1.57079	-2.03175E-006	-0.00297196	-0.000741234
33	1	Update	0.995361	0	4.71239	0	-1.57079	-1.52387E-006	-0.00371319	-0.000925558
34	1	Update	0.994206	0	4.71239	0	-1.57079	-1.14296E-006	-0.00463875	-0.00115541
35	1	Update	0.992764	0	4.71239	0	-1.57079	-8.57278E-007	-0.00579416	-0.00144187
36	1	Update	0.990965	0	4.71239	0	-1.57079	-6.43009E-007	-0.00723603	-0.0017986
37	1	Update	0.988723	0	4.71239	0	-1.57079	-4.82304E-007	-0.00903463	-0.00224244
38	1	Update	0.985929	0	4.71239	0	-1.57079	-3.61772E-007	-0.0112771	-0.00279402
39	1	Update	0.982451	0	4.71239	0	-1.5708	-2.71371E-007	-0.0140711	-0.00347847
40	1	Update	0.978126	0	4.71239	0	-1.5708	-2.03567E-007	-0.0175496	-0.00432629
41	1	Update	0.972753	0	4.71239	0	-1.5708	-1.52712E-007	-0.0218759	-0.00537409
42	1	Update	0.966091	0	4.71239	0	-1.5708	-1.14569E-007	-0.0272499	-0.0066654
43	1	Update	0.957846	0	4.71239	0	-1.5708	-8.59598E-008	-0.0339153	-0.00825123
44	1	Update	0.947667	0	4.71239	0	-1.5708	-6.4501E-008	-0.0421666	-0.0101903
45	1	Update	0.93514	0	4.71239	0	-1.5708	-4.84053E-008	-0.0523569	-0.0125484
46	1	Update	0.919784	0	4.71239	0	-1.5708	-3.6332E-008	-0.0649052	-0.0153968
47	1	Update	0.901051	0	4.71239	0	-1.5708	-2.72757E-008	-0.080302	-0.0188088
48	1	Update	0.878337	0	4.71239	0	-1.5708	-2.04823E-008	-0.0991108	-0.0228538
49	1	Update	0.851004	0	4.71239	0	-1.5708	-1.53861E-008	-0.121965	-0.0275882
50	1	Update	0.818418	0	4.71239	0	-1.5708	-1.1563E-008	-0.149553	-0.0330421
51	1	Update	0.780016	0	4.71239	0	-1.5708	-8.69483E-009	-0.182595	-0.0392031
52	1	Update	0.735395	0	4.71239	0	-1.5708	-6.54284E-009	-0.221798	-0.0459963
53	1	Update	0.684428	0	4.71239	0	-1.5708	-4.92805E-009	-0.267794	-0.0532648
54	1	Update	0.627396	0	4.71239	0	-1.5708	-3.71619E-009	-0.321059	-0.0607548
55	1	Update	0.565101	0	4.71239	0	-1.5708	-2.80649E-009	-0.381814	-0.0681125
56	1	Update	0.498937	0	4.71239	0	-1.5708	-2.12336E-009	-0.449926	-0.0748998
57	1	Update	0.43086	0	4.71239	0	-1.5708	-1.61008E-009	-0.524826	-0.0806333
58	1	Update	0.363227	0	4.71239	0	-1.5708	-1.22409E-009	-0.605459	-0.0848458
59	1	Update	0.298525	0	4.71239	0	-1.5708	-9.3347E-010	-0.690305	-0.0871597
60	1	Update	0.239021	0	4.71239	0	-1.5708	-7.14276E-010	-0.777465	-0.0873556



Rysunek 5.6. Na rysunku przedstawiono różne stabilne stany sieci uzyskane w wyniku uczenia metodą sterowanych kompromisów dla rozważanego problemu XOR oraz dla sieci neuronowej, w której zastosowano funkcję sinusoidalną (por. rozdz. 4.4.2.) jako funkcję aktywacji efektora. We wszystkich czterech przypadkach uzyskano poprawne rozwiązanie dla wzorców ciągu uczącego, odpowiadającym wartościom uzyskanym w rogach powyższych obrazów (np. we wszystkich lewych górnych rogach obrazów jest kolor czerwony reprezentujący wartość prawdy, zaś we wszystkich prawych górnych rogach jest kolor niebieski reprezentujący wartość fałszu). W dwóch pierwszych obrazach – zgodnie z intuicją – mamy do czynienia z płynnymi przejściami pomiędzy wartościami funkcji logicznej XOR, jednak sieć neuronowa opierając się tylko i wyłącznie o dane zadane ciągiem uczącym prawidłowo znalazła rozwiązanie we wszystkich czterech przypadkach.

b) dodać przynajmniej jeden neuron „ukryty” przy zachowaniu pierwiastkowych (ew. kwadratowych– por. rozdz. 4.1.) funkcji aktywacji neuronu oraz efektora:



Rysunek 5.7. Powyższy rysunek uwidacznia przebieg nauki sieci z wykorzystaniem metody sterowanych kompromisów dla problemu XOR po rozbudowie architektury sieci neuronowej o neuron „ukryty”. Jako funkcje aktywacji zastosowano funkcje pierwiastkowe. Sytuacja wolniejszej zbieżności może być wytłumaczona faktem braku zoptymalizowania metody sterowanych kompromisów pod kątem uczenia sieci wielowarstwowych, które są obiektem dalszych badań i nie stanowią zasadniczej części tej pracy.

1. 

4. 

22. 

32. 

44. 

80. 

150. 

250. 

Rysunek 5.8. Na rysunku zobrazowano kolejne obrazy stanu sieci neuronowej o rozbudowanej architekturze podczas jej nauki dla opisanego problemu XOR dla wybranych epok uczących: 1, 4, 22, 32, 44, 80, 150 i 250.

Tabela 5.4. W poniższej tabeli zobrazowano proces zmian wag (W) i progów (THR) w kolejnych epokach uczących (Epoch). W przedstawionej tabeli można zaobserwować również dwuetapowy (St: 1. i 2.) proces korekty wag i progów rozważanej sieci neuronowej (por. rozdz. 4.4.3.).

