Algorytmy przeszukiwania w zastosowaniu do perceptrona wielowarstwowego

Mirosław Kordos

Autoreferat rozprawy doktorskiej

promotor: prof. dr hab. Włodzisław Duch

Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

Gliwice 2005

Spis treści

Teza pracy

Algorytmy oparte na systematycznych technikach przeszukiwania mogą być skutecznie zastosowanie do uczenia sieci neuronowej typu perceptrona wielowarstwowego (MLP – multilayer perceptron) przeznaczonej do klasyfikacji danych oraz do ekstrakcji reguł logicznych. Zaproponowane rozwiązania są proste w implementacji i często osiągają lepsze wyniki od tradycyjnych metod opartych na gradiencie analitycznym.

Cel pracy

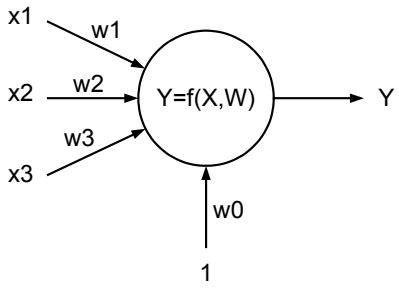
Celem pracy było opracowanie algorytmów opartych na idei lokalnego przeszukiwania przestrzeni rozwiązań do wyznaczania optymalnych wartości wag w procesie uczenia sieci MLP w zastosowaniu zarówno do klasyfikacji danych jak i do ekstrakcji reguł logicznych.

Oryginalny wkład autora

Oryginalny wkład autora obejmuje: szczegółową analizę powierzchni błędu sieci MLP popartą wizualizacją powierzchni błędu w oparciu o redukcję wymiarowości przy pomocy analizy składowych głównych (PCA – principal component analysis), analizę istotnych kierunków w przestrzeni wag sieci MLP, opracowanie dwóch algorytmów uczenia sieci MLP opartych na idei lokalnego przeszukiwania: gradientu numerycznego i algorytmu zmiennego kroku przeszukiwania (VSS - variable step search algorithm) oraz opracowanie dwóch metod uczenia sieci MLP o specjalnej strukturze (SMLP) przeznaczonej do ekstrakcji reguł logicznych z danych: metody bezpośredniego przeszukiwania i metody opartej na zmodyfikowanym algorytmie VSS.

Wprowadzenie

Najczęściej stosowanym typem sieci neuronowych jest perceptron wielowarstwowy (MLP). Podstawowym elementem sieci MLP jest neuron.

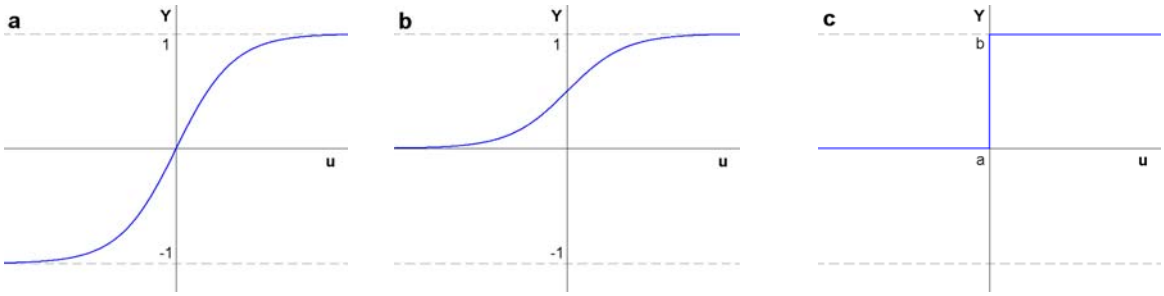


Rys. 1. Model neuronu.

Parametry w_i zwane są wagami neuronu, zaś wielkości x_i są sygnałami wejściowymi. Waga w_0 zwana jest biasem jest do niej podłączona stała wartość sygnału wejściowego równa 1 lub –1. Neuron dokonuje przekształcenia ważonej sumy wejść u na wielkość wyjściową Y według przekształcenia zwanego funkcją transferu. Funkcje transferu neuronów w sieci MLP są funkcjami monotonicznymi, najczęściej używane to tangens hiperboliczny (rys. 2.a), sigmoida logistyczna (rys. 2.b) i funkcja skokowa (rys. 2.c). Większość analizy typowych sieci MLP jest w niniejszej pracy przeprowadzona w oparciu o sigmoidy logistyczne o następującej funkcji transferu:

$$Y = \frac{1}{1+\exp(-\beta \cdot u)} \qquad \text{gdzie} \qquad u = \sum_{i=0}^{N} x_i w_i \tag{1}$$

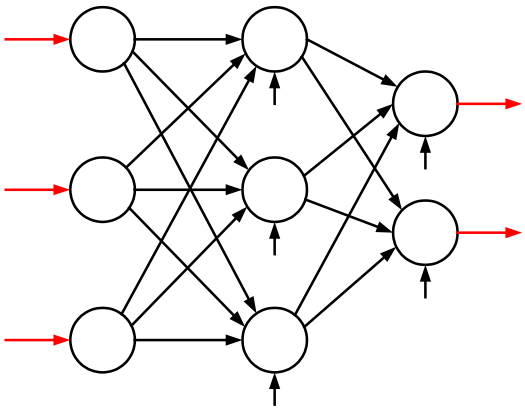
gdzie β jest współczynnikiem stromości sigmoidy. Na ogół przyjmuje się β=1.



Rys. 2. Najczęściej stosowane funkcje transferu neuronów w sieci MLP: a – tangens hiperboliczny, b – sigmoida logistyczna, c – skokowa funkcja transferu.

Sieć MLP składa się z warstw połączonych ze sobą neuronów. Pierwsza warstwa to neurony wejściowe, które posiadają liniowe funkcje transferu i ich zadaniem jest jedynie dystrybucja sygnałów wejściowych do kolejnych warstw sieci. W kolejnych warstwach

stosowane są neurony o sigmoidalnych (lub innych nieliniowych) funkcjach transferu. Wagi tych neuronów podlegają modyfikacji w procesie uczenia sieci, którego celem jest doprowadzenie sieci do stanu, w którym będzie ona wykonywać żądane odwzorowanie przestrzeni wejściowej w przestrzeń wyjściową. W praktyce stosuje się od dwóch do czterech warstw neuronów (wliczając w to warstwę wejściową). Sygnały są propagowane przez sieć tylko w jednym kierunku od wejścia do wyjścia, zgodnie ze zwrotem strzałek na rys. 3.



Rys. 3. Model trójwarstwowej sieci MLP.

Sieci MLP mogą być zastosowane do klasyfikacji danych oraz do aproksymacji wartości funkcji. Praca ta koncentruje się jedynie na zagadnieniu klasyfikacji. Przez pojęcie klasyfikacji rozumie się dzielenie dowolnego zbioru elementów na grupy, do których zalicza się elementy różniące się, ale podobne, tj. mające właściwości wyróżniające daną grupę. Zbiór elementów należących do jednej grupy nazywa się klasą, a każdy element klasy obiektem lub wektorem. Każdy wektor opisany jest zbiorem cech. Cechy mogą mieć wartości numeryczne, lub symboliczne. Sieć MLP wymaga jednak by dane, które będą jej sygnałami wejściowymi były numeryczne, dlatego cechom symbolicznym należy przypisać wartości numeryczne, które będą podane na wejścia sieci.

