ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1997
Seria: INFORMATYKA z. 33 Nr kol. 1377

Krzysztof CYRAN
Politechnika Śląska, Instytut Informatyki
Sławomir LETKIEWICZ, Piotr WOJCIECHOWSKI
Specjalistyczny Szpital nr 1 w Bytomiu, Oddział Urologii
Dariusz KOŁOCZEK
Politechnika Śląska, Instytut Informatyki

ZASTOSOWANIE SIECI NEURONOWEJ DO PROGNOZOWANIA WYLECZENIA CHORYCH Z NOWOTWOREM NEREK

Streszczenie. W artykule przedstawiono proces projektowania sieci neuronowej przeznaczonej do dawania rokowań odnośnie do trwałego wyleczenia pacjentów z nowotworem nerki. W szczególności opisano poszukiwanie sieci optymalnej pod względem jej architektury, parametrów użytych do jej uczenia oraz funkcji przejść neuronów. Omówiono również wady i zalety wielowymiarowej optymalizacji automatycznej i optymalizacji genetycznej.

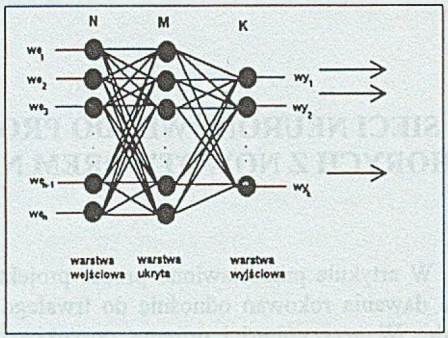
USE OF NEURAL NETWORK TO RECOVERY PROGNOSIS FOR PATIENTS WITH RENAL CANCER

Summary. In the article, we present the design of the neural network, used to predict the durable recovery for patients with renal cancer. In particular we describe process of looking for optimal network in terms of its architecture, training parameters and neuron transfer functions. We also discuss the advantages and disadvantages of multidimensional automatic optimization and genetic optimization.

1. Wprowadzenie

Sztuczne sieci neuronowe stanowią alternatywny wobec architektur von Neumanowskich paradygmat przetwarzania informacji. Podstawową ich zaletą jest możliwość zastosowania do rozwiązywania problemów, dla których dokładne algorytmy nie są znane. Wystarczy zgroma-

dzić wystarczająco dużo reprezentatywnych przykładów, by na ich podstawie wytrenować sieć. Typowym problemem, który nie daje się łatwo zalgorytmizować, jest określenie rokowań co do stanu zdrowia pacjentów, na podstawie zespołu wskaźników medycznych. Zastosowaniu sieci neuronowych do szczególnego przypadku, spośród tego typu zadań, poświęcona jest niniejsza praca. Ponieważ, w praktycznych zastosowaniach prognozujących prawie wyłącznie stosuje się architektury feed-forward z jedną warstwą neuronów ukrytych (rys.1), uczone metodą wstecznej propagacji błędu, skoncentrowano się na takich sieciach, czyniąc wyjątek tylko w metodzie uczenia, gdzie przy okazji rozpatrywania algorytmów genetycznych, w dwóch przypadkach, zrezygnowano ze szlifowania metodą back-propagation.

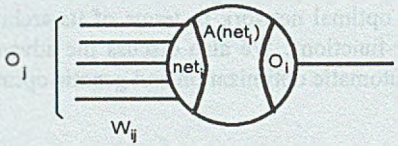


Rys.1. Sieć neuronowa typu feed-forward z jedną warstwą ukrytą
Fig. 1. Feed-forward neural network with one hidden layer

Przyjmując poniższe oznaczenia:

- net_i - pobudzenie sieciowe i-tego neuronu
- A_i - aktywacja i-tego neuronu
- O_i - wyjście i-tego neuronu
- w_{ij} - waga połączenia od neuronu j do i,

w sieciach tego typu, działanie el. przetwarzających (rys. 2.), zdefiniowane jest wzorem (4).