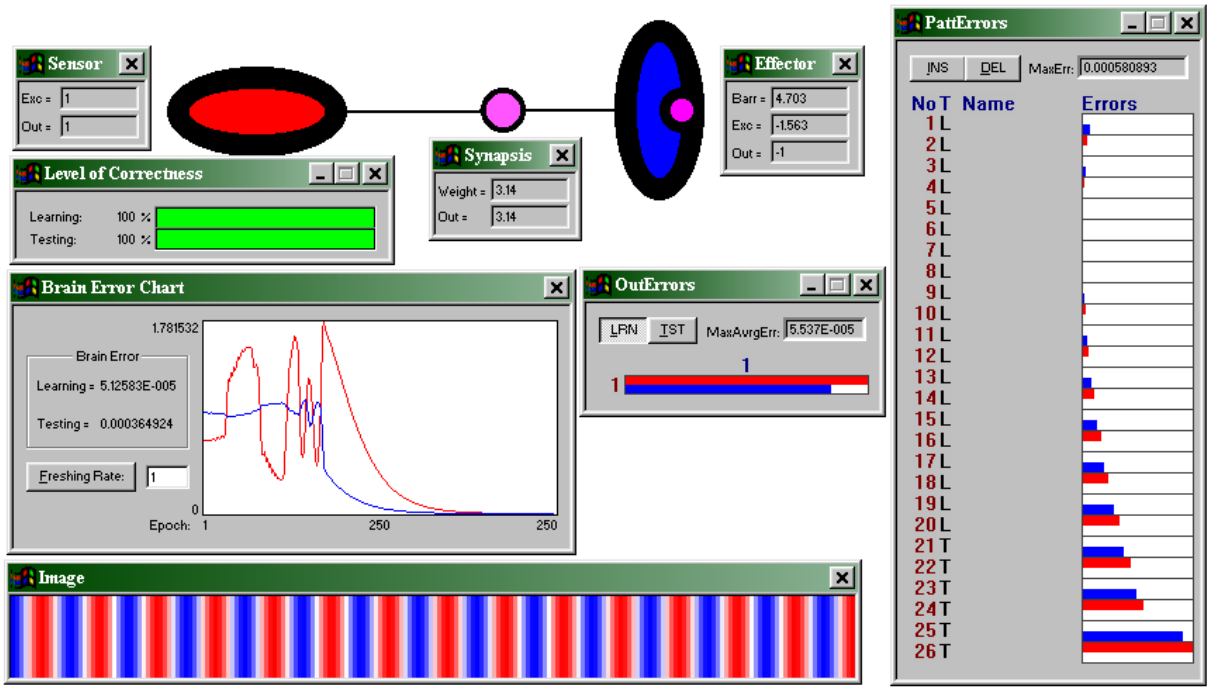
			Neural Network		Effector	1	Synapse	2	Synapse	3	Synapse	4	Neuron	5	Synapse	6	Synapse	7
			Err	Err	THR	THR	W	W	W	W	W	W	THR	THR	W	W	W	W
Epoch	St	Patt	Lrn	Tst	Is	Corr	Is	Corr	Is	Corr	Is	Corr	Is	Corr	Is	Corr	Is	Corr
1	1	Update			0.1	-0.039841	0.1	-0.210461	0.1	-0.039841	0.1	-0.039841	0.1		0.1		0.1	
	2	Update	0.823155	0	0.0601583		-0.110461		0.0601583		0.0601583		0.1	3.05584	0.1	3.05584	0.1	3.05584
2	1	Update			0.0601583	-0.000981	-0.110461	-0.128467	0.0601583	-0.000981	0.0601583	-0.000981	3.15584		3.15584		3.15584	
	2	Update	0.7428	0	0.0591765		-0.238928		0.0591765		0.0591765		3.15584	-0.401769	3.15584	-0.401769	3.15584	-0.401769
3	1	Update			0.0591765	0.0087552	-0.238928	-0.102558	0.0591765	0.0087552	0.0591765	0.0087552	2.75408		2.75408		2.75408	
	2	Update	0.708549	0	0.0679317		-0.341486		0.0679317		0.0679317		2.75408	-0.556046	2.75408	-0.556046	2.75408	-0.556046
4	1	Update			0.0679317	0.0128594	-0.341486	-0.070411	0.0679317	0.0128594	0.0679317	0.0128594	2.19803		2.19803		2.19803	
	2	Update	0.698114	0	0.0807912		-0.411897		0.0807912		0.0807912		2.19803	-0.44086	2.19803	-0.44086	2.19803	-0.44086
5	1	Update			0.0807912	0.0152563	-0.411897	-0.061921	0.0807912	0.0152563	0.0807912	0.0152563	1.75717		1.75717		1.75717	
	2	Update	0.693716	0	0.0960475		-0.473819		0.0960475		0.0960475		1.75717	-0.327904	1.75717	-0.327904	1.75717	-0.327904
6	1	Update			0.0960475	0.0171477	-0.473819	-0.062217	0.0960475	0.0171477	0.0960475	0.0171477	1.42927		1.42927		1.42927	
	2	Update	0.690399	0	0.113195		-0.536037		0.113195		0.113195		1.42927	-0.243519	1.42927	-0.243519	1.42927	-0.243519
7	1	Update			0.113195	0.0187921	-0.536037	-0.064715	0.113195	0.0187921	0.113195	0.0187921	1.18575		1.18575		1.18575	
	2	Update	0.687037	0	0.131987		-0.600752		0.131987		0.131987		1.18575	-0.183619	1.18575	-0.183619	1.18575	-0.183619
8	1	Update			0.131987	0.0202183	-0.600752	-0.067776	0.131987	0.0202183	0.131987	0.0202183	1.00213		1.00213		1.00213	
	2	Update	0.683401	0	0.152206		-0.668528		0.152206		0.152206		1.00213	-0.141561	1.00213	-0.141561	1.00213	-0.141561
9	1	Update			0.152206	0.0214073	-0.668528	-0.071131	0.152206	0.0214073	0.152206	0.0214073	0.860567		0.860567		0.860567	
	2	Update	0.679472	0	0.173613		-0.739659		0.173613		0.173613		0.860567	-0.112028	0.860567	-0.112028	0.860567	-0.112028
10	1	Update			0.173613	0.0223381	-0.739659	-0.074861	0.173613	0.0223381	0.173613	0.0223381	0.748539		0.748539		0.748539	
	2	Update	0.675287	0	0.195951		-0.814521		0.195951		0.195951		0.748539	-0.091193	0.748539	-0.091193	0.748539	-0.091193
11	1	Update			0.195951	0.0229988	-0.814521	-0.079136	0.195951	0.0229988	0.195951	0.0229988	0.657345		0.657345		0.657345	
	2	Update	0.670892	0	0.21895		-0.893657		0.21895		0.21895		0.657345	-0.076360	0.657345	-0.076360	0.657345	-0.076360
12	1	Update			0.21895	0.023388	-0.893657	-0.084132	0.21895	0.023388	0.21895	0.023388	0.580985		0.580985		0.580985	
	2	Update	0.666328	0	0.242338		-0.97779		0.242338		0.242338		0.580985	-0.065630	0.580985	-0.065630	0.580985	-0.065630
13	1	Update			0.242338	0.0235142	-0.97779	-0.090014	0.242338	0.0235142	0.242338	0.0235142	0.515354		0.515354		0.515354	
	2	Update	0.661627	0	0.265852		-1.0678		0.265852		0.265852		0.515354	-0.057668	0.515354	-0.057668	0.515354	-0.057668
14	1	Update			0.265852	0.0233937	-1.0678	-0.096933	0.265852	0.0233937	0.265852	0.0233937	0.457686		0.457686		0.457686	
	2	Update	0.656808	0	0.289246		-1.16474		0.289246		0.289246		0.457686	-0.051534	0.457686	-0.051534	0.457686	-0.051534
15	1	Update			0.289246	0.0230486	-1.16474	-0.10503	0.289246	0.0230486	0.289246	0.0230486	0.406151		0.406151		0.406151	
	2	Update	0.651879	0	0.312294		-1.26977		0.312294		0.312294		0.406151	-0.046581	0.406151	-0.046581	0.406151	-0.046581
16	1	Update			0.312294	0.0225054	-1.26977	-0.114437	0.312294	0.0225054	0.312294	0.0225054	0.35957		0.35957		0.35957	
	2	Update	0.646835	0	0.3348		-1.38421		0.3348		0.3348		0.35957	-0.042365	0.35957	-0.042365	0.35957	-0.042365
17	1	Update			0.3348	0.0217929	-1.38421	-0.12529	0.3348	0.0217929	0.3348	0.0217929	0.317205		0.317205		0.317205	
	2	Update	0.641574	0	0.356593		-1.5095		0.356593		0.356593		0.317205	-0.038250	0.317205	-0.038250	0.317205	-0.038250
18	1	Update			0.356593	0.020987	-1.5095	-0.137271	0.356593	0.020987	0.356593	0.020987	0.278954		0.278954		0.278954	
	2	Update	0.636018	0	0.377579		-1.64677		0.377579		0.377579		0.278954	-0.034127	0.278954	-0.034127	0.278954	-0.034127
19	1	Update			0.377579	0.0201643	-1.64677	-0.149978	0.377579	0.0201643	0.377579	0.0201643	0.244827		0.244827		0.244827	
	2	Update	0.630172	0	0.397744		-1.79674		0.397744		0.397744		0.244827	-0.030213	0.244827	-0.030213	0.244827	-0.030213
20	1	Update			0.397744	0.01936	-1.79674	-0.163295	0.397744	0.01936	0.397744	0.01936	0.214613		0.214613		0.214613	
	2	Update	0.624062	0	0.417104		-1.96004		0.417104		0.417104		0.214613	-0.026607	0.214613	-0.026607	0.214613	-0.026607
21	1	Update			0.417104	0.01859	-1.96004	-0.17721	0.417104	0.01859	0.417104	0.01859	0.188006		0.188006		0.188006	
	2	Update	0.61771	0	0.435694		-2.13725		0.435694		0.435694		0.188006	-0.023344	0.188006	-0.023344	0.188006	-0.023344
22	1	Update			0.435694	0.0178613	-2.13725	-0.191749	0.435694	0.0178613	0.435694	0.0178613	0.164662		0.164662		0.164662	
	2	Update	0.611133	0	0.453555		-2.329		0.453555		0.453555		0.164662	-0.020429	0.164662	-0.020429	0.164662	-0.020429
23	1	Update			0.453555	0.0171761	-2.329	-0.206949	0.453555	0.0171761	0.453555	0.0171761	0.144232		0.144232		0.144232	
	2	Update	0.604334	0	0.470731		-2.53595		0.470731		0.470731		0.144232	-0.017847	0.144232	-0.017847	0.144232	-0.017847
24	1	Update			0.470731	0.0165339	-2.53595	-0.222851	0.470731	0.0165339	0.470731	0.0165339	0.126385		0.126385		0.126385	
	2	Update	0.597306	0	0.487265		-2.7588		0.487265		0.487265		0.126385	-0.015574	0.126385	-0.015574	0.126385	-0.015574
25	1	Update			0.487265	0.015933	-2.7588	-0.239497	0.487265	0.015933	0.487265	0.015933	0.110811		0.110811		0.110811	
	2	Update	0.590023	0	0.503198		-2.9983		0.503198		0.503198		0.110811	-0.013583	0.110811	-0.013583	0.110811	-0.013583
26	1	Update			0.503198	0.0153712	-2.9983	-0.256929	0.503198	0.0153712	0.503198	0.0153712	0.0972272		0.0972272		0.0972272	
	2	Update	0.582442	0	0.518569		-3.25523		0.518569		0.518569		0.0972272	-0.011845	0.0972272	-0.011845	0.0972272	-0.011845
27	1	Update			0.518569	0.0148457	-3.25523	-0.275187	0.518569	0.0148457	0.518569	0.0148457	0.085382		0.085382		0.085382	
	2	Update	0.57449	0	0.533415		-3.53041		0.533415		0.533415		0.085382	-0.010330	0.085382	-0.010330	0.085382	-0.010330
28	1	Update			0.533415	0.0143538	-3.53041	-0.294312	0.533415	0.0143538	0.533415	0.0143538	0.0750515		0.0750515		0.0750515	
	2	Update	0.566049	0	0.547769		-3.82472		0.547769		0.547769		0.0750515	-0.009012	0.0750515	-0.009012	0.0750515	-0.009012
29	1	Update			0.547769	0.013893	-3.82472	-0.314342	0.547769	0.013893	0.547769	0.013893	0.0660387		0.0660387		0.0660387	
	2	Update	0.556917	0	0.561662		-4.13907		0.561662		0.561662		0.0660387	-0.007867	0.0660387	-0.007867	0.0660387	-0.007867
30	1	Update			0.561662	0.0134608	-4.13907	-0.335318	0.561662	0.0134608	0.561662	0.0134608	0.0581709		0.0581709		0.0581709	
	2	Update	0.546708	0	0.575123		-4.47438		0.575123		0.575123		0.0581709	-0.006873	0.0581709	-0.006873	0.0581709	-0.006873

Jak widać z powyższych przykładów rozwiązanie z zastosowaniem sinusoidalnej funkcji aktywacji efektora dało w tym wypadku dużo lepsze i szybsze rozwiązanie dla rozważanego problemu XOR. W przypadku zastosowania neuronu ukrytego zbieżność przebiega wolniej.