Typowo przed rozpoczęciem uczenia sieci, ustala się wartości wszystkich wag losowo, np. z przedziału (-1;1). Podczas uczenia sieci wektory zbioru treningowego są kolejno podawane na wejścia sieci i dla każdego wektora sieć generuje pewne sygnały wyjściowe. Każdy neuron wejściowy przypisany jest do jednej cechy, zaś każdy neuron wyjściowy przypisany jest do jednej klasy. Celem procesu uczenia sieci jest doprowadzenie do takiego stanu, że w odpowiedzi na każdy wektor wejściowy tylko neuron wyjściowy przypisany do tej samej klasy, co aktualny wektor generuje sygnał bliski jedności, zaś pozostałe neurony wyjściowe – sygnały bliskie zeru. Przyjmuje się, że sieć sklasyfikowała wektor do klasy odpowiadającej neuronowi wyjściowemu, który wygenerował największy sygnał.

Wprowadza się dwa rodzaje błędów: błąd klasyfikacji i błąd odległości. Błąd klasyfikacji podaje jaka ilość wektorów została nieprawidłowo sklasyfikowana. Zazwyczaj używa się jego dopełnienia do jedności zwanego poprawnością klasyfikacji i wyrażanego w procentach. Błąd odległości E podaje o ile suma wartości wszystkich sygnałów wyjściowych wygenerowanych dla wszystkich wektorów różni się od wartości zadanej:

$$E = \sum_v \sum_c f(d_{v,c} - s_{v,c}) \tag{2}$$

gdzie d jest sygnałem oczekiwanym (przeważnie 0 lub 1), natomiast s jest sygnałem wygenerowanym przez neuron c w odpowiedzi na wektor v. Funkcję f nazywa się funkcją błędu. Najczęściej używaną funkcją błędu jest funkcja kwadratowa:

$$E = \sum_v \sum_c (d_{v,c} - s_{v,c})^2 \tag{3}$$

W procesie uczenia sieci na ogół nie minimalizuje się bezpośrednio błędu klasyfikacji, lecz błąd odległości. Błąd odległości może przyjmować ciągłe wartości i dlatego jest łatwiejszy w minimalizacji od błędu klasyfikacji, który przyjmuje tylko skokowe wartości, z dokładnością do jednego wektora. Metody uczenia sieci MLP oparte na gradiencie analitycznym, jak np. algorytm propagacji wstecznej błędu, nie są nawet w stanie bezpośrednio minimalizować błędu klasyfikacji.

Ostatecznym celem uczenia sieci jest jednak zapewnienie poprawnej klasyfikacji wektorów z poza zbioru treningowego. Sieć powinna zatem nauczyć się rozpoznawać właściwości charakterystyczne dla wektorów danej klasy (tzw. generalizacja), a nie nauczyć się tych wektorów na pamięć.

Standardowa sieć MLP klasyfikuje jedynie wektory, nie dostarcza natomiast żadnej informacji, czym się kierowała przy podjęciu określonej decyzji. Użytkownicy systemów klasyfikacji mogą jednak wymagać uzasadnienia decyzji klasyfikatora. Brak uzasadnienia decyzji stanowi często podstawę braku zaufania do działania klasyfikatora, szczególnie w takich dziedzinach, jak. np. medycyna. Dlatego w pracy zaproponowano również metodykę pozyskiwania reguł logicznych objaśniających decyzję podjęte przez sieć. Wykorzystano specjalnie do tego celu zmodyfikowaną strukturę sieci MLP. Reguły logiczne otrzymuje się w postaci:

jeżeli *określona kombinacja cech wektora przyjmuje daną wartość*
to w*ektor należy do określonej klasy*

Do uczenia sieci MLP powszechnie stosowane są metody oparte na gradiencie analitycznym, czasem używa się również metod globalnej minimalizacji, jak np. algorytmy genetyczne, mają one jednak bardzo duży nakład obliczeniowy i dlatego stosowanie ich może być uzasadnione tylko w nielicznych przypadkach, gdy tradycyjne metody gradientowe nie są w stanie znaleźć zadawalającego rozwiązania. Metody oparte na przeszukiwaniu, powszechnie stosowane w innych dziedzinach inteligencji obliczeniowej, w odniesieniu do sieci neuronowych są zupełnie zaniedbane.

Metodyka Badań

Badania miały w większości charakter eksperymentów numerycznych, połączonych z analizą teoretyczną. Większość eksperymentów została przeprowadzona w programie napisanym przez autora w środowisku Delphi. Niektóre badania były przeprowadzone w Matlabie, zwłaszcza w Neural Network Toolbox. Do badań wykorzystywano wiele różnych zbiorów danych, łącznie z danymi z UCI Machine Learning Repository, powszechnie stosowanymi do testowania algorytmów klasyfikacji, których wykorzystanie umożliwiło porównanie wyników z wynikami uzyskanymi przez innych autorów.

Zakres pracy

Praca składa się z trzech zasadniczych części.

Część pierwsza poświęcona jest analizie właściwości sieci MLP, ze szczególnym uwzględnieniem powierzchni błędu sieci MLP, m. in. z wykorzystaniem wizualizacji powierzchni błędu w oparciu o redukcję wymiarowości przy pomocy analizy składowych głównych (PCA). Część ta obejmuje także analizę trajektorii algorytmów uczenia, istotnych kierunków w przestrzeni wag sieci MLP oraz analizę możliwości uczenia sieci w zredukowanej przestrzeni parametrów.

W części drugiej wykorzystano badania przeprowadzone w części pierwszej do opracowania dwóch algorytmów uczenia sieci MLP opartych na idei lokalnego przeszukiwania: gradientu numerycznego i algorytmu zmiennego kroku przeszukiwania (VSS - variable step search algorithm). Algorytmy te porównano z innymi, powszechnie uznanymi na najskuteczniejsze, algorytmami uczenia sieci MLP.

W trzeciej części zaproponowano dwie metody uczenia sieci MLP o specjalnej strukturze (SMLP) przeznaczonej do ekstrakcji reguł logicznych z danych: metodę bezpośredniego przeszukiwania i metodę opartą na zmodyfikowanym algorytmie VSS oraz omówiono metodę pozyskiwania reguł na podstawie analizy wag sieci. Przeprowadzono porównania z dostępnymi w literaturze wynikami uzyskanymi przy pomocy innych klasyfikatorów.