Rys. 2. Schemat neuronu
Fig. 2. Neuron scheme

$$net_i = \sum_j w_{ij} O_j \quad (1)$$

$$A_i = A\ (net_i) \quad (2)$$

$$O_i = TF\ (A_i) \quad (3)$$

zbierając:

$$O_i = TF\ (A\ (\Sigma_j w_{ij} O_j)) \quad (4)$$

2. Zdefiniowanie problemu

W ostatnich latach, w zachodnich czasopismach dedykowanych zastosowaniom informatyki w badaniach medycznych, pojawiło się sporo artykułów dotyczących powstania sieci neuronowych przydatnych w prognozowaniu w urologii (np. [1, 2, 3]). Mimo tego nie wyczerpują one wielu chorób, z jakimi styka się urolog w swej pracy, co więcej, w przypadku każdej choroby można postawić mnóstwo pytań (a sieć potrafi odpowiedzieć tylko na jedno konkretne, dla którego została przygotowana). Niektóre z nich wykorzystują do swej diagnozy zestawy wskaźników, nie oznaczane w Polsce. Ponadto sieci uczone w Stanach Zjednoczonych i dające tam doskonałe rezultaty co do trafności prognoz, nie muszą sobie radzić tak dobrze w Polsce, choćby ze względu na inne nawyki żywieniowe społeczeństwa, inny stopień zanieczyszczenia powietrza, inne uwarunkowania genetyczne pacjentów. Dlatego postanowiliśmy zaprojektować, nauczyć i zoptymalizować własną sieć, uczoną na przykładzie pacjentów leczonych w Specjalistycznym Szpitalu nr 1 w Bytomiu.

Pytanie, na które sieć miała odpowiadać, zostało zaproponowane przez specjalistę urologa, dr n. med. S. Letkiewicza, tak, by ewentualnie, w przypadku odpowiednio dużej trafności odpowiedzi, mogły one być pomocne w praktyce lekarskiej. Pytanie dotyczyć miało rokowań co do trwałego wyleczenia pacjentów z jasnokomórkowym nowotworem nerki. Uzyskanie wiarygodnej odpowiedzi na tak postawione pytanie, może wspomóc lekarza w jego ocenie stanu zdrowia pacjenta, dzięki czemu możliwy będzie wybór właściwszego sposobu dalszego leczenia, począwszy od obserwacji i nieszkodliwych leków, w wypadku osób, dla których rokowania są pomyślne, aż po zastosowanie leków z silnym działaniem ubocznym czy wręcz leczenia eksperymentalnego, u pacjentów, których choroba jest już b. mocno zaawansowana.

3. Przygotowanie danych

Do opisu stanu zdrowia pacjentów wybrano 13 danych wejściowych: wiek pacjenta (w latach), płeć (0-kobieta, 1-mężczyzna), faza T (1-4), faza N (0-3), faza M. (0-2), nefrektomia (0-brak, 1-była), przerzuty do płuc (0-brak, 1-były), przerzuty do kości (0-brak, 1-były), faza G (0-3), rozmiar guza (w centymetrach), chemioterapia (0-brak, 1-była), hormonoterapia (0-brak, 1-była), radioterapia (0-brak, 1-była)

Wartością wyjściową stanowiła dwuwartościowa informacja o tym, czy dana osoba została trwale wyleczona, czy też nie, zdobyta z perspektywy historycznej. Próby bardziej

szczegółowego stopniowania odpowiedzi sieci natrafiały na przeszkodę w postaci braku jednoznacznych kryteriów oceny, a co za tym idzie, przenosiłyby do wzorców uczących wiedzę eksperta, zamiast rzeczywistych faktów.

Tak zaprojektowane dane zebrano dla 113 osób. Podzielono je na trzy zbiory: uczący (83 osoby), testowy (9 osób), testowy rozszerzony (9 osób z testowego + 21 osób). Potrzeba stworzenia trzech zbiorów, a nie tylko uczącego i testowego, jak zwykle przyjmuje się przy uczeniu sieci, wyjaśniona jest dokładniej w rozdz.5.5., a wynikła ona z dążenia do zapewnienia możliwie obiektywnych warunków testowych.