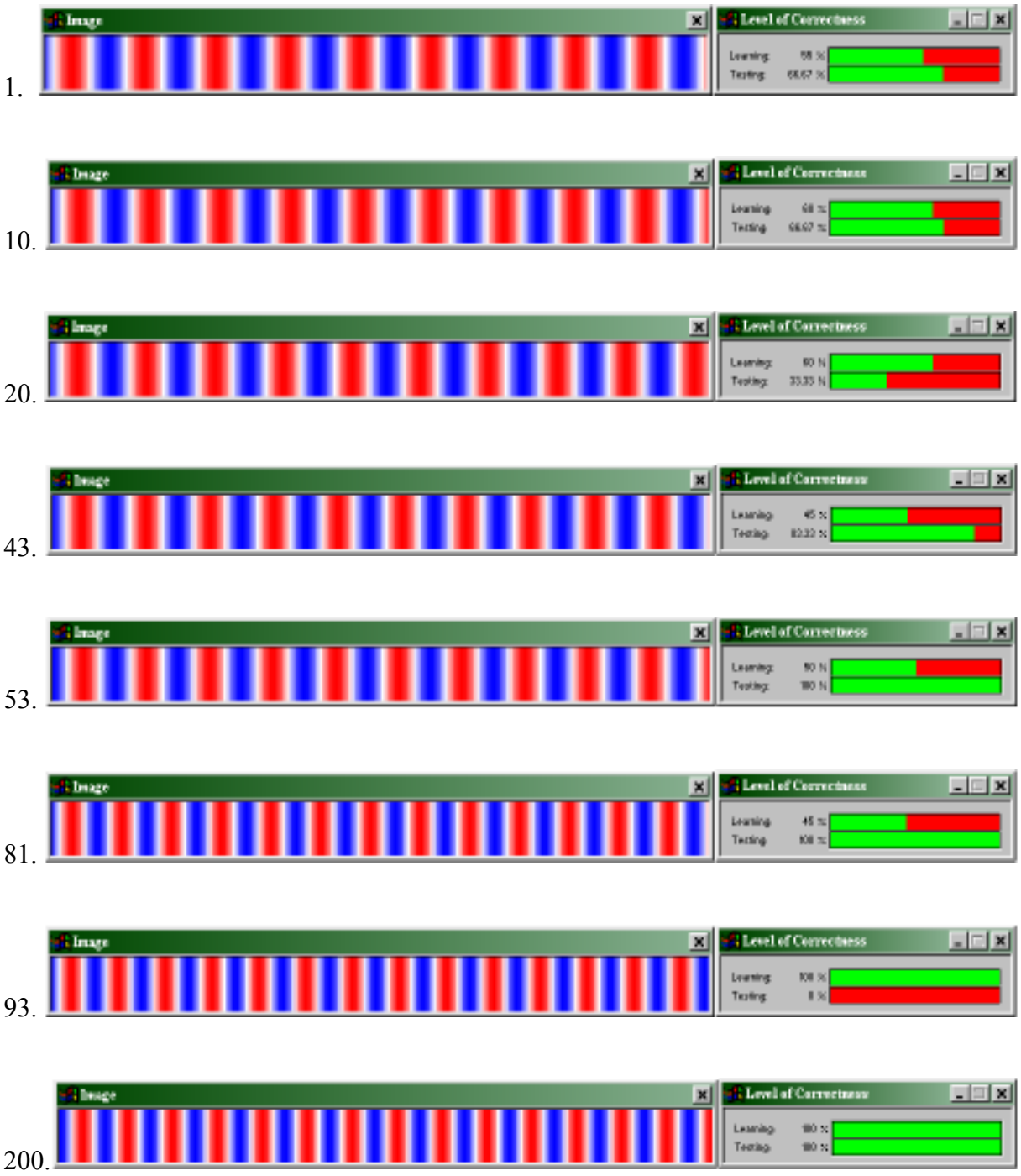
Metodę przetestowano również na wielu innych funkcjach logicznych. Przytaczania wyników takich symulacji jednak nie miałoby głębszego sensu, albowiem wyniki działania metody sterowanych kompromisów są analogiczne, jak dla opisanych w tym rozdziale reprezentatywnych funkcji logicznych OR i XOR.

5.3. Problem parzystości.

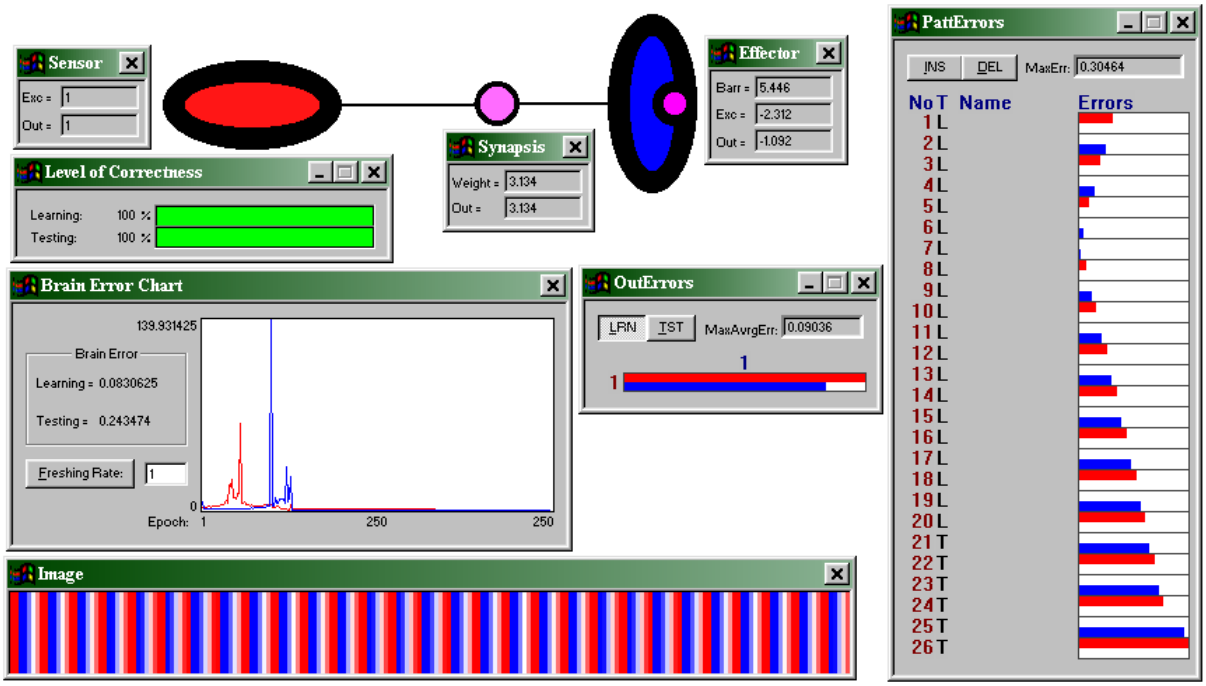
Problem parzystości, polegający na poprawnej klasyfikacji liczb parzystych i nieparzystych, jest często używanym benchmarkiem do testów, sprawdzających algorytmy uczące. W przypadku metody sterowanych kompromisów, która umożliwia stosowanie funkcji periodycznych jako funkcji aktywacji neuronów i efektorów, można było zastosować wprost trywialną architekturę sieci neuronowej (rysunek 5.9.) do rozwiązania tego problemu. Natomiast sam algorytm uczący musiał się w tym wypadku uporać z przeskalowaniem i przesunięciem danych do okresu zastosowanej funkcji periodycznej. Działanie metody przedstawiono (rys. 5.9.-12.) dla dwóch funkcji aktywacji efektora: sinusoidalnej oraz hybrydowej tangensoidalnej (por. rozdz. 4.4.2.). W przypadku nauki tego problemu posłużono się bardziej zaawansowanym sposobem obliczania kompromisu wykorzystującym mechanizm przełączników.



Rysunek 5.9. Powyższy rysunek przedstawia przebieg nauki sieci neuronowej metodą sterowanych kompromisów dla problemu parzystości. Jako funkcję aktywacji efektora zastosowano funkcję sinusoidalną (por. rozdz. 4.4.2.). Z powyższego wykresu błędu sieci neuronowej wynika, że nauka przebiega bardzo burzliwie. Fakt ten związany jest w tym wypadku z zastosowania bardziej zaawansowanego mechanizmu obliczania kompromisów oraz z konieczności przeskalowania i przesunięcia za pomocą parametrów wolnych sieci tak, żeby liczby parzyste odpowiadały wzgórkom a liczby nieparzyste dolinom funkcji sinus. Proces przeskalowania i przesunięcia objawia się w tym wypadku zagęszczaniem się kolorowych paseczków, jak to jest pokazane na rys. 5.10.



Rysunek 5.10. Na rysunku można zaobserwować proces przesuwania i zagęszczania się kolorowych paseczków (czerwone odpowiadają liczbom parzystym, niebieskie liczbom nieparzystym) związany z procesem formowania się parametrów sieci w procesie jej uczenia się dla przykładu z rysunku 5.9. Dla uzupełnienia z lewej strony obrazów podano odpowiadającą epokę uczącą, zaś z lewej procent poprawności klasyfikacji liczb parzystych i nieparzystych odpowiednio dla wzorców uczących i testujących.



Rysunek 5.11. Powyższy rysunek przedstawia przebieg nauki sieci neuronowej metodą sterowanych kompromisów dla problemu parzystości. W tym przypadku jako funkcję aktywacji efektora zastosowano hybrydową funkcję tangensoidalną (por. rozdz. 4.4.2.). Z powyższego wykresu błędu sieci neuronowej wynika, że nauka przebiega również bardzo burzliwie. Duże skoki błędu spowodowane są tym, że zbiór wartości zastosowanej hybrydowej funkcji tangensoidalnej leży w dziedzinie liczb rzeczywistych, a podczas procesu skalowania dla niektórych wzorców wartości funkcji mogą trafiać w bardzo duże lub bardzo małe liczby. Jak widać taki proces zmian nie dezorientuje metody sterowanych kompromisu w jej dążeniu do poprawnego rozwiązania.

1. 

20. 

30. 

52. 

58. 

61. 

65. 

110. 

Rysunek 5.12. Na rysunku można zaobserwować proces przesuwania i zagęszczania się kolorowych paseczków (czerwone odpowiadają liczbom parzystym, niebieskie liczbom nieparzystym) związany z procesem formowania się parametrów sieci w procesie jej uczenia się dla przykładu z rysunku 5.11. Z lewej strony obrazów podano odpowiadającą epokę uczącą, zaś z lewej procent poprawności klasyfikacji liczb parzystych i nieparzystych odpowiednio dla wzorców uczących i testujących, które w tym wypadku świadczą o poprawnej ekstrapolacji.