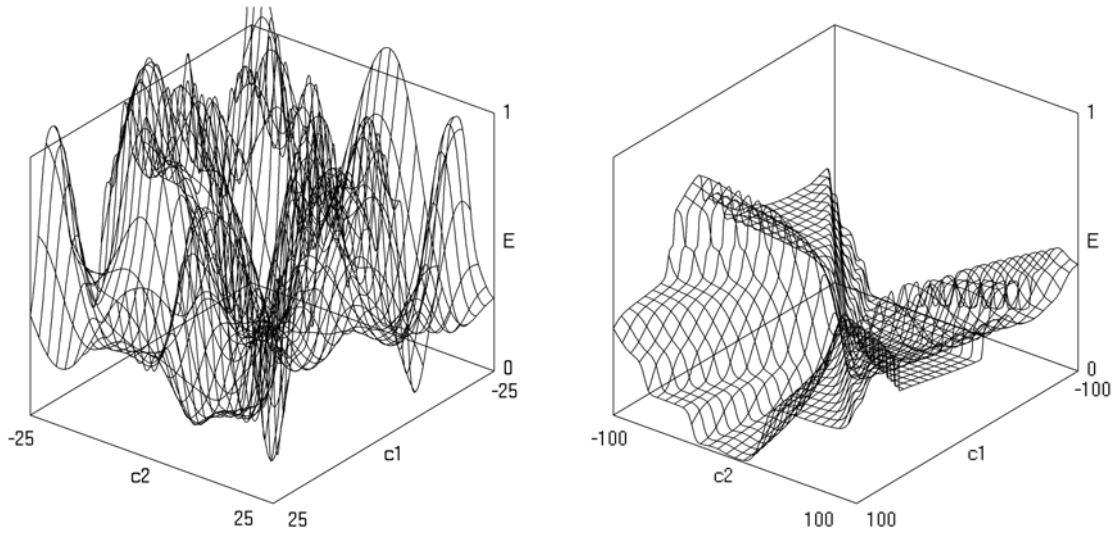
Analiza właściwości sieci MLP

Błąd odległości sieci MLP określony równaniem (2) zależy od wartości wag w sieci oraz od parametrów takich jak: postać funkcji błędu, struktura sieci, funkcja transferu neuronów, zbiór treningowy. Przy określonych pozostałych parametrach, a zmieniających się jedynie wagach, jak to ma na ogół miejsce podczas uczenia sieci funkcję błędu E można traktować jako powierzchnię, zwaną powierzchnią błędu w wielowymiarowej przestrzeni wag.

W literaturze można spotkać często stwierdzenia, że powierzchnia błędu sieci MLP jest trudna do wyobrażenia sobie i posiada wiele minimów lokalnych, jak rys. 4.a. Jednak takie powierzchnie uzyskuje się tylko przy niemonotonicznych funkcjach transferu neuronów, których w sieciach MLP się nie stosuje. Przy zastosowaniu funkcji

monotonicznych, np. sigmoidalnych, powierzchnia błędu ma strukturę wielowymiarowej rozgwiazdy. W jednym z jej wąwozów położona jest trajektoria procesu uczenia sieci.

W eksperymentach mających na celu uzyskanie 3-wymiarowych projekcji powierzchni błędu, przeprowadzono uczenie sieci następującymi metodami: standardowa propagacja wsteczna, algorytm Levenberga-Marquardta (LM), metoda skalowanych gradientów sprzężonych (SCG), gradient numeryczny (NG) i metoda zmiennego kroku przeszukiwania (VSS). Wektor wag po każdym cyklu treningowym wypełniał jeden rząd macierzy wag. Następnie obliczono macierz kowariancji wag i korzystając z metody SVD (singular value decomposition) obliczono jej wartości własne i wektory własne. Procedura ta, będąca jedną z odmian PCA (analizy składowych głównych) stosunkowo dobrze nadaje się do uzyskania takich projekcji, gdyż w większości przypadków w dwóch pierwszych kierunkach PCA zawiera się ok. 95% wariancji wag podczas uczenia. Tabela 1 zawiera typowe wartości własne macierzy kowariancji wag oraz odpowiadające im wariancje zawarte w aktualnej składowej PCA i całkowite wariancje zawarte łącznie w składowych PCA od pierwszej do aktualnej.



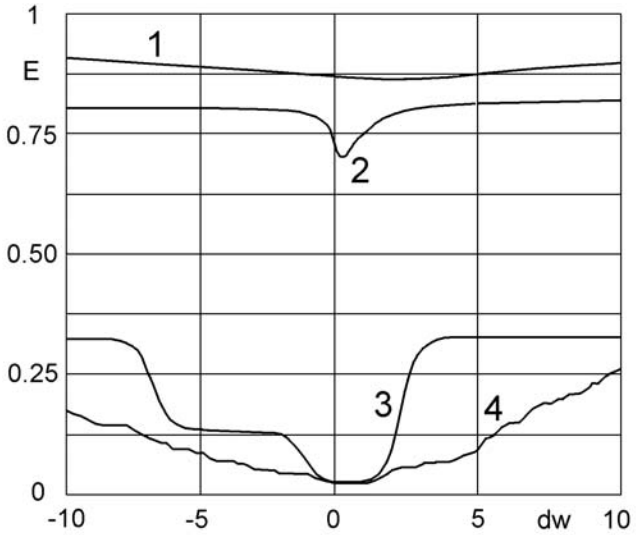
Rys. 4. Po lewej: przykładowa projekcja powierzchni błędu sieci MLP z sinusoidalnymi funkcjami transferu w dwa pierwsze kierunki PCA. Po prawej: przykładowa projekcja powierzchni błędu sieci MLP z sigmoidalnymi funkcjami transferu w dwa pierwsze kierunki PCA. Na osiach poziomych dwa pierwsze kierunki PCA na osi pionowej błąd względny sieci $E=E(\boldsymbol{W})/N_V N_C$, gdzie N_V –liczba wektorów, N_C –liczba klas.

Tabela. 1. Typowy rozkład wartości własnych macierzy kowariancji wag.

składowa PCA	1	2	3	4	5	6	7	8	9	10
wartość własna	33.204	1.4550	0.5969	0.2554	0.1578	0.0679	0.0547	0.0324	0.0265	0.0191
% akt. wariancji	0.9245	0.0405	0.0166	0.0071	0.0044	0.0019	0.0015	0.0009	0.0007	0.0005
% całk. wariancji	0.9245	0.9651	0.9817	0.9888	0.9932	0.9951	0.9966	0.9975	0.9982	0.9988

Zaobserwowano sposób, w jaki powierzchnia błędu zależy od poszczególnych parametrów, takich jak struktura sieci, charakter zbioru uczącego, rodzaj funkcji błędu i funkcji transferu neuronów. Jednak najważniejsze wnioski, które stosują się do każdego przypadku są następujące:

1. Powierzchnia błędu ma strukturę rozgwiazdy.
2. Minima lokalne położone w kraterach (jak na rys. 4a), choć możliwe, w sieciach uczonych na rzeczywistych danych występują niezmiernie rzadko.
3. Istnieje wiele równoważnych minimów globalnych, które bez użycia członu regularyzacyjnego w funkcji błędu znajdują się w nieskończoności, na dnie najniżej położonych wąwozów.
4. Minima lokalne również znajdują się w nieskończoności, lecz w wyżej położonych wąwozach.
5. Przyczyną problemów ze zbieżnością niektórych algorytmów uczących jest raczej to, że pochodne w poszczególnych kierunkach różnią się nawet o kilka rzędów wielkości (ill-conditioning), utykanie trajektorii na dużych płaskich obszarach, które podobnie zerują gradient, jak utknięcie w minimum lokalnym lub taka inicjalizacja wag, z której dany algorytm nie jest w stanie dojść do jednego z minimów globalnych, a tylko do wyżej położonego wąwozu.
6. Wąwozy te nie są proste, jak na rys. 4.b, lecz tworzą łuki. Łuków tych nie widać na projekcjach PCA, gdyż PCA jako metoda liniowa daje w wyniku jedynie średnie kierunki wąwozów, natomiast widoczne są one przy projekcjach trajektorii uczenia na pierwszy i drugi kierunek PCA.
7. Trajektorie na ogół jedynie nieznacznie modyfikują swój kierunek między dwoma sąsiednimi epokami uczenia.
8. Istnieją istotne różnice między przecięciami powierzchni błędu w kierunkach wag warstwy ukrytej i warstwy wyjściowej w 3-warstwowej sieci MLP. W przypadku sieci 4-warstwowych wyższa warstwa ukryta (leżąca bliżej warstwy wyjściowej) ma właściwości pomiędzy właściwościami warstwy niższej ukrytej (a ta ma takie same, jak jedyna warstwa ukryta w sieciach 3-warstwowych) a warstwy wyjściowej.



Rys. 5. Typowe przekroje powierzchni błędu 3-warstwowej sieci MLP: 1) w kierunku wagi warstwy ukrytej na początku uczenia, 2) w kierunku wagi warstwy wyjściowej na początku uczenia, 3) w kierunku wagi warstwy wyjściowej na końcu uczenia, 4) w kierunku wagi warstwy ukrytej na końcu uczenia.

Algorytmy uczenia sieci MLP oparte na przeszukiwaniu

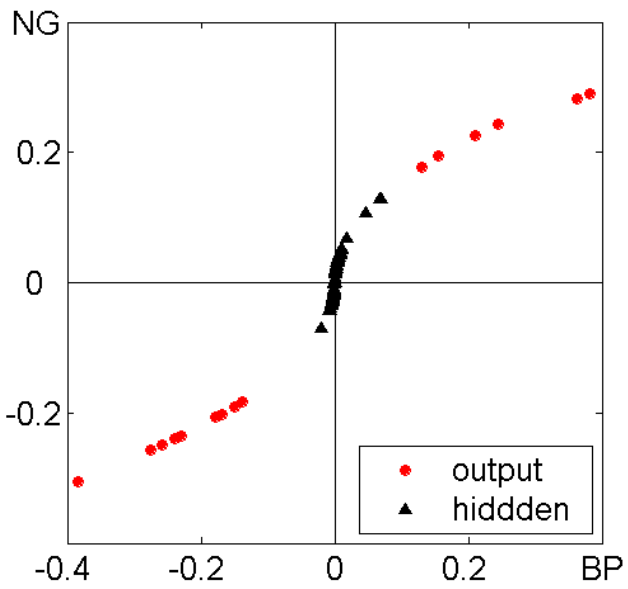
Zaproponowano dwa nowe algorytmy uczenia sieci MLP: gradient numeryczny (NG) i algorytm zmiennego kroku przeszukiwania (VSS - variable step search algorithm).

Gradient Numeryczny

Aby wyznaczyć numerycznie kierunek gradientu należy zmodyfikować daną wagę o niewielką wartość *dw*, zmierzyć jak to wpływa na błąd sieci i powtórnie przywrócić oryginalną wartość wagi. Operację należy powtórzyć z każdą wagą w sieci. W ten sposób otrzymamy składowe gradientu *dE(w)* w kierunkach wszystkich wag:

$$dE(w) = [E(w) - E(w + dw)] \tag{4}$$

Można zaobserwować, że zmniejszanie *dw* poniżej pewnej granicy (ok. 0.02) nie wpływa już na dalsze zmiany składowych gradientu i niezależnie od wielkości *dw* otrzymujemy nieco inny kierunek, niż ten wyznaczony przy pomocy metody wstecznej propagacji błędu (BP), co pokazano na rys. 6.



Rys. 6. Typowa zależność między składowymi gradientu wyznaczonymi przez propagację wsteczną (BP) i przez gradient numeryczny (NG).

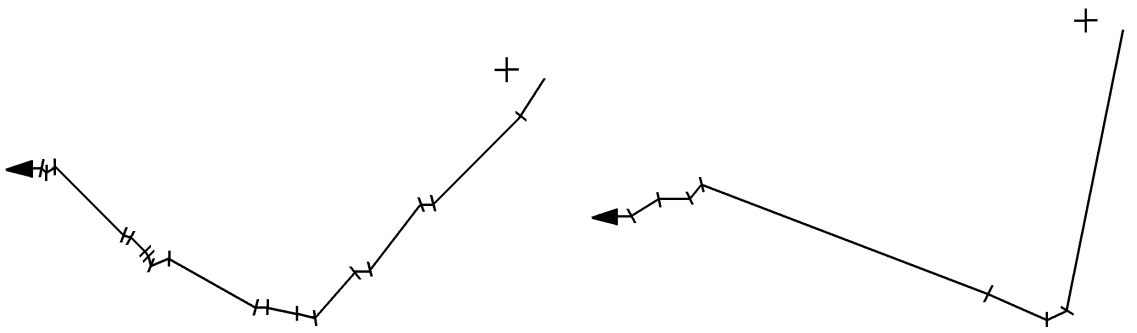
Należy zauważyć, że przy identycznej metodzie określania długości kroku w kierunku gradientu, a zwłaszcza przy zastosowaniu minimalizacji kierunkowej używając gradientu numerycznego uzyskuje się zbieżność procesu uczenia w mniejszej ilości kroków, niż przy gradiencie analitycznym (BP). Ponadto zaobserwowano czasami przypadki utknięcia procesu uczenia sieci metodą BP w danym punkcie i możliwość wyjścia z tego punktu, kontynuując uczenie gradientem numerycznym. Wynika to stąd, że gradient numeryczny, jako wielkość mierzona bezpośrednio dla każdej wagi jest bliższy

kierunkowi rzeczywistego minimum, niż gradient analityczny, który dla warstw ukrytych jest jedynie szacowany.

Jednak, jak można zaobserwować z rysunku 5, poruszanie się w kierunku gradientu (nawet numerycznie określonego) nie jest wcale poruszaniem się w kierunku optymalnym. Na początku uczenia na ogół pochodne w kierunku wag wyjściowych są większe, niż w kierunku wag ukrytych, pomimo, że odległości do minimum na powierzchni błędu są większe w kierunku wag ukrytych niż wyjściowych. Zaproponowano więc następującą modyfikację kierunku poruszania się w oparciu o statystyczne zależności między wartością pochodnej w kierunku danej wagi, fazą procesu uczenia, warstwą sieci i odległością do minimum na powierzchni błędu:

$$dS(w) = (1 + a \cdot \exp(-b \cdot epoch)) \cdot sign(dE(w)) \cdot \sqrt{|dE(w)|} \quad \text{dla for } -dE_1 \le dE(w) \le dE_1 \qquad (5)$$
$$dS(w) = (1 + a \cdot \exp(-b \cdot epoch)) \cdot sign(dE(w)) \cdot \sqrt{dE_1} \quad \text{w przeciwnym razie}$$

gdzie *dS(w)* jest składową wagi *w* w kierunku bieżącego kroku. Typowe wartości współczynników to a=10÷20 dla warstwy ukrytej, a=0 dla warstwy wyjściowej, b=0.1÷0.2, dE_1 jest równe 4÷8-krotnej wartości odchylenia standardowego pochodnych funkcji błędu względem poszczególnych wag. Użycie tej modyfikacji pozwala znacząco skrócić trening, nie powodując jednocześnie niestabilności oraz dużego wzrostu nakładu obliczeniowego przy wzroście wielkości sieci, typowego dla algorytmów wykorzystujących drugie pochodne.