3.1. Normalizacja i reprezentacja danych

Dane te należało poddać procesowi transformacji do danych akceptowalnych przez elementy wejściowe sieci neuronowej. W rozpatrywanej sieci neurony wejściowe nie stanowią elementów przetwarzających, lecz tylko buforują podawane dane (posiadają liniową funkcję przejścia o współczynniku liniowym równym 1). Ponieważ wyjście każdego neuronu przyjmuje wartości z zakresu [0,1] (dla niektórych funkcji przejścia przedział ten jest otwarty), więc by spełniony był ten warunek dla neuronów warstwy wejściowej, należy ograniczyć zakres zmienności danych wejściowych również do przedziału [0,1]. W przypadku wejścia oznaczającego wiek pacjenta, należałoby teoretycznie przyjąć zakres zmienności od 0 do np. 120 (aby w przyszłości nie zdarzył się przypadek, że wiek pacjenta znajdzie się poza zakresem). Jednakże ustalenie tak szerokiego zakresu zmienności i liniowe znormalizowanie go do przedziału [0,1] spowodowałoby słabe rozróżnianie wieku pacjentów najczęściej cierpiących na tę chorobę. Lepszym rozwiązaniem jest przyjęcie zakresu zmienności na podstawie wartości minimalnej i maksymalnej ze zbioru uczącego i obcinanie do jednej ze skrajnych wartości późniejszego ewentualnego przekroczenia zakresu. Jest to dopuszczalne, gdyż sieć rozpozna wówczas wiek, spoza zakresu, jako wiek skrajny, dla którego potrafi dawać właściwą odpowiedź, a różnice wieku u najczęściej badanych pacjentów też będą dla niej dość dobrze widoczne. (Oczywiście, jako przekształcenie normalizujące, można by przyjąć i funkcję nieliniową, ale z opisanych w literaturze przykładów wynika jasno, że normalizacja liniowa z dyskryminacją rzadko pojawiających się przypadków spoza zakresu daje dobre rezultaty [5]). Analogiczne rozważania można przeprowadzić dla neuronu odczytującego rozmiar guza.

W przypadku pozostałych wejść istnieją jasno określone granice ich zmienności, dlatego sposób normalizacji liniowej jest tutaj oczywisty. Wejścia binarne (np. typu płeć), w ogóle nie wymagają normalizacji, gdyż ich wartości odpowiadają dopuszczalnym skrajnym wartościom wyjściowym neuronów warstwy wejściowej [0,1].

3.2. Reprezentacja danych brakujących

W praktyce medycznej nierzadko zdarza się, że trzeba podjąć decyzję odnośnie do dalszego leczenia pacjenta nie dysponując wszystkimi wskaźnikami charakteryzującymi jego stan zdrowia. Tymczasem sieć neuronowa musi mieć podane jakieś wartości na wszystkie jej wejścia. Pojawia się problem reprezentacji danych brakujących. W tym celu trzeba wprowadzić dodatkowy neuron skojarzony z neuronem, dla którego przewidujemy możliwość zaistnienia braku jakiejkolwiek wartości. Dla istniejących wartości, na neuron ten podawać należy wartości ze znakiem przeciwnym. W ten sposób sieć nauczy się, iż neurony te mają przeciwny wpływ na wynik. Jeżeli teraz, przy braku wartości, na obydwa neurony podamy wartości dla każdego z nich minimalne, to ich wpływ na odpowiedź sieci zostanie w ten sposób zniesiony, a sieć odpowie na podstawie wartości podanych na pozostałe wejścia. Takimi neuronami, dla których wartości mogą nie być znane, są: T, N, M, przerzuty do płuc, przerzuty do kości, G, rozmiar guza. Po dodaniu neuronów skojarzonych, rozmiar warstwy wejściowej wzrósł do 20 elementów buforujących.

3.3. Sprawdzanie wiarygodności odpowiedzi

Sieć neuronowa, stosowana do tak odpowiedzialnej pracy, jaką jest wspomaganie decyzji lekarza ratującego ludzkie życie, powinna się charakteryzować mechanizmem pozwalającym ocenić wiarygodność generowanej przez nią odpowiedzi. Mechanizm ten można zaimplementować przez dodanie neuronu wyjściowego o wzorcu uczącym przeciwnym do podstawowego. Ponieważ sieć nie jest nigdy uczona przy tolerancji błędu równej zero, a ponadto, początkowe jej wartości w synapsach są przypadkowe (a nie równe), proces uczenia obu neuronów przebiega nieco odmiennie, a w konsekwencji wartości wyjściowe na obu neuronach, mają zbliżone, lecz **nie identyczne** wartości absolutne przy przeciwnych znakach. Dla danych testowych, których sieć nie „widziała", spowoduje to, co prawda, pogorszenie współczynnika prawidłowych odpowiedzi (gdyż za błędną uważana jest odpowiedź, gdy dowolny z dwóch neuronów nie zmieści się w przedziale określonym przez tolerancję testową), ale w zamian za to uzyskujemy możliwość oceny wiarygodności odpowiedzi w czasie działania sieci, gdy nie będą znane odpowiedzi prawidłowe. Jeżeli różnica w wartościach bezwzględnych będzie duża, znaczy to że sieć „zgaduje", bo jakoś musi odpowiedzieć na każde pobudzenie. Jeżeli odpowiedzi są zbliżone co do modułu, jest to oznaką, że odpowiedź sieci można przyjąć za dużo lepiej uzasadnioną. Oczywistym sposobem zwiększenia możliwości oceny wiarygodności rokowań sieci jest również wytrenowanie wielu sieci i porównanie ich odpowiedzi. W zastosowaniach medycznych, gdzie czas nie odgrywa decydującej roli, a wiarygodność tak, należałoby skorzystać z obu tych metod (w istocie swej podobnych).