5.4. Problem dwóch spiral.

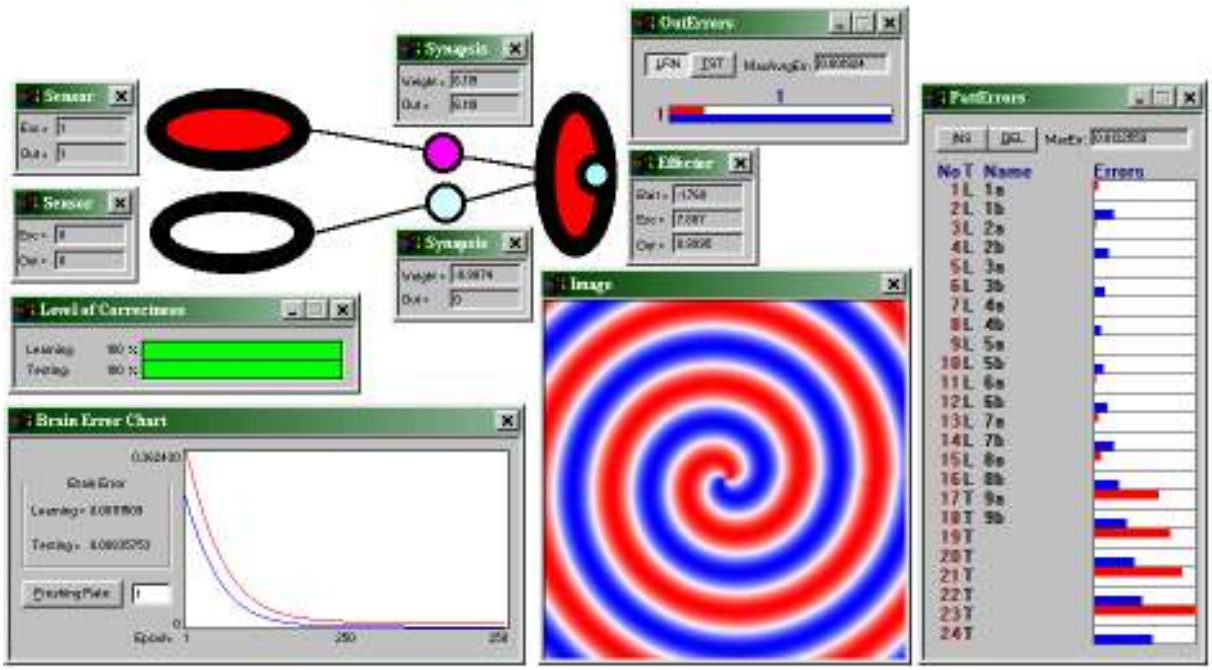
Problem dwóch spiral jest klasycznym przykładem trudnego benchmarku, używanego do testowania zawansowanych algorytmów uczących sieci neuronowych. Trudność tego problemu polega na osiągnięciu poprawnej ekstrapolacji, co w przypadku zastosowania klasycznych funkcji sigmoidalnych osiągalne jest dla nieskończonej ilości neuronów sieci oraz nieskończenie długiego ciągu uczącego umożliwiającego nauczenie takiej sieci (.....dopisać odnośniki). Obydwa te warunki są oczywiście z praktycznego punktu widzenia nie do zrealizowania. Problem dwóch spiral ma naturę periodyczną, a metoda sterowanych kompromisów umożliwia uczenie sieci opartych o periodyczne funkcje aktywacji. Ten fakt sprawia, że problem ten może być rozwiązany przy użyciu wprost trywialnej architektury sieci neuronowej, jak widać to na rysunku 5.13. W przypadku niedużych odległości stanu inicjalnego sieci neuronowej od rozwiązania, metoda sterowanych kompromisów bez trudu prowadzi proces nauki w kierunku tego rozwiązania. W przypadku spiral liniowych, jakie są w tym przykładzie rozważane, istnieje nieskończenie wiele rozwiązań dla przedstawionej na rysunku 5.13. architektury sieci neuronowej. Rozwiązania te można opisać następująco:

$$\tau_1 = \frac{-\frac{\pi}{2} + 2k\pi}{\beta}, \quad w_2 = \frac{-1}{\beta}, \qquad w_3 = \frac{2\pi}{\beta}$$

oraz

$$\tau_1 = \frac{-\frac{\pi}{2} + 2k\pi}{\beta}, \quad w_2 = \frac{1}{\beta}, \qquad w_3 = \frac{-2\pi}{\beta}.$$

gdzie β jest parametrem stromości funkcji aktywacji, który zazwyczaj nie zmieniany w trakcie procesu uczenia, lecz wybierany *a priori*. W następujących przykładach jego wartość jest stała i równa 1. Jak widać z opisu rozwiązań dla ustalonego parametru β wagi mogą przyjmować tylko jedną z dwu możliwych wartości, zaś próg posiada nieskończoną ilość poprawnych wartości. Stąd można wnioskować, że najtrudniejszym zadaniem dla algorytmu uczącego jest znalezienie wartości wag. Zadanie to nie jest łatwe, ze względu na to, że stosowana tutaj periodyczna funkcja aktywacji efektora posiada automatycznie nieskończoną ilość minimów lokalnych, które mogą przeszkadzać w procesie zbieżności metody do rozwiązania zadanego ciągiem uczącym i w procesie minimalizacji błędu sieci.



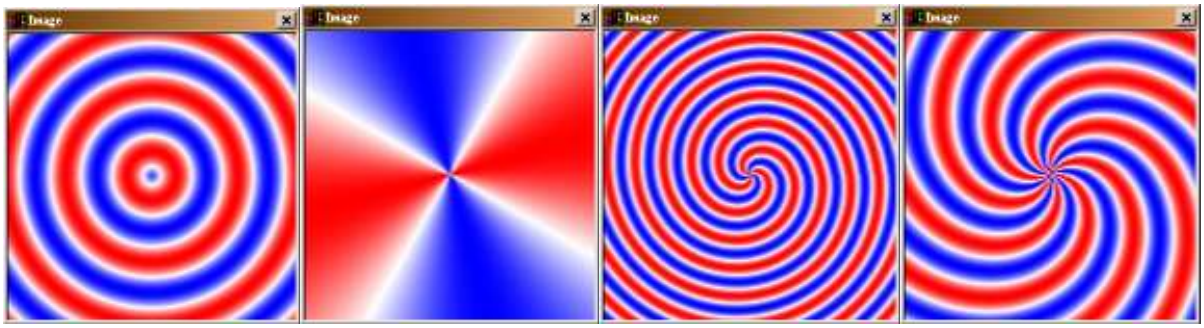
Rysunek 5.13. Na rysunku przedstawiono naukę sieci neuronowej metodą sterowanych kompromisów dla problemu dwóch liniowych spiral. Zastosowano sinusoidalną funkcję aktywacji efektora (por. rozdz. 4.4.2.). Z powyższego wykresu błędu sieci neuronowej wynika, że proces zbieżności metody przebiega prawidłowo przy założeniu, że sieć została zainicjowana w niezbyt dużym oddaleniu od jednego z możliwych rozwiązań. Dla uzupełnienia na rysunku 5.14. uwidoczniono graficznie, jak przebiega proces zbiegania się spiral.

1. 

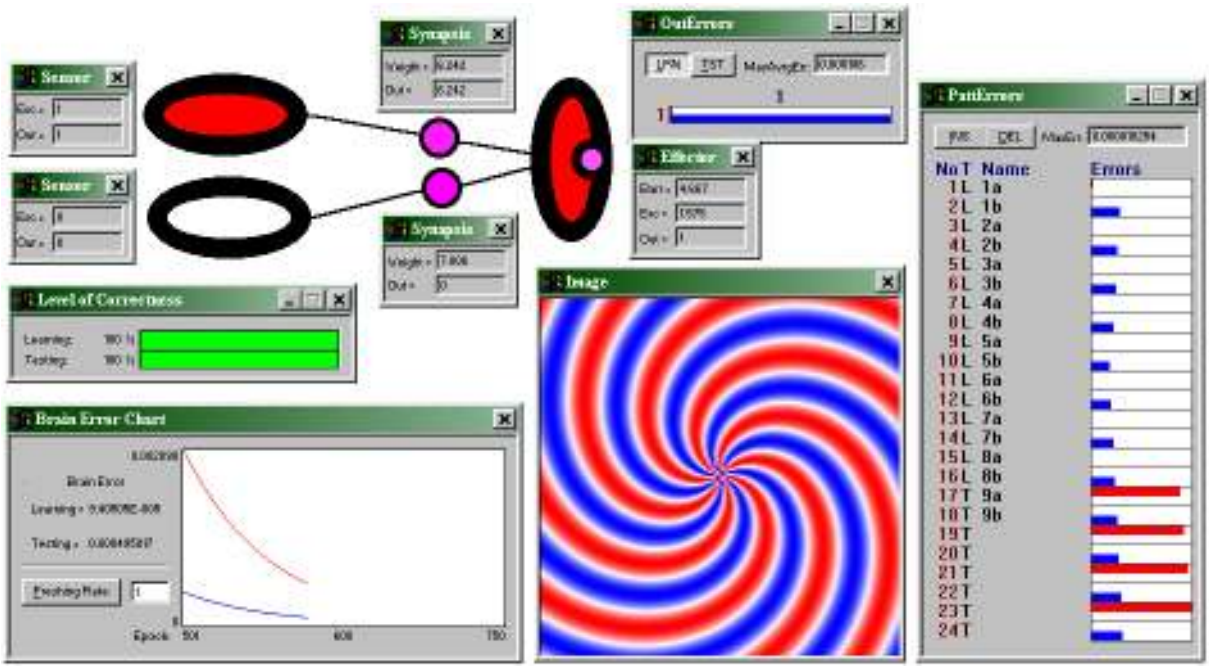
30. 

150. 