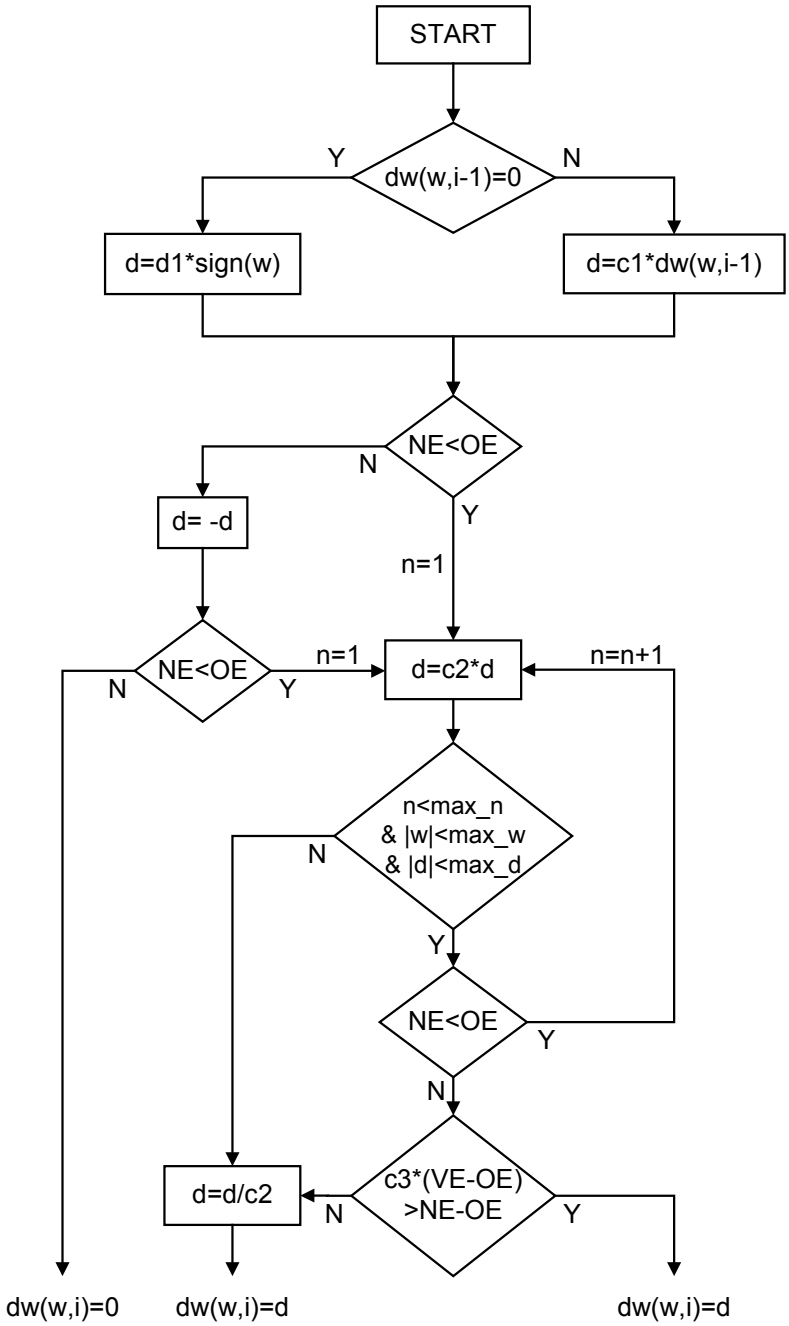
Rys. 7. Projekcja w dwóch kierunkach PCA trajektorii uczenia sieci (zbiór Iris, struktura sieci 4-4-3). Po lewej: poruszając się w kierunku gradientu numerycznego określonego równaniem (4). Po prawej: poruszając się w kierunku określonym równaniem (5).

Do gradientu numerycznego można dodać czynnik momentum, podobnie, jak do gradientu analitycznego, lecz nie na wszystkich zbiorach danych momentum dobrze działa.

Ponieważ przy określaniu kierunku w gradiencie numerycznym, jak i w pozostałych zaproponowanych tu metodach (VSS, oraz metody uczenia sieci SMLP) podczas całego uczenia zmieniana jest tylko jedna (na ogół) waga na raz, więc nie ma potrzeby, by po każdej takiej zmianie sygnały były propagowane przez całą sieć. Dlatego wszystkie iloczyny sygnałów wchodzących do neuronów i odpowiadających im wag są pamiętane dla każdego wektora w specjalnej tablicy nazwanej tabelą sygnałów. Przy modyfikacji pojedynczej wagi modyfikuje się tylko odpowiadające jej wpisy w tabeli

sygnałów, co od kilku (przy bardzo małych sieciach) do kilkuset razy (przy dużych sieciach) skraca czas operacji w porównaniu z propagacją sygnałów przez całą sieć przy każdej zmianie wagi. Skuteczność tabeli sygnałów oprócz wielkości sieci zależy też od jej struktury i typu funkcji transferu. Przykładowo dla sieci MLP o strukturze: 125 wejść, 8 neuronów ukrytych i 2 wyjścia uzyskuje się około 35-krotne przyspieszenie, zaś dla sieci SMLP o podobnej strukturze około 60-krotne.

Algorytm zmiennego kroku przeszukiwania (VSS)



Rys. 8. Określanie wartości pojedynczej wagi w jednym cyklu uczenia metodą zmiennego kroku przeszukiwania (VSS).

VSS nie korzysta z informacji o gradiencie, a raczej korzysta z własności powierzchni błędu przedstawionych w punktach 5÷7. Algorytm ten podobnie, jak gradient numeryczny zmienia jedną wagę na raz, lecz od razu szuka w bardzo grubym przybliżeniu minimum w kierunku tej wagi i od razu posuwa się do tego minimum. Już z tego nowego punktu szuka minimum w kierunku kolejnej wagi i tak dalej, aż wszystkie wagi zostaną sprawdzone. Podobnie, jak gradient numeryczny, VSS korzysta z tabeli sygnałów. VSS wykorzystuje tą własność, że dna wąwozów powierzchni błędu są wolno zmieniającymi kierunek wąwozami. A zatem optymalna zmiana danej wagi w następnym cyklu treningowym jest podobna do jej zmiany w poprzednim cyklu.