4. Uczenie sieci

Uczenie sieci metodą wstecznej propagacji błędu polega na poszukiwaniu sieci z minimalnym funkcjonałem błędu średniokwadratowego, dla całego ciągu uczącego. Jeżeli przez T oznaczymy odpowiedź prawidłową, a przez O odpowiedź rzeczywistą sieci, to dla i-tego neuronu i p-tego wzorca uczącego, połowa kwadratu błędu (zwana dalej dla błędem E) wynosi:

$$E_i^{(p)} = \frac{1}{2}(T_i^{(p)} - O_i^{(p)})^2 \tag{5}$$

Całkowity błąd E dla p-tego wzorca:

$$E^{(p)} = \sum_i E_i^{(p)} = \frac{1}{2}\sum_i (T_i^{(p)} - O_i^{(p)})^2 \tag{6}$$

Całkowity błąd E dla całego ciągu uczącego:

$$E = \sum_p E^{(p)} = \frac{1}{2}\sum_p \sum_i (T_i^{(p)} - O_i^{(p)})^2 \tag{7}$$

Trenowanie sieci, aby asocjowała wejście i wyjście, jest zatem procesem minimalizacji błędu E, gdzie jako zmienne niezależne występują wagi w_{ij}. Ponieważ nawet najmniejsze sieci mają setki połączeń i skojarzonych z nimi wag, jest to proces minimalizacji pola skalarnego nad przestrzenią wektorową o setkach wymiarów. Jedną z metod znajdowania minimum takiego pola jest metoda spadku gradientu. Znajduje ona jednak minima lokalne.:

$$\Delta w_{ij} = -\eta \frac{\delta E}{\delta w_{ij}}, \tag{8}$$

Zgodnie z metodą spadku gradientu wagi należałoby modyfikować po podaniu całego ciągu uczącego. W praktyce częściej wagi modyfikuje się po każdym wzorcu uczącym, co w większości przypadków daje lepsze rezultaty, mimo że w ten sposób nie spełnienia się dokładnie metody spadku gradientu. W pracy badano modyfikację zarówno po każdym wzorcu, jak i po podaniu całego ich ciągu. Przy modyfikacji po każdym wzorcu, mamy zmianę wagi po podaniu p-tego wzorca równą:

$$\Delta w_{ij}^{(p)} = -\eta \frac{\delta E^{(p)}}{\delta w_{ij}} = -\eta \frac{\delta E_i^{(p)}}{\delta w_{ij}} \tag{9}$$

Oznaczając przez:

$$\delta_i^{(p)} = -\frac{\delta E_i^{(p)}}{\delta A_i^{(p)}} \tag{10}$$

dostajemy:

$$\Delta w_{ij}^{(p)} = \eta\ \delta_i^{(p)} O_j^{(p)} \tag{11}$$

Dla neuronów warstwy wyjściowej, jest spełniona zależność:

$$\delta_i^{(p)} = TF'(A_i^{(p)})(T_i^{(p)} - O_i^{(p)}) \tag{12}$$

Dla neuronów warstw ukrytych nieznane jest $T_i^{(p)}$, dlatego rzutuje się wstecz błędy z warstwy następnej względem analizowanej:

$$\frac{\delta E^{(p)}}{\delta O_i^{(p)}} = \sum_k \frac{\delta E_k^{(p)} \delta A_k^{(p)}}{\delta A_k^{(p)} O_i^{(p)}} = -\sum_k \delta_k^{(p)} w_{ki}^{(p)} \tag{13}$$

czyli:

$$\delta_i^{(p)} = TF'(A_i^{(p)}) \sum_k \delta_k^{(p)} w_{ki}^{(p)} \quad (14)$$

W badaniach używano tego algorytmu rozwiniętego o współczynnik wygładzający μ, powodujący, że w bieżącym kroku uwzględniano modyfikację w kroku poprzednim:

$$\Delta w_{ij}^{(p)} = \eta \left((1 - \mu)\delta_i^{(p)} O_j^{(p)} + \mu \Delta w_{ij}^{(p-1)}\right) \quad (15)$$

Uwzględniono również przypadek szczególny, gdy $\mu = 0$, i wtedy (15) przechodzi w (11).