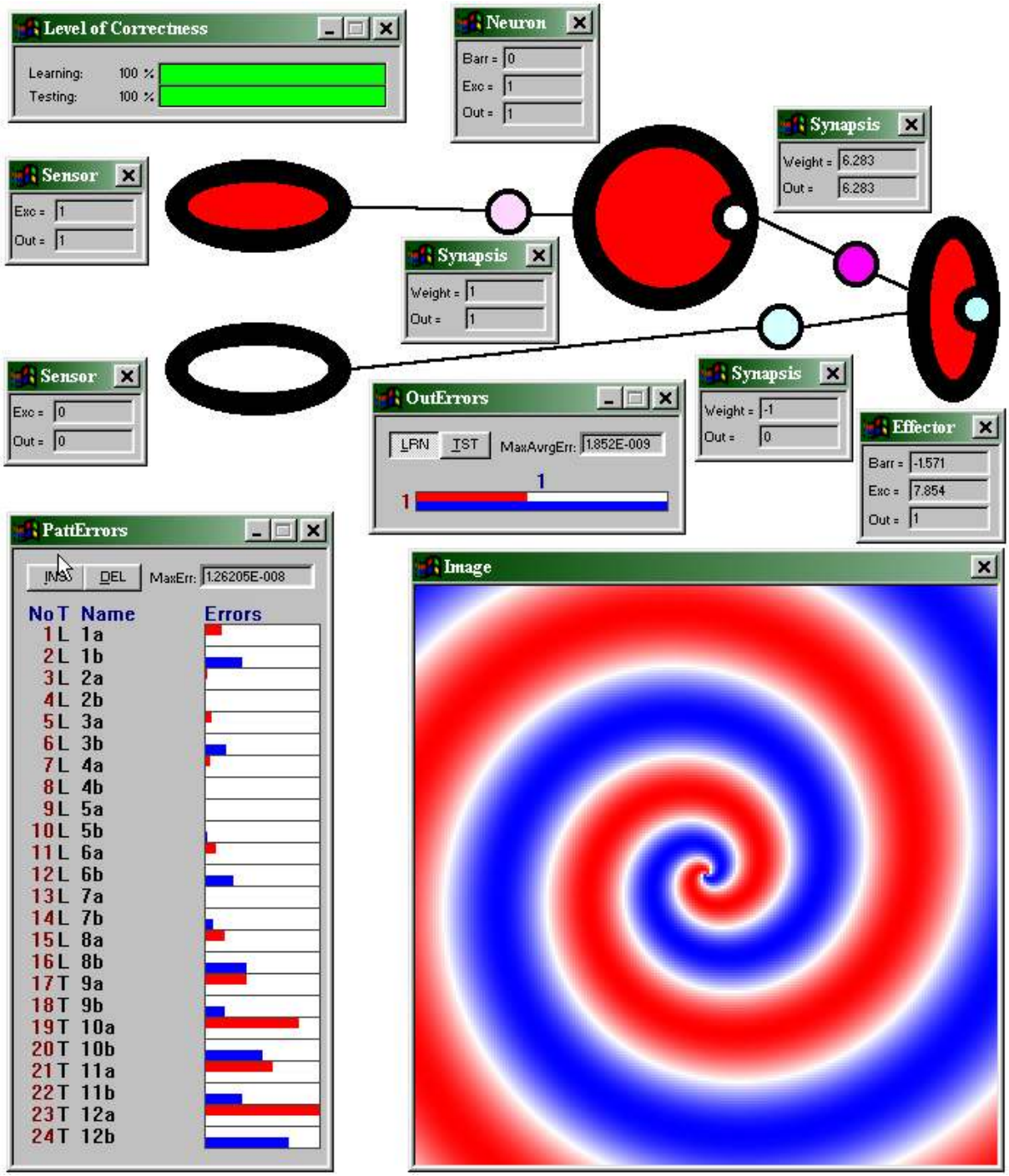
Rysunek 5.14. Rysunek przedstawia proces zbiegania się spiral dla wybranych epok uczących: 1, 30, 150. Na początku spirale zupełnie nie nawiązują na siebie. Po 30. epoce widoczny jest już proces formowania się dwóch spiral. W 150. epoce spirale są już prawie idealnie uformowane w wyniku procesu uczącego metodą sterowanych kompromisów.



Rysunek 5.15. Powyższe rysunki ilustrują stany sieci neuronowej, powstałe w wyniku zatrzymania się procesu uczenia w jednym z minimów lokalnych.



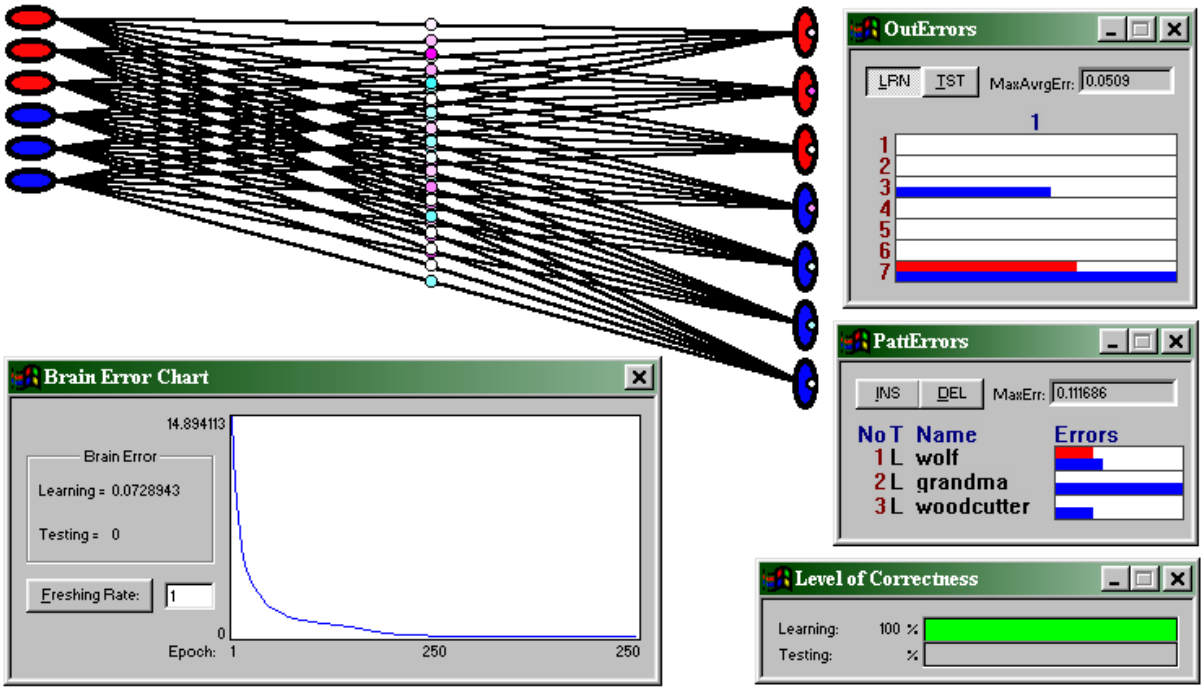
Rysunek 5.16. Rysunek przedstawia ciekawy przykład „uczenia się na pamięć” – sieć uzyskuje dobre wyniki dla konkretnego ciągu uczącego (tutaj również dla ciągu testowego), ale w istocie nie wykrywa rzeczywistej natury badanego problemu. Winę za taki stan rzeczy nie ponosi jednak algorytm uczący, który prawie idealnie (błąd uczenia jest rzędu 9.4E-5) dostosował sieć do zadanego ciągu uczącego. Aby nie dopuścić do takiego stanu, należałoby w tym wypadku rozbudować ciąg uczący i uczynić go bardziej reprezentatywny dla danego problemu.



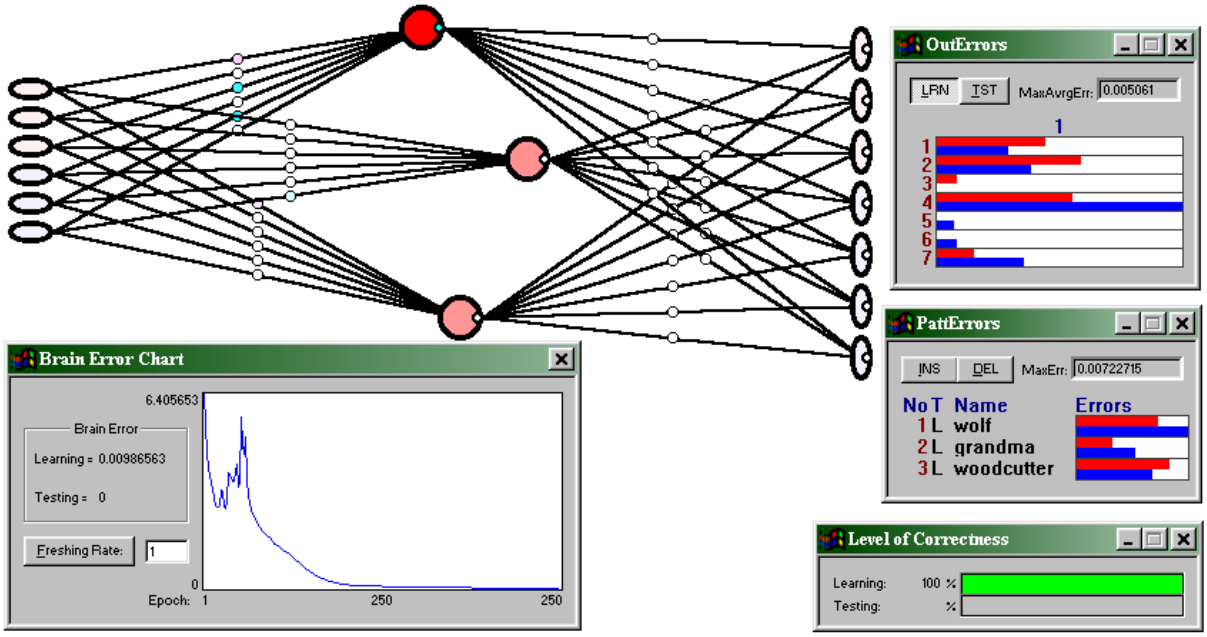
Rysunek 5.17. Na rysunku pokazano, że potencjalne możliwości rozbudowy sieci opartych o różnorodne funkcje aktywacji neuronów i metodę sterowanych kompromisów są duże. Na przykład problem dwóch liniowych spiral może zostać rozszerzony tak, by posłużył do rozwiązywania problemu nieliniowych spiral, co jest zadaniem dużo bardziej skomplikowanym. W tym przypadku konieczne jest jednak usprawnienie metody sterowanych kompromisów tak, by efektywnie potrafiła radzić sobie z uczeniem sieci wielowarstwowych.

5.5. Problemu czerwonego kapturka.

Problem czerwonego kapturka (LRRH – Little Red Ridinghood [31]), który ma za zadanie rozróżnić wilka, drwala oraz babcię na podstawie wyglądu i zachowania się (czy ma duże uszy, duże oczy, duże zęby, zmarszczki, czy jest przyjacielski, przystojny). Na tej podstawie ma podjąć decyzję o swoim dalszym zachowaniu (uciekać, krzyczeć o pomoc, szukać drwala, pocałować w czoło, zbliżyć się, zaoferować jedzenie, poflirtować). Uczenie sieci neuronowej dla tego problemu przy pomocy metody sterowanych kompromisów zrealizowano dla sieci dwuwarstwowej (rys. 5.18.) oraz trójwarstwowej posiadającej 3 neurony ukryte (rys. 5.19.).