Objaśnienie symboli występujących na schemacie (rys. 8):
dw(w,i) – zmiana wagi w w cyklu i,
c1, c2, c3, d1 – współczynniki, typowe wartości: c1=0.35 , c2=2 , c3=0.3, d1=0.02
Jeśli algorytm przejdzie przez d=d1*sign(w) to opcjonalnie daną wagę można zamrozić
dw(w,0)=1 dla każdego w (w pierwszym cyklu treningowym każda waga jest na początku zmieniana o d=c1*dw(w,0)=0.35.
NE – aktualny błąd sieci
OE – poprzedni błąd sieci
VE – błąd sieci dla tego samego i oraz n mniejszego o 2 od aktualnego n
max_n, max_d, max_w – opcjonalne parametry ograniczające wzrost wag

Tabela 2. Porównanie algorytmów VSS, NG (gradient numeryczny) , LM (algorytm Levenberga-Marquardta) i SCG (skalowany gradient sprzężony)

zbiór	%	VSS				NG				LM				SCG			
strukt. sieci	test	N_t	MB	CR	C_t	N_t	MB	CR	C_t	N_t	MB	CR	C_t	N_t	MB	CR	C_t
Iris 4-4-3	96.0	3.5	-	100	112 84	11	-	100	175 143	20	-	80	-	54	-	90	245
Breast 10-4-2	96.0	1.5	-	100	64 38	4	-	100	112 76	15	1.5	85	-	38	0.4	60	165
Mushrooms 125-4-2	99.7	2.0	0.4	100	160 64	21	0.4	100	1070 464	6	240	90	566	45	40	100	90
Thyroid 21-4-3	98.0	10	0.2	95	697 366	80 84	0.2	0	1724 998	43	30	60	1333	186	1.0	75	619
Shuttle 9-6-7	99.0	6.0	1.6	100	457 287	40 42	1.6	90	1200 789	15	1400	60	1280	46	20	60	238

N_t – liczba cykli treningowych przez jaką należy sieć uczyć, by osiągnęła podaną poprawność na zbiorze testowym lub w kroswalidacji równą %test
MB – zajętość pamięci przez macierze na których operuje algorytm, nie licząc pamięci wykorzystywanej przez sam program i przez zbiór treningowy
CR – procent treningów, które zakończyły się sukcesem (uzyskaniem zbieżności)
C_t – łączny nakład obliczeniowy, wyrażony stosunkiem całkowitego czasu treningu do czasu działania sieci w trybie testu na zbiorze treningowym (górna pozycja: dla sigmoidalnych, dolna: dla (wielo)schodkowych funkcji transferu)

VSS stosuje schemat określania wartości zmian wag przedstawiony na rys 8. Nakład obliczeniowy na jeden cykl treningowy dla algorytmu VSS jest większy niż dla

SCG, lecz nie rośnie ze wzrostem sieci tak, jak rośnie nakład obliczeniowy algorytmu Levenberga-Marquardta, lecz w przybliżeniu jak log(N_w), w stosunku do czasu samej propagacji sygnałów przez sieć podczas pracy w trybie testu, gdzie N_w - liczba wag. Natomiast VSS wymaga minimalnej liczby cykli treningowych, co powoduje, że na ogół ten nakład jest i tak znacznie mniejszy, niż dla innych algorytmów. VSS jest w stanie znaleźć obszary o bardzo niskich wartościach błędu oraz cechuje się dużą powtarzalnością wyników. To, że VSS nie korzysta z informacji o gradiencie powoduje, że nie zmniejsza on kroku, jeżeli gradient maleje, tak jak robią to metody oparte na gradiencie (co powoduje długie „ogony" uczenia). Krok algorytmu VSS zależy od krzywizny powierzchni błędu, a nie od jej stromości, dopóki stromość jest większa od zera. Przy zerowej stromości algorytm kończy działanie.

Ekstrakcja reguł logicznych z sieci MLP

Struktura sieci SMLP i ekstrakcja reguł logicznych

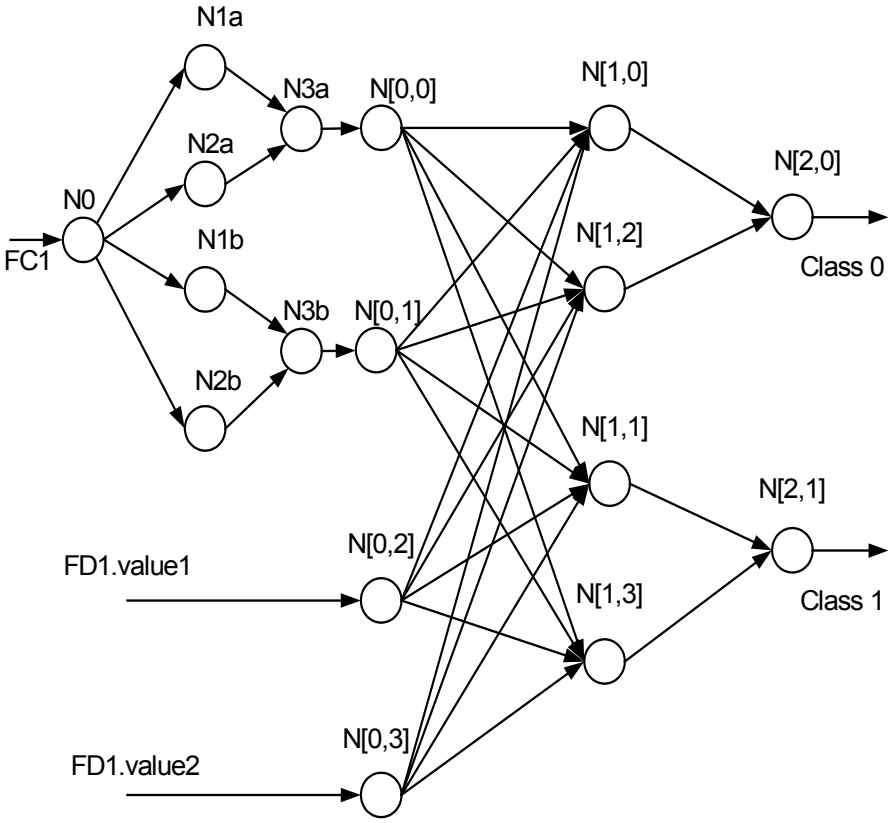
Przedstawiono sieć SMLP służącą do klasyfikacji danych i ekstrakcji reguł logicznych o hiperprostokątnych granicach decyzyjnych, prostopadłych do osi w przestrzeni cech. Rozwiązanie takie może czasami nie być wystarczające dla bardzo skomplikowanych danych. Istnieje wówczas możliwość uzyskania innych granic decyzyjnych poprzez utworzenie dodatkowych cech będących dowolną kombinacją cech oryginalnych. Jednak w większości przypadków rozwiązanie to zapewnia najbardziej zrozumiałą postać reguł i pozwala na osiągnięcie wysokich wyników klasyfikacji w porównaniu z innymi metodami. Dodatkowo, jeżeli nie można znaleźć reguł o prostokątnych granicach decyzyjnych opisujących dane zagadnienie – może świadczyć to często o niespójności lub złej jakości danych dostarczonych do analizy.