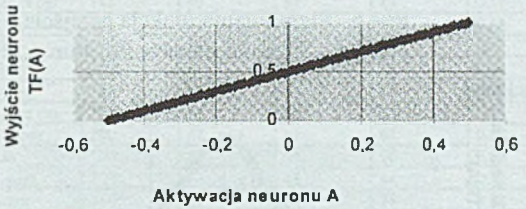
5. Optymalizacja sieci

Poniższe podrozdziały opisują poszukiwanie sieci możliwie najlepiej oddającej istniejące odwzorowanie wartości wejściowych w wyjściową.

5.1. Funkcje przejścia neuronów

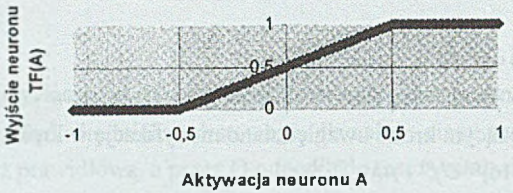
Pierwszym krokiem było poszukiwanie funkcji przejścia TF elementów przetwarzających. Rozpatrywano następujące funkcje:

- funkcja liniowa (rys. 3). Dla funkcji tej konieczne było sprawdzanie, czy potencjalna aktywacja neuronu A (patrz (1) i (2)) będzie się mieścić w zakresie [-0.5, 0.5], gdyż tylko dla tego zakresu określona jest funkcja przejścia. W wypadku przeciwnym trzeba proporcjonalnie zmniejszać wagi wszystkich neuronów w warstwie. Dla liniowej funkcji przejść zmiany takie są dopuszczalne.



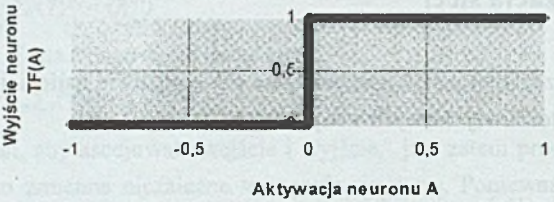
Rys. 3. Liniowa funkcja przejść
Fig. 3. Linear transfer function

- funkcja liniowa z ograniczeniem (rys. 4). Ta i kolejne funkcje są funkcjami nieliniowymi, określonymi na zbiorze $\Re$, dlatego aktywacja mogła być dowolna.



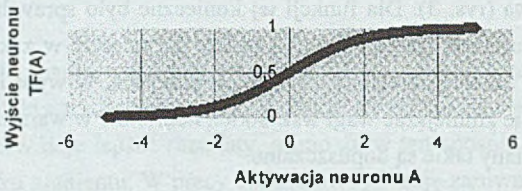
Rys. 4. Liniowa funkcja przejść z ograniczeniem
Fig. 4. Linear threshold transfer function

- funkcja schodkowa (rys. 5)



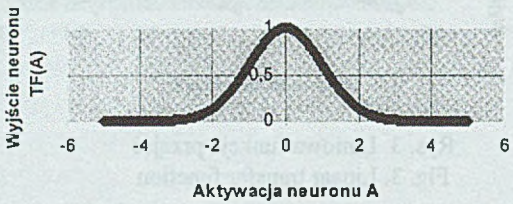
Rys. 5. Schodkowa funkcja przejść
Fig. 5. Step transfer function

- funkcja sigmoidalna (rys. 6)



Rys.6. Sigmoidalna funkcja przejścia
Fig.6. Sigmoid transfer function

- funkcja Gaussa (rys. 7)



Rys. 7. Funkcja przejścia Gaussa
Fig. 7. Gaussian transfer function

Przebadano wszystkie możliwe kombinacje tych typów funkcji przejść dla warstwy neuronów ukrytych oraz warstwy wyjściowej. Dało to w sumie 25 sieci neuronowych (z 20 neuronami warstwy ukrytej), trenowanych przy współczynniku uczenia 1, współczynniku wygła-