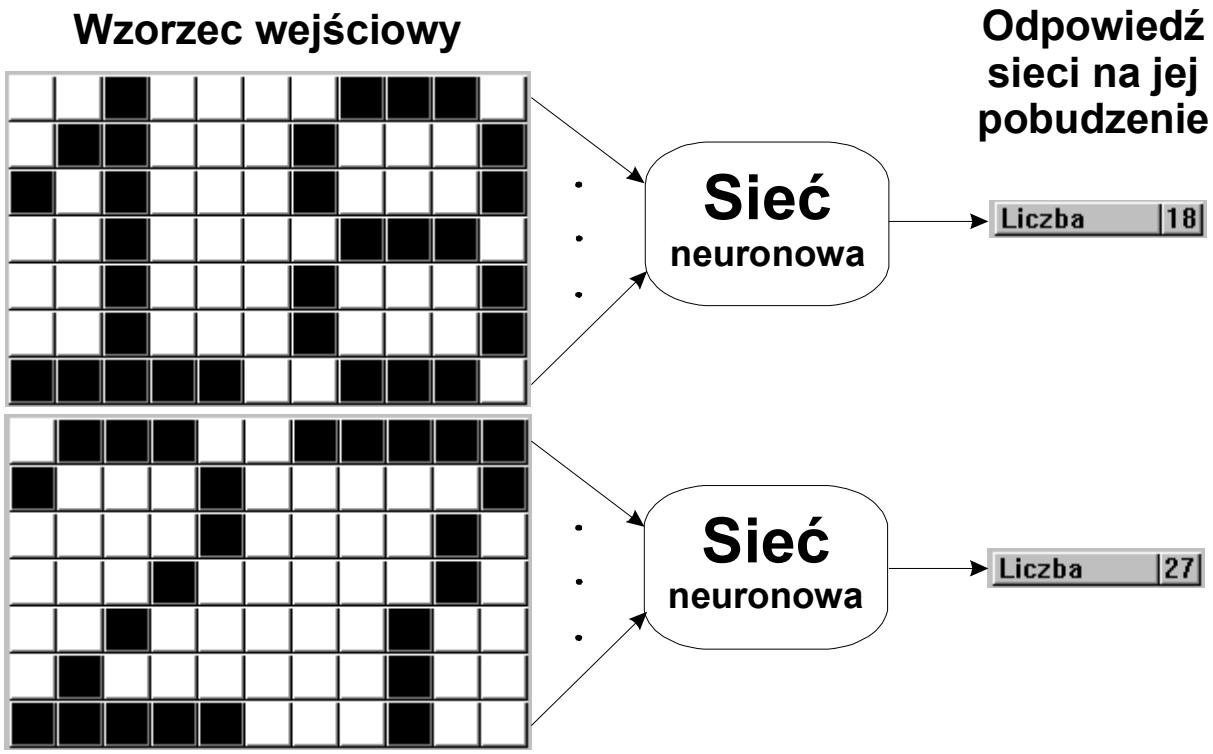
Rysunek 5.18. Rysunek przedstawia uwieńczony sukcesem proces uczenia problemu czerwonego kapturka przy pomocy metody sterowanych kompromisów. W wyniku nauki sieć neuronowa (tj. czerwony kapturek) nauczyła się prawidłowo podejmować decyzję o dalszym postępowaniu na podstawie wyglądu i zachowania postaci, którą spotkała.



Rysunek 5.19. Na rysunku pokazano naukę problemu czerwonego kapturka zaaplikowanego na sieci trójwarstwowej posiadającej 3 neurony ukryte przy pomocy metody sterowanych kompromisów. Również w tym wypadku nauka zakończyła się bardzo szybko sukcesem i sieć nauczyła się prawidłowych reakcji.

5.6. Problem rozpoznawania obrazów.

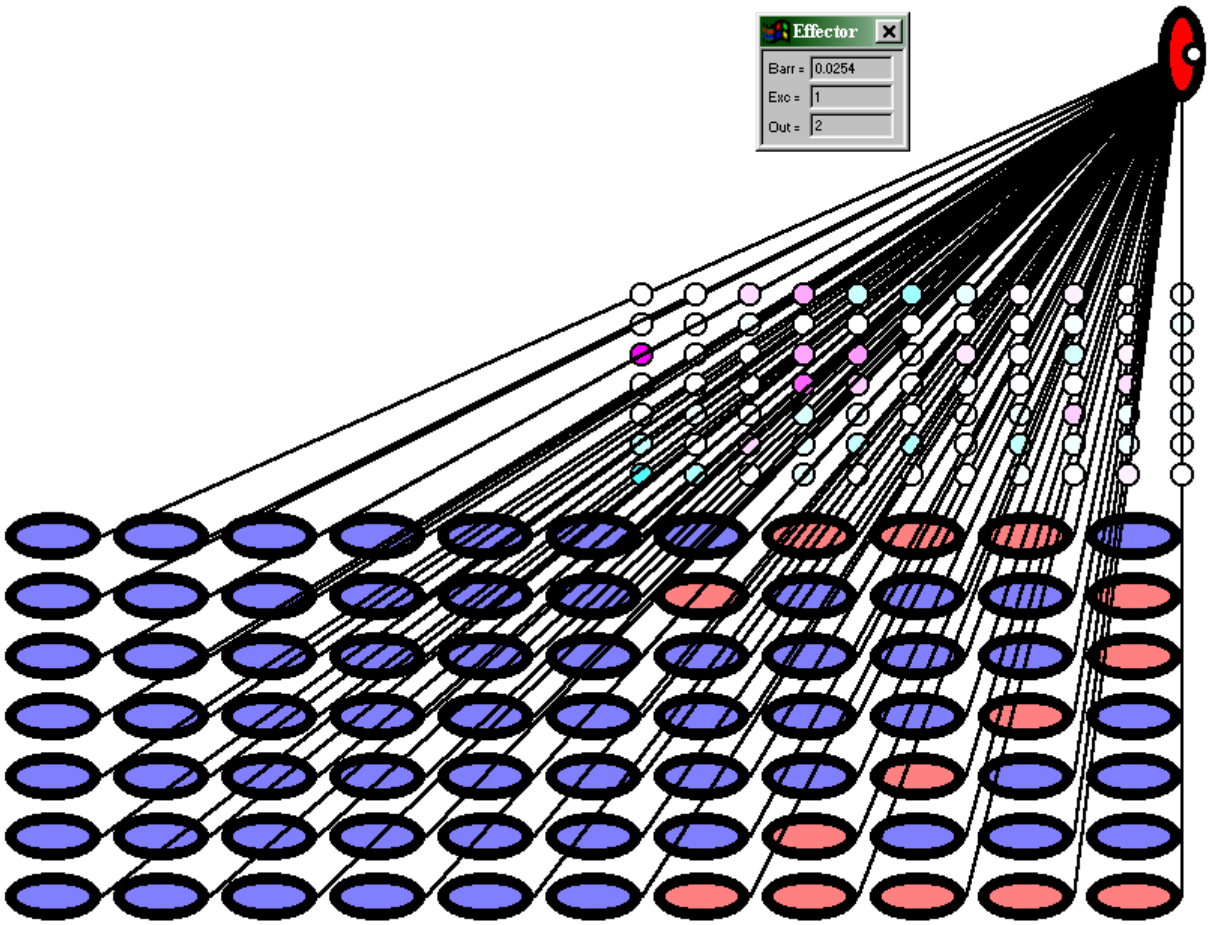
Celem tego przykładu jest przedstawienie możliwości zastosowania metody sterowanych kompromisów do problemu rozpoznawania obrazów. Dla celów demonstracyjnych wybrano problem rozpoznawania dwucyfrowych liczb z zakresu od 0 do 99 przedstawianych sieci w postaci matrycy włączonych i wyłączonych punktów tak, jak to przedstawiają rysunki 5.20.-22. dla przykładowych liczb: 18, 27 i 2 .



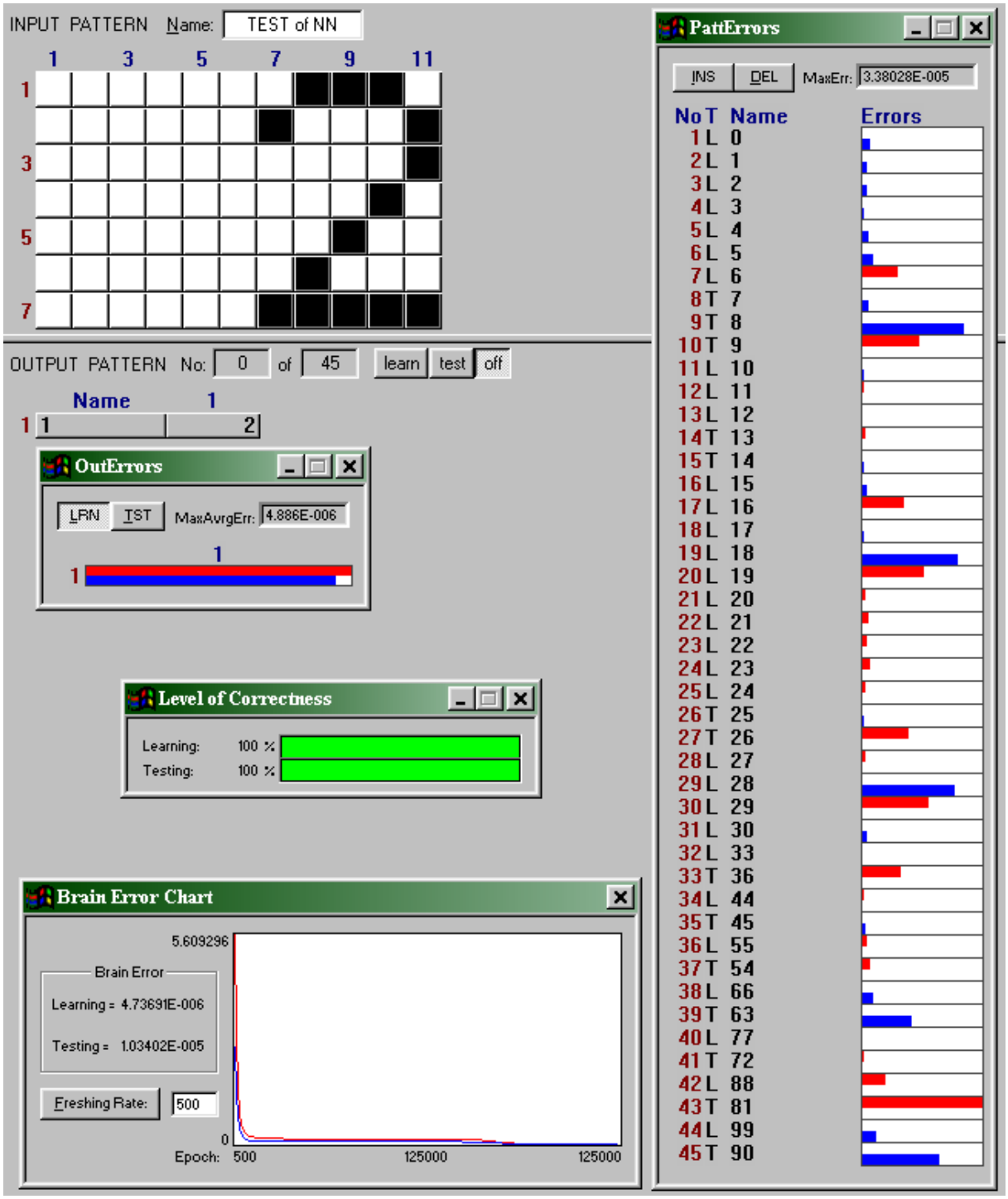
Rysunek 5.20. Na rysunku pokazano dwa wybrane wzorce uczące (liczby 18 i 27) oraz pożądane odpowiedzi, jakie nauczona sieć neuronowa ma dać dla nich na swoim wyjściu.