Sieć SMLP (Search-based MLP) w podstawowej wersji ma 3 warstwy i skokowe funkcje transferu neuronów. Taka podstawowa wersja jest rozwiązaniem wystarczającym dla większości praktycznych danych.

Sieć SMLP wymaga, aby dane wejściowe były dyskretne lub symboliczne. Jeżeli dane są ciągłe muszą zostać zdyskretyzowane przed podaniem na wejście sieci. Wykorzystuje się do tego specjalny moduł dyskretyzacji danych, tzw. L-unit (logical unit) złożony z trzech neuronów, które realizują mechanizm okienkowy, generując sygnał wejściowy do odpowiedniego neuronu właściwej sieci, tylko jeśli dana ciągła jest z zadanego przedziału. Przed rozpoczęciem treningu dane ciągłe zostają wstępnie podzielone na kilka równych lub równolicznych części, a jeżeli podczas uczenia sieci okazuje się, że dane z dwóch sąsiednich przedziałów dają taki sam wynik, to przedziały te zostają automatycznie scalone w jeden. Podczas treningu zostają także dopasowywane optymalnie punkty podziału realizowane przez L-unity.

Ekstrakcja reguł logicznych dokonywana jest na podstawie analizy wartości wag po zakończonym treningu sieci. Sieć ma skokowe funkcje transferu neuronów i wagi będące liczbami –1, 0 lub +1, dzięki czemu każdy jej neuron realizuje operacje logiczną typu M-of-N. Zawsze jest możliwa redukcja tych reguł do prostych reguł typu OR/AND,

najczęściej bez dodawania dodatkowych neuronów. Biasy neuronów warstwy ukrytej mają wartości będące liczbą całkowitą ±0.5, zaś biasy warstwy wyjściowej – zawsze +0.5. Każdy neuron warstwy ukrytej dedykowany jest do określonej klasy (nie ma pełnych połączeń między warstwą ukrytą a wyjściową). Dlatego sieć może być uczona dla każdej klasy oddzielnie.

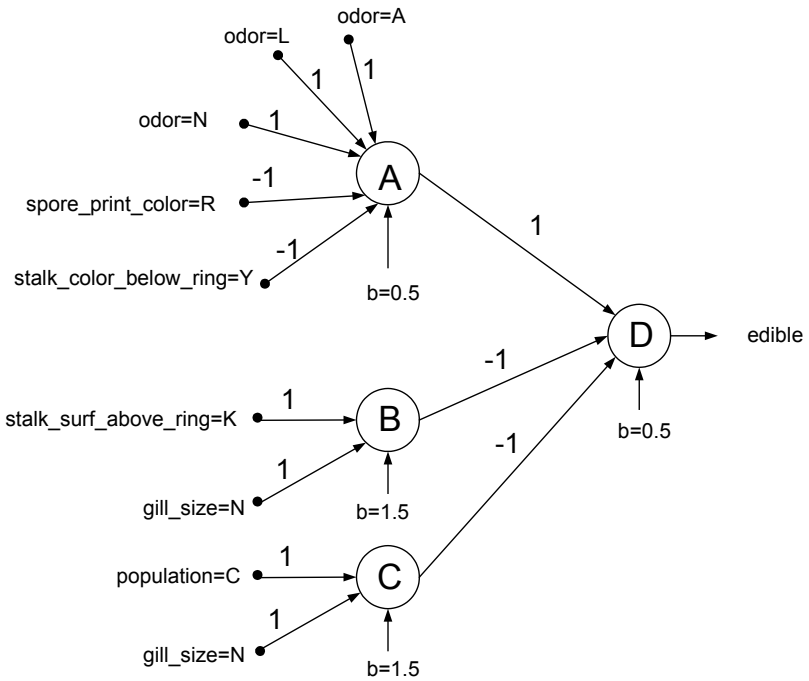


Rys. 9. Schemat sieci SMLP z widocznymi dwoma jednostkami typu L służącymi do dyskretyzacji ciągłej cechy FC1.

Jeżeli wagi wchodzące do neuronu ukrytego należą do różnych wartości tej samej cechy, to przy tworzeniu reguły łączymy je operatorem OR, ponieważ dla danego wektora dana cecha może przyjmować tylko jedną wartość. Każda wartość cechy, której odpowiada waga ujemna interpretowana jest jako AND NOT. Wystarczy, aby dany wektor miał jedną wartość jednej cechy, która ma wagę ujemną, aby spowodować niespełnienie całej reguły. Każda grupa wag dodatnich należących do tej samej cechy generuje regułę typu M-of-N, gdzie N jest ilością grup wag dodatnich, zaś bias neuronu ma wartość M-0.5. (neuron generuje sygnał, gdy suma sygnałów wejściowych pomnożonych przez odpowiadające im wagi jest większa, niż bias neuronu). Jeżeli dany neuron ukryty jest połączony z neuronem wyjściowym dodatnią wagą (tak jest zawsze w przypadku pierwszego neuronu ukrytego), to wartości cech sprzyjające przynależności obiektu do danej klasy tworzą wagi dodatnie (+1), niesprzyjające wagi ujemne (-1), zaś niewpływające na tą przynależność – wagi zerowe. Przy kolejnych neuronach ukrytych jest to uzależnione od wagi z jaką są one połączone z neuronem wyjściowym.

Pierwszy neuron ukryty klasyfikuje zawsze najwięcej obiektów, jeżeli nie sklasyfikował wszystkich, to dodawany jest kolejny neuron ukryty połączony dodatnią wagą z neuronem wyjściowym, by sklasyfikował pozostałe, Jeśli zaś pierwszy neuron sklasyfikował (przepuścił) także obiekty nie należące do jego klasy, to dodawany jest kolejny neuron połączony ujemną wagą z neuronem wyjściowym, którego zadaniem jest klasyfikacja wyjątków, czyli wektorów błędnie sklasyfikowanych przez pierwszy neuron. Pod wpływem tych wektorów pierwszy neuron wygeneruje wprawdzie sygnał, ale drugi neuron też wygeneruje sygnał i sygnały te się nawzajem zniosą w neuronie wyjściowym. Jeżeli by wystąpiła struktura danych nie dająca się opisać przy pomocy takiej architektury sieci (np. konieczność połączenia reguł częściowych zarówno operatorem OR jak i AND) to można lokalnie dodać neurony warstwy pośredniej, jednak w praktycznych testach taka sytuacja nie wystąpiła).

Istnieje zawsze pewna optymalna dokładność klasyfikacji i związana z nią złożoność reguł. Nie należy ogólnie dążyć do poprawnego sklasyfikowania 100% danych ze zbioru treningowego, bo prowadzi to (tak samo, jak w przypadku większości innych klasyfikatorów) do przeuczenia się sieci, co skutkuje pogorszeniem się wyników na zbiorze testowym. Na rys. 10 przedstawiono schemat sieci nauczonej do rozpoznawania obiektów klasy “edible” znanego zbioru testowego o nazwie “Mushrooms” wraz z regułami wygenerowanymi na podstawie wartości wag.



Rys. 10. Sieć SMLP dla jednej klasy danych “Mushrooms”. Pokazano tylko niezerowe wagi.