dzającym 0.9, tolerancji uczenia 0.1. Ograniczono liczbę przebiegów uczących do 1000. Modyfikacji wag dokonywano po każdej próbce. W badaniu nie uwzględniono mechanizmu sprawdzania wiarygodności odpowiedzi. Następnie poddano sieci próbie na zbiorze testowym, przy tolerancji testowej równej: 0.4, 0.2 i 0.1. Wyniki testów prezentuje tabela 1. Dla funkcji liniowej z ograniczeniem, w algorytmie uczenia użyto przybliżenia przez funkcję sigmoidalną o tej samej pochodnej w punkcie A=0. Dla funkcji schodkowej liczono pochodną przybliżającej ją funkcji sigmoidalnej, dla której styczna w punkcie zero miała współczynnik liniowy równy 100. Z przedstawionej tabeli widać, iż ocena jakości sieci uzależniona jest od wyboru czynnika klasyfikującego. Tylko sieć sigmtres jest najlepsza zarówno pod względem błędu średniokwadratowego, średniego, i ilości prawidłowych rokowań dla trzech różnych tolerancji testowych. Ponadto widać, iż sieci trestres, steptres, linesigm, linetres i linestep bardzo dobre pod względem poprawności rokowań w zbiorze testowym, nie dały się nauczyć bezbłędnie zbioru uczącego. Ze względu na prawidłowe odpowiedzi w zbiorze testowym, zostały jednak wybrane do dalszych badań. Ponadto dalszej analizie poddano sieć sigmsigm zajmującą drugie miejsce pod względem błędu średniokwadratowego i średniego oraz stepsigm ze względu na dobrą ilość poprawnych rokowań w zbiorze test. przy 100% poprawności w zbiorze uczącym.

Tabela 1

Wyniki uzyskane przez sieci o różnych funkcjach przejścia

Nazwa sieci	TF warstwy ukrytej	TF warstwy wyjściowej	Ilość przebiegów uczących	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.4	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.2	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.1	Błąd średniokwadratowy	Błąd średni	% poprawnych odpowiedzi dla zbioru uczącego przy tolerancji 0.1
sigmsigm	sigm.	Sigm.	103	89	89	67	0.16	0.1	100
sigmtres	sigm.	Lin. z ogr.	142	89	89	89	0.15	0.05	100
sigmstep	sigm.	Schodk.	40	67	67	67	0.58	0.33	100
sigmline	sigm.	Lin.	1000	56	44	11	0.59	0.45	48
sigmgaus	sigm.	Gauss.	1000	44	44	44	0.75	0.56	43
tressigm	lin. z ogr.	Sigm.	124	67	67	56	0,39	0,23	100
trestres	lin. z ogr.	Lin. z ogr.	1000	89	89	89	0.33	0.11	82
tresstep	lin. z ogr.	Schodk.	200	67	67	67	0.58	0.33	100
tresline	lin. z ogr.	Lin.	1000	56	56	56	0.62	0.45	5
tresgaus	lin. z ogr.	Gauss.	1000	44	44	44	0.75	0.55	43
stepsigm	schodk.	Sigm.	380	89	78	78	0.32	0.14	100
steptres	schodk.	Lin. z ogr.	1000	89	89	89	0.33	0.11	88
stepstep	schodk.	Schodk.	1000	67	67	67	0.58	0.33	84
stepline	schodk.	Lin.	1000	56	0	0	0.50	0.49	6
stepgaus	schodk.	Gauss.	1000	44	44	44	0.75	0.56	43
linesigm	lin.	sigm.	1000	89	89	89	0.25	0.09	82
linetres	lin.	lin. z ogr.	1000	89	89	89	0.33	0.11	89
linestep	lin.	schodk.	1000	89	89	89	0.33	0.11	90
lineline	lin.	lin.	1000	44	44	44	0.88	0.78	46
linegaus	lin.	gauss.	1000	44	44	44	0.74	0.56	43
gaussigm	gauss.	Sigm.	73	56	44	11	0.41	0.35	100
gausstres	gauss.	Lin.z ogr.	1000	78	78	78	0.31	0.14	65
gausstep	gauss.	Schodk.	25	33	33	33	0.82	0.67	100
gausline	gauss.	Lin.	1000	11	0	0	0.52	0.51	10
gausgaus	gauss.	Gauss.	1000	44	44	44	0.75	0.56	43