W przykładzie tym skupiono się nad zademonstrowaniem zdolności sieci neuronowej do uogólniania zdobytej wiedzy w procesie uczącym przeprowadzonym metodą sterowanych kompromisów. Mianowicie, jako ciąg uczący wybrano 30 reprezentatywnych liczb (0, 1, 2, 3, 4, 5, 6, 10, 11, 12, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 27, 28, 29, 30, 33, 44, 55, 66, 77, 88, 99) z zakresu o 0 do 99. Następnie stosując metodę sterowanych kompromisów starano się w procesie uczącym nakłonić sieć do prawidłowego rozpoznawania tak zbudowanego ciągu uczącego, jak to pokazano schematycznie na rysunku 5.20.

Dla tak zdefiniowanego ciągu uczącego posłużono się siecią neuronową składającą się z sensorów odpowiadającym poszczególnym polom matrycy liczbowej oraz jednego efektora, na którego wyjściu dla nauczonej sieci neuronowej ma się pojawić liczba odpowiadająca tej podanej w postaci matrycy na wejście sieci (rys. 5.21.).



Rysunek 5.21. Rysunek przedstawia sieć neuronową, która została poddana nauce metodą sterowanych kompromisów, której celem było rozpoznawanie dwucyfrowych matryc liczbowych przedstawiających liczby z zakresu od 0 do 99. Okienko dynamicznego podglądu efektora pokazuje, że nauczona sieć na pobudzenie matrycą przedstawiającą liczbę 2 dała na swoim wyjściu odpowiedź (Out) w postaci liczby 2.



Rysunek 5.22. Na rysunku pokazano przebieg nauki sieci z rysunku 5.21. dla problemu rozpoznawania dwucyfrowych matryc liczbowych metodą sterowanych kompromisów. W okienku błędu sieci widać, iż po wykonaniu 125000 epoki uczącej sieć neuronowa wykazuje błąd uczenia równy 4,73E-6 oraz błąd dla wzorców testujących równy 1,03E-5. Z prawej strony zaobserwować można wielkości błędów dla poszczególnych wzorców uczących (oznaczonych literką L) oraz testujących (oznaczonych literką T).

Z przytoczonych rysunków (5.20.-22.) widać, że sieć neuronowa prawidłowo nauczyła się postawionego jej problemu przy pomocy ciągu uczącego metodą sterowanych kompromisów. W wyniku nauki uzyskaliśmy w pełni skonfigurowaną sieć neuronową zdolną nie tylko prawidłowo i dokładnie (z dokładnością średnio 4,73E-6) rozpoznawać wzorce uczące, ale również zdolną prawidłowo uogólnić zdobytą wiedzę i całkowicie poprawnie (z dokładnością średnio 1E-5) rozpoznawać wszystkie wzorce testujące. Wynik ten został osiągnięty po wykonaniu 125000 epok, lecz należy zaznaczyć, że już po około 45000 epok sieć neuronowa prawidłowo rozpoznawała wszystkie przedstawione jej wzorce uczące oraz testujące z tym, że z mniejszą dokładnością.

5.7. Dyskusja zbiorcza wyników testowego stosowania metody sterowanych kompromisów.

.....

6. Podsumowanie

W pracy zaprezentowano dwie nowe metody umożliwiające ustalenie parametrów sieci neuronowej na bazie jej nauki bądź konfiguracji na podstawie zadanego ciągu uczącego.

W pierwszej metodzie (opisanej w rozdziale 3.) w oryginalny sposób podjęto udaną próbę automatycznej konfiguracji zarówno architektury jak i parametrów sieci neuronowej przeznaczonej dla problemów rozpoznawania wzorców binarnych. W wyniku działania opisanej w pracy metody uzyskuje się kompletnie skonfigurowaną jednowarstwową sieć neuronową o liniowych funkcjach aktywacji oraz obliczone parametry wolne sieci, na które składają się wagi synaptyczne. Proces konfiguracji struktury i parametrów sieci przebiega w tym przypadku podczas zaledwie dwukrotnego przeglądnięcia ciągu uczącego, co sprawia, że opisana metoda pod względem szybkości działania jest na swój sposób bezkonkurencyjna w porównaniu do tradycyjnych metod uczących sieci tradycyjną techniką korygowania błędów. Zalety przedstawionej metody są jednak okupione ograniczonym zakresem jej stosowalności. Immanentną cechą tej metody jest bowiem okoliczność, że jak na razie pozwala ona w efektywny sposób rozwiązywać tylko problemy, które możliwe są do zakodowania w formie binarnej (takiej postaci danych wejściowych i wyjściowych wymaga bowiem pierwsza rozpatrywana w pracy sieć) i są dodatkowo wolne od problemów związanych z nie kontrolowaną zmianą dyslokacji danych uczących w wejściowej strukturze danych. Oznacza to, że dla szeregu zadań praktycznych użycie opracowanej przez autora sieci uzależnione jest od możliwości zastosowania w tych problemach odpowiednich procedur wstępnego przetwarzania danych – polegających na odpowiednim skalowaniu (binaryzacja danych) oraz na ich normalizacji (zapobieganie dyslokacjom). Dodatkowo stosując tę pierwszą opracowaną w ramach pracy metodę uczenia trzeba wziąć pod uwagę, iż zakłada ona pełną odpowiedniość odwzorowania danych do ilość klas, co sprawia, że metoda nie nadaje się do klasyfikacji i rozpoznawania wzorców w przypadku dużych baz danych.

Mimo tych ograniczeń koncepcja sieci odwzorowującej wzorce binarne wydaje się interesującym uzupełnieniem w stosunku do wszystkich wcześniej znanych metod konfigurowania i uczenia sieci neuronowych. Zaprezentowana metoda pozwala bowiem wyjątkowo sprawnie optymalizować architekturę sieci neuronowej dla zadanego ciągu uczącego, co stanowi jej cechę szczególną, zdecydowanie odróżniającą przedstawione w pracy podejście od typowych aplikacji sieci neuronowych, gdzie architekturę sieci wybiera się zwykle arbitralnie lub dopasowuje do cech zadania *ex post* za pomocą różnych

pracochłonnych procedur (np. pruning). W opisanej metodzie konfiguracja sieci powstaje jak gdyby „na zamówienie” w oparciu o dane uczące, co stanowi dużą zaletę proponowanej metody. Nie są to jedyne zalety zaproponowanego podejścia. Przy użyciu przedstawionej w pracy metody możliwa jest również ocena jakości stworzonej sieci, a zwłaszcza jakości uzyskanego uogólnienia. W rozdziale 3. pracy zdefiniowane zostały takie pojęcia, jak jakość uzyskanego rozpoznawania oraz jakość uzyskanego uogólnienia, które umożliwiają ocenę jakości uzyskanego rozwiązania *ex post*, ale pozwalają także odpowiednio sterować procesem redukcji synaps dla osiągnięcia struktury sieci o pożądanej jakości działania i możliwie najmniejszej (najmniej kosztownej) strukturze. Wszystkie uzyskane wyniki symulacyjne (częściowo przedstawione w pracy, ale w rzeczywistości prowadzone w znacznie szerszym zakresie) potwierdzają słuszność założeń przedstawionej metody, jak również potwierdzają zreferowane w pracy oraz w skrócie zasygnalizowane wyżej zalety i wady opisanej metody.

Jak wynika z przedstawionych uwag, opisana w rozdziale 3. rozprawy oryginalna metoda uczenia i konfigurowania jednowarstwowych sieci linowych klasyfikujących sygnały binarne, wykazywała pewne zalety (szczególnie wyrażające się w szybkości działania), miała jednak także zdecydowane wady i uciążliwe ograniczenia. Analiza natury tych wad i powodów ograniczeń doprowadziła do zaproponowania w pracy kolejnej oryginalnej metody uczenia i konfigurowania sieci neuronowych mogących rozpoznawać sygnały wejściowe przyjmujące wartości ze zbioru liczb rzeczywistych, a jednocześnie algorytm klasyfikacji mógł być bardziej wyrafinowany, ze względu na możliwość użycia w tym przypadku sieci zawierających zarówno elementy liniowe, jak i nieliniowe. Metoda ta, nazwana (ze względu na sposób działania) metodą sterowanych kompromisów została opisana w rozdziale 4. pracy. Jest ona metodą bardziej ogólną od wyżej dyskutowanej metody tworzenia sieci do klasyfikacji sygnałów binarnych., Sieci uczone metodą sterowanych kompromisów powinny pozwolić rozwiązywać bardzo szeroką grupę problemów, chociaż ze względów czasowych w ramach tej pracy nie udało się wyeksploatować wszystkich ich potencjalnych możliwości.

W obecnej postaci metoda sterowanych kompromisów umożliwia efektywne uczenie sieci jednowarstwowych. Pracuje na ciągłych danych uczących, nie wymuszając ich sztucznego skalowania, ponieważ pozwala dokonywać transformacji z i do zbioru liczb rzeczywistych. Własność ta wynika z faktu, iż funkcje aktywacji neuronów są w tej metodzie dobierane w sposób niestandardowy. W rozdziale 4. pracy pokazano, jak można dobierać nieliniowe funkcje przejścia neuronów w taki sposób, aby poprawić właściwości ekstrapolacyjne uzyskanych sieci i „zmusić” je do uogólnień wiedzy wydobywanej z ciągu uczącego. W pracy

zdefiniowane zostały również podstawy matematyczne zaproponowanej metody, pozwalające wykorzystać jako funkcje aktywacji neuronów prawie dowolne funkcje (nawet okresowe), podlegające tylko nieznacznym ograniczeniom. Można oczekiwać, że dzięki temu możliwe będzie w wybranych zadaniach bardziej oszczędne konstruowanie architektur sieci neuronowych o charakterystykach nieliniowych dopasowywanych do natury rozwiązywanego zadania. Warto podkreślić, że podejście takie znacząco odróżnia metodę sterowanych kompromisów, opracowaną i przebadaną w tej pracy, w stosunku do podejść tradycyjnych, w których dużo uwagi poświęca się np. kunsztownemu dopasowywaniu superpozycji „urwisk sigmoidalnych" albo funkcji dzwonokształtnych (dla sieci RBF) do kształtu aproksymowanych funkcji – podczas gdy ten sam cel można czasem osiągnąć bez porównania mniejszym wysiłkiem uwalniając nieliniowe funkcje przejścia używanych neuronów od ograniczeń narzucanych przez tradycję sztucznych sieci neuronowych. Właśnie taka próba opisana jest w tej pracy.