Neuron A generuje następującą regułę częściową:
if odor=(A or L or N) and (not s_p_color=R) and (not stalk_color_below_ring=Y) then reguła A

Neuron B generuje następującą regułę częściową:
if (gill_size=N and stalk_surface_above_ring=K) then reguła B

Neuron B generuje następującą regułę częściową:
if (gill_size=N and population=C) then reguła C

Neuron D generuje następującą regułę końcową:
if reguła A and (not reguła B) and (not reguła C) then edible

Metody uczenia sieci SMLP

Zostały opracowane dwie metody uczenia sieci SMLP: bezpośrednie przeszukiwanie (SMLP-DS) polegające na zmianie najpierw jednej wagi na raz, a gdy to nie przynosi skutku wówczas dwóch wag na raz, lub stosowaniu przeszukiwania wiązką oraz uczenie oparte na zmodyfikowanym algorytmie VSS (SMLP-VSS). W obu algorytmach uczeniu podlegają tylko wagi neuronów ukrytych, na początku sieć ma po jednym neuronie ukrytym na klasę i kolejne neurony dostawiamy tylko, gdy pierwszy neuron nie zapewnia żądanej jakości klasyfikacji. Po dostawieniu kolejnego neuronu, wagi pierwszego neuronu zostają zamrożone i uczy się tylko nowo dodany neuron. Uczenie można przeprowadzać dla podsieci należącej do każdej klasy oddzielnie. Na początku uczenia wszystkie wagi neuronu ukrytego mają zerowe wartości, zaś bias wynosi na ogół +0.5, jest więc zerowy sygnał na wyjściu neuronu.

SMLP-DS

Zmieniamy wartość każdej wagi po kolei, najpierw na +1, potem na –1. Jeżeli po zmianie błąd zmaleje, to zmianę tą zostawiamy i przechodzimy do następnej wagi. W obu algorytmach kolejność w jakiej przeszukiwane są wagi odgrywa pewną rolę, jednak w SMLP-DS rola ta jest znacznie większa. Ma to pewne zalety, ale i wady. Zaletą jest to, że w zależności od tego, w jakiej kolejności przeszukujemy wagi, możemy otrzymać różne zestawy reguł logicznych. Zestawy te są często prawie równoważne, co do ilości poprawnie sklasyfikowanych wektorów, jednak mogą pozwolić na odkrycie dodatkowych informacji zawartych w strukturze danych. Wadą tego rozwiązania jest natomiast to, że niewłaściwie dobierając kolejność przeszukiwania można otrzymać zbyt skomplikowane reguły, jednocześnie odznaczające się słabą generalizacją. Dlatego wskazanym jest stosowanie jednego z dwóch rozwiązań: albo uszeregowanie kolejności przeszukiwania cech według ich wzajemnej informacji (np. najpierw uczymy sieć z każdą cechą pojedynczo i przeprowadzamy ranking cech) albo wprowadzenie minimalnego progu zmiany wagi. Jeżeli po zmianie wagi błąd sieci nie zmaleje przynajmniej o wartość określoną przez ten próg – wówczas pozostawia się zerową wartość tej wagi. Czasami dobre wyniki uzyskuje się stosując przeszukiwanie wiązką.

Czasami może się okazać nie wystarczająca zmiana jednej wagi na raz – można wówczas zmieniać dwie wagi jednocześnie (tylko na +1 i +1, inne kombinacje daję się sprowadzić do tej kombinacji poprzez odpowiednie przekształcenia), choć wiąże się to ze znacznie wyższym kosztem obliczeniowym. W przeprowadzonych testach zmiana dwóch wag na raz była zawsze wystarczająca.

SMLP-VSS

Budujemy sieć SMLP o zwykłej architekturze, lecz z sigmoidalnymi funkcjami transferu, zarówno w neuronach ukrytych, jak i wyjściowych. Użycie funkcji sigmoidalnych pozwala na skorzystanie z błędu odległości, zamiast z błędu klasyfikacji, co często ułatwia proces uczenia sieci. Uczymy sieć przy pomocy algorytmu VSS, jednak używamy mniejszych kroków (mniejsze stałe c1, c2, c3, d1), niż przy VSS zastosowanym do standardowych sieci MLP, ponieważ po nauczeniu sieć osiągnie mniejsze wartości wag. Stosowana funkcja błędu ma postać:

$$E = \sum_v (out_v - desired_v)^2 + c\sum_i |w_i(w_i - 1)(w_i + 1)| \qquad (6)$$

gdzie do funkcji kwadratowej dodany jest człon regularyzacyjny wymuszający wartości wag równe –1, 0, lub +1. Jeżeli zmiana wartości wagi powoduje zbyt słabe zmniejszenie błędu średniokwadratowego, to na skutek działania członu regularyzacyjnego wartość tej wagi pozostanie zerowa. Po kilku cyklach treningowych wartości wszystkich wag ustalają się na -1±0.05, 0±0.5 lub +1±0.05. W ostatnim cyklu treningowym dodajemy człon regularyzacyjny dla biasu, podobny do członu w równaniu (6), który wymusza przyjęcie przez bias wartości dowolnej dodatniej liczby całkowitej mniejszej od ilości cech ±0.5. Po zakończeniu uczenia przekształcamy funkcje transferu z sigmoidalnych na skokowe i otrzymujemy identyczną sieć, jak po uczeniu metodą SMLP-DS.

Podsumowanie

Dokonano analizy powierzchni błędu sieci MLP popartej wizualizacją powierzchni błędu i analizy istotnych kierunków w przestrzeni wag sieci MLP. Zostały opracowane dwa algorytmy uczenia sieci MLP oparte na idei lokalnego przeszukiwania: gradient numeryczny (NG) i algorytm zmiennego kroku przeszukiwania (VSS). Oba algorytmy są bardzo proste i mają niewielkie wymagania pamięciowe. Osiągnięte wyniki są w pełni konkurencyjne dla metod opartych na gradientach analitycznych, często przewyższając je pod względem jakości osiągniętego rozwiązania przy jednocześnie efektywniejszym sposobie uczenia sieci. Szczególnie algorytm VSS cechuje się wyjątkowo dobrymi właściwościami odnoście tempa zbieżności i jakości znajdywanych rozwiązań.

Opracowano dwie metody uczenia sieci SMLP przeznaczonej do ekstrakcji reguł logicznych z danych: metodę bezpośredniego przeszukiwania i metodę opartą na zmodyfikowanym algorytmie VSS oraz przedstawiono możliwość dopasowania struktury sieci do struktury danych. Rozwiązania te cechują się prostym i szybkim w działaniu algorytmem zapewniającym zrozumiałą i prostą formę uzyskiwanego opisu danych przy pomocy reguł logicznych przy jednoczesnym poziomie dokładności klasyfikacyjnej porównywalnym z najlepszymi ze znanych klasyfikatorów.

Można zatem stwierdzić zgodnie z tezą pracy, że algorytmy oparte na przeszukiwaniu mogą być skutecznie zastosowanie do uczenia sieci MLP przeznaczonej do klasyfikacji danych oraz do ekstrakcji reguł logicznych z danych przy pomocy sieci MLP o specjalnie dobranej strukturze (SMLP).