5.2. Momenty modyfikacji wag

W dalszych badaniach, do sieci z wybranymi typami neuronów dodano mechanizm sprawdzania wiarygodności odpowiedzi. Przy zachowaniu pozostałych parametrów uczono sieci modyfikując wagi po każdej próbce i po całym przebiegu. Otrzymane wyniki ilustruje tabela 2. Modyfikacje po całym zbiorze uczącym okazały się lepsze tylko dla sieci sigmsigm. Rezultaty zbliżone miały sieci steptres, natomiast w przypadku sieci stepsigm o kolejności decyduje wybór kryterium. W pozostałych pięciu sieciach modyfikacja po podaniu próbki dawała rezultaty lepsze lub dużo lepsze niż po całym zbiorze uczącym. Z sieci ze wszystkimi odpowiedziami poprawnymi dla zbioru uczącego, najlepszą w testach pod względem błędu średniokwadratowego i średniego okazała się sigmtres modyfikowana po próbce, drugą była sieć sigmsigm modyfikowana na końcu zbioru, trzecią sigmsigm, dla której zmieniano wagi po próbce i czwartą sigmtres modyfikowana po całym ciągu. Sieć trestres, rewelacyjna w testach, tylko w 77% odpowiadała poprawnie dla zbioru uczącego.

Tabela 2.

Wyniki sieci modyfikowanych po każdej próbce i po całym ciągu

Nazwa sieci	Czas modyfikacji	Ilość przebiegów uczących	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.4	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.2	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.1	Błąd średniokwadratowy	Błąd średni	% poprawnych odpowiedzi dla zbioru uczącego przy tolerancji 0.1
sigmsigm	próbka	94	78	78	56	0.35	0.19	100
sigmsigm	ciąg	476	100	67	67	0.17	0.10	100
sigmtres	próbka	94	100	89	89	0.10	0.03	100
sigmtres	ciąg	560	78	78	78	0.40	0.16	100
trestres	próbka	1000	100	100	100	0	0	77
trestres	ciąg	1000	56	56	56	0.56	0.33	76
stepsigm	próbka	225	67	67	67	0.55	0.32	100
stepsigm	ciąg	1000	89	89	67	0.27	0.12	57
steptres	próbka	1000	89	89	89	0.33	0.11	81
steptres	ciąg	1000	89	89	89	0.33	0.11	84
linesigm	próbka	1000	89	89	89	0.21	0.08	78
linesigm	ciąg	1000	0	0	0	0.71	0.50	1
linetres	próbka	1000	89	89	89	0.33	0.11	75
linetres	ciąg	1000	0	0	0	0.71	0.50	0
linestep	próbka	1000	89	89	89	0.24	0.06	88
linestep	ciąg	1000	0	0	0	0.71	0.50	4

5.3. Testowanie podczas uczenia

Dalsze badanie doprowadzić miało do znalezienia optymalnej liczby neuronów warstwy ukrytej. W tym celu zastosowano testowanie wplecione w cykl uczenia. Po każdym przebiegu uczącym zapisywano statystyczne wskaźniki dla tego przebiegu, jak również dla zbioru testowego. Późniejsza analiza tych danych doprowadzić miała do wyboru sieci najlepiej generalizującej. Badaniom poddano już tylko sieci sigmsigm i sigmtres.

5.3.1. Strojenie tolerancji uczenia

Dla obu typów sieci zastosowano strojenie tolerancji uczenia od wartości 0.2 do 0.05 oraz od 0.2 do 0.1. Po osiągnięciu bezbłędnego przebiegu uczącego, zmniejszano dotychczasową tolerancję, mnożąc ją przez czynnik 0.8.

5.3.2. Poszukiwanie optymalnej liczby neuronów

Uczenie rozpoczęto dla 20 neuronów warstwy ukrytej. Gdy sieć osiągnęła bezbłędny przebieg dla końcowej tolerancji, odejmowano z jej warstwy ukrytej jeden neuron, po czym uczono dalej do kolejnego osiągnięcia przebiegu bezbłędnego. Koniec uczenia ustalono na bezbłędny przebieg przy 5 neuronach ukrytych. Górna granica wynikła z doświadczalnego wzoru na maksymalną liczbę neuronów ukrytych (16), zaś dolna jest połową optymalnej liczby według innej reguły (17):

$$H = \frac{1}{2}F - I - O \tag{16}$$

$$H = \frac{1}{2}(I + O) \tag{17}$$

gdzie: H - ilość neuronów warstwy ukrytej, F - ilość faktów w zbiorze uczącym, I- ilość neuronów w warstwie wejściowej, O - ilość neuronów w warstwie wyjściowej.