Ważnym punktem opracowanej w pracy metody sterowanych kompromisów jest zdefiniowanie w niej tzw. odchyleń od uzyskanego kompromisu, pozwalających bardziej inteligentnie sterować procesem nauki. Dzięki bieżącemu pomiarowi uzyskanego kompromisu opracowana metoda może sprawniej zmierzać do nadrzędnego celu, którym jest odnalezienie minimum globalnego funkcji błędu dla zadanej architektury sieci neuronowej i dla konkretnego ciągu uczącego. Mechanizm ten nie ma swojego bezpośredniego odpowiednika w żadnej znanej autorowi metodzie uczenia sieci neuronowych. W perspektywie czasu opisane odchylenia powinny pozwolić również na skonstruowanie metod automatycznej rekonfiguracji wieloznacznych obecnie części architektury tworzonej sieci, co umożliwiłoby w efekcie zbudowanie także dla metody sterowanych kompromisów modułów całkowicie automatyzujących proces konfiguracji architektury i uczenia sieci neuronowej - ściśle dla zadanego ciągiem uczącym problemu.

Warto jeszcze przytoczyć kilka uwag wskazujących na odmienności rozważanej w pracy metody w stosunku do tradycyjnych technik uczenia sieci. W metodzie sterowanych kompromisów zrezygnowano ze stosowania współczynnika uczenia, który jest ściśle związany z obecnie często stosowanymi metodami gradientowymi. Zaproponowana metoda nie wymaga jego istnienia, dzięki czemu proces uczenia wykonywany zgodnie z opisaną metodą sterowanych kompromisów nie jest niepotrzebnie zwalniany, gdy współczynnik uczenia ma małą wartość. Ponadto w porównaniu do metod gradientowych, zmierzających (jak wiadomo) często do minimów lokalnych (pod wpływem kierunku wyznaczanego przez

spadek gradientu), metoda sterowanych kompromisów jest zdolna kierować się do minimum globalnego.

Kolejnym udogodnieniem opisanej metody sterowanych kompromisów jest możliwość uniezależnienia w niej korekt błędów od reprezentacji liczbowej wzorców uczących tak, by wielkość ich reprezentacji liczbowej nie przekładała się na stopień ich wpływu na zawierany kompromis. Z tego powodu metoda sterowanych kompromisów może być traktowana (w odniesieniu do wzorców) jako metoda samonormalizująca. Udogodnienie to powoduje względne równouprawnienie wzorców uczących z punktu widzenia procesu uczenia.

Na zakończenie tej dyskusji warto dodać, że istotną zaletą zaprezentowanej metody sterowanych kompromisów jest możliwość wykorzystania w niej niepełnych danych wejściowych (niekompletnych przykładów uczących) dla potrzeb uczenia sieci. W wyniku tego możliwe jest na przykład podjęcie procesu uczenia sieci techniką sterowanych kompromisów również na podstawie danych doświadczalnych, które z jakichś powodów nie są lub nie mogą być zebrane w całości. Stanowi to ważną, wręcz trudną do przecenienia zaletę przedstawionej metody w kontekście wielu ważnych zastosowań sieci neuronowych do opracowywania danych empirycznych – zwłaszcza w biologii i medycynie.

Metoda sterowanych kompromisów została w tej pracy przedstawiona ze wszystkimi znanymi autorowi pracy jej zaletami i wadami. Ze względu na swą dużą rozpiętość podjętej problematyki pokazano w pracy tylko te jej części, które są zakończone i przynoszą już oczekiwane efekty. Wzmiankowano jednak również te zagadnienia, cechy i właściwości metody sterowanych kompromisów, które są problematyczne i stanowią przedmiot dalszych badań. W pracy rozważano na przykład (wstępnie) problem uogólnienia metody sterowanych kompromisów dla potrzeb uczenia sieci wielowarstwowych. Jednak w tym przypadku badania referowane w pracy nie przyniosły jeszcze finalnego rezultatu, konieczne więc jest jeszcze dopracowanie metody określającej w tym przypadku sposób zawierania kompromisu tak, by ten nie powodował problemów natury numerycznej, destabilizującej zbieżność metody. Zagadnienie to okazało się kłopotliwe w warstwie implementacyjnej (a nie merytorycznej!) i będzie wymagało wielu prac związanych z rozwojem towarzyszącego tej pracy oprogramowania *„Brain for Problem”*, ponieważ jednak rozwiązanie problemu samej tylko stabilności numerycznej obliczeń wnosiło to stosunkowo niewiele w obszarze naukowym, a było bardzo czasochłonne - pozostawiono w związku z tym odpowiednie problemy do rozwiązania już poza niniejszą pracą.

Literatura:

1. Anderson J. A., *An Introduction to Neural Networks*, The MIT Bradford Press, Cambridge, 1995.
2. Arbib M. A., Érdi P. and Szentágothai J., *Neural Organization (Structure Function and Dynamics)*, A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England, 1998 Massachusetts Institute of Technology.
3. Atmanspacher H., Ruhnau E. (red.), *Time, Temporality, Now*, Springer Verlag, Berlin, 1997.
4. Bartsch H. J., *Matematické vzorce*, SNTL – Nakladatelství technické literatury, Praha 1987.
5. Budohoska W., Grabowska A., *Dwie półkule – jeden mózg*, Wiedza Powszechna, Seria Omega, Warszawa 1994.
6. Davis R., Lenat D. B., *Knowledge-Based Systems in Artificial Intelligence*, McGraw-Hill, Inc., 1982.
7. Fiesler E., Beale R., *Handbook of Neural Computation*, IOP Publishing Ltd and Oxford University Press, Bristol & New York, 1997.
8. Forsyth R. and Rada R., *Machine Learning Applications in Expert Systems and Information Retrieval*, Ellis Horwood Limited Publishers, New York, 1986.
9. Freeman J. A. and Skapura D. M., *Neural Networks: Algorithms, Applications and Programming Techniques*, MA: Addison-Wesley, Reading, 19991.
10. Freeman W. J., The physiology of perception, *Scientific American*, 1991, s. 78-85.
11. Górska T., Grabowska A., Zagrodzka J. (red.), *Mózg a zachowanie*, PWN, Warszawa, 1997.
12. Grossberg S., *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control*, Reidell Press, Boston, 1982.
13. Hebb D. O., *The Organization of Behaviour, A Neurobiological Theory*, Wiley, New York, 1949.
14. Hertz J., Krogh A., Palmer R., *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Company, The Advanced Book Program, 350 Bridge Parkway, Redwood City, CA 94065, 1991.
15. Horzyk A., A Hybrid Neural Network Used for the Color Conversion from RGB to CMY and a New Method for Choosing a Learning Set, *Image Processing & Communications*, Vol. 4, Num. 1-2, ATR Bydgoszcz, 1998.
16. Horzyk A., A Neural Network Used for the Color Conversion from RGB to CMY, *Prace informatyczne*, Zeszyty Naukowe Uniwersytetu Jagiellońskiego, Zeszyt 9, Kraków, 1999.
17. Jean J. S. N. and Wang J., Weight smoothing to improve network generalization, *IEEE Trans. Neural Networks 5*, 1994, s. 752-763.
18. Kandel E. R., Schwartz J. H., *Principles of Neural Science*, 4-th edn, Elsevier, New York, 1996.
19. Kohonen T., *Self-Organization and Associative Memory*, Springer-Verlat, Berlin, 1984.
20. Konturek S., *Fizjologia człowieka (Neurofizjologia)*, tom IV, Skrypt Wydziału Lekarskiego i Stomatologii, Akademia Medyczna im. M. Kopernika w Krakowie, Kraków, 1992.
21. Kosko B., *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, NJ Prentice Hall, Englewood Cliffs, 1992.

22. Krogh A. and Hertz J. A., A simple weight decay can improve generalization, *Advances in Neural Information Processing Systems*, vol 4. ed Moody J., Hanson S. J. and Lippman R. P., CA: Morgan Kaufmann, San Mateo, 1992.
23. Michalski R., Tecuci G., *Machine Learning – A Multistrategy Approach*, Volume IV, Morgan Kaufmann Publishers, Inc., San Francisco, 1994.
24. Michalski R. S., Bratko I., Kubat M., *Machine Learning and Data Mining, Methods and applications*, John Wiley & Sons Ltd., West Sussex, 1998.
25. Nilsson N. J., *Principles of Artificial Intelligence*, Tioga Publishing Co., Palo Alto, 1980.
26. Novák M., Faber J., Kufudaki O., *Neuronové sítě a informační systémy živých organismů*, Grada a.s., Praha, 1993.
27. Pöppel E., *Granice świadomości*, PIW, Warszawa, 1989.
28. Rosenblatt F., *Principles of Neurodynamics: Perceptron and the Theory of Brain Mechanisms*, Spartan Books, Washington D.C., 1961.
29. Rumelhart D. E. and McClelland J. L., *Parallel Distributed Processing*, vol. 1, MA: MIT, 1986.
30. Smith M., *Neural Networks for Statistical Modeling*, NK: Van Nostrand Reinhold, New York, 1993.
31. Spitzer M., *The Mind within the Net, Models of Learning, Thinking, and Acting*, A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England,1999.
32. Stevenson M., R. Winter and B. Widrow, Sensitivity of feedforward neural networks to weight errors, *IEEE Trans. Neural Networks 1*, 1990, s. 71-80,.
33. Tadeusiewicz R., *Elementarne wprowadzenie do techniki sieci neuronowych z przykładowymi programami*, Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1998.
34. Tadeusiewicz R., *Sieci neuronowe*, Akademicka Oficyna Wydawnicza RM, Warszawa, 1993.
35. Tadeusiewicz R., Flasiński M., *Rozpoznawanie obrazów*, PWN, 1991, Warszawa, 1991.
36. Wasserman P. D., *Advanced Methods in Neural Computing*, Van Nostrand Reinhold, New York, 1993.
37. Wróbel A., *Jak działa mózg – czyli od receptora do percepcji,* PWN, Warszawa, 1994.
38. Żurada J., Barski M., Jędruch W., *Sztuczne sieci neuronowe*, PWN, Warszawa, 1996.