Z przeanalizowania otrzymanych statystyk wynikło, iż optymalną liczbą neuronów ukrytych jest liczba 11, co idealnie potwierdza regułę 17. Jeśli zaś chodzi o porównanie sieci typu sigmsigm oraz sigmtres, to ta druga miała lepsze wskaźniki. Powodem tego jest fakt, że wzorzec uczący przyjmował wartości binarne, a funkcja liniowa z ograniczeniem w zakresie nasycenia ma również tylko dwie wartości. Wynika stąd wniosek, że w celu rozwiązywania zagadnień, w których wartości wyjściowe w sposób naturalny są binarne, należy uwzględniać jako funkcję przejścia funkcję liniową z ograniczeniem. W przypadku jednak przez nas rozpatrywanym, binarny charakter wzorca uczącego jest sztuczny. Przyjęto go tylko z braku obiektywnych możliwości rozróżnienia wartości pośrednich. Dlatego, choć sieci o obu warstwach sigmoidalnych odpowiadały nieco gorzej, to one wybrane zostały do dalszej optymalizacji. W nich bowiem ma sens stosowanie mechanizmu sprawdzania wiarygodności odpowiedzi, gdyż różnice w aktywacji neuronu podstawowego i sprawdzającego przenoszone są na ich wyjście i są możliwe do obserwacji. W wypadku sieci sigmtres, pracującej w nasyceniu, otrzymuje się zgodność obu neuronów, choć ich aktywacje mogą się znacznie różnić. Dawałoby to złudzenie wiarygodności w zupełnie nieuzasadnionych przypadkach,dlatego w tym konkretnym zastosowaniu, w którym sprawdzenie wiarygodności odgrywa podstawową rolę, trzeba było z sieci tego typu zrezygnować, mimo ich lepszych wskaźników statystycznych.

5.4. Współczynniki uczenia

Sieć o architekturze sigmsigm z 11 neuronami ukrytymi, uczono z kolei używając różnych wartości współczynników η i μ (15). Wyniki zamieszczono w tabeli 3.

Tabela 3

Wyniki sieci uczonych z różnymi wartościami η i μ

Nazwa sieci	μ	η	Ilość przebiegów uczących (max 5000)	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.4	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.2	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.1	Błąd średnio-kwadratowy	% poprawnych odpowiedzi dla zbioru uczącego przy tolerancji 0.1
mi00et01	0	0.1	1137	78	67	44	0.25	100
mi00et02	0	0.2	522	89	67	44	0.23	100
mi00et04	0	0.4	267	89	67	44	0.22	100
mi00et08	0	0.8	152	89	56	44	0.21	100
mi00et16	0	1.6	95	89	67	56	0.23	100
mi00et32	0	3.2	89	89	89	78	0.26	100
mi00et48	0	4.8	1830	100	89	89	0.08 !!!	100
mi00et64	0	6.4	5000	89	89	78	0.18	82
mi01et02	0.1	0.2	515	78	67	44	0.22	100
mi01et04	0.1	0.4	267	78	67	44	0.22	100
mi01et08	0.1	0.8	166	89	89	44	0.19	100
mi01et16	0.1	1.6	97	67	67	44	0.29	100
mi01et32	0.1	3.2	90	78	78	67	.30	100
mi01et64	0.1	6.4	2098	89	89	89	0.29	100
mi01et77	01	7.7	5000	67	67	44	0.31	80
mi02et04	0.2	0.4	262	89	78	44	0.21	100
mi02et08	0.2	0.8	167	89	89	56	0.17	100
mi02et16	0.2	1.6	97	67	67	44	0.28	100
mi02et32	0.2	3.2	90	78	78	67	.30	100
mi02et64	0.2	6.4	2291	89	89	89	0.16 !!!	100
mi02et77	0.2	7.7	5000	100	100	89	0.07	78
mi04et04	0.4	0.4	264	89	78	44	0.20	100
mi04et08	0.4	0.8	162	89	78	44	0.19	100
mi04et16	0.4	1.6	100	78	67	44	0.26	100
mi04et32	0.4	3.2	96	56	44	0	0.3	100
mi04et64	0.4	6.4	2231	89	89	89	0.22 !!!	100
mi04et77	0.4	7.7	5000	100	100	89	0.06	82
mi08et08	0.8	0.8	230	89	67	56	0.19	100
mi08et16	0.8	1.6	131	89	67	56	0.21	100
mi08et32	0.8	3.2	91	89	78	44	0.21	100
mi08et64	0.8	6.4	695	100	100	100	0.02 !!!	100
mi08et77	0.8	7.7	5000	1000	89	78	0.13	90
mi09et16	0.99	1.6	463	78	67	67	0.32	100
mi09et32	0.99	3.2	859	100	100	89	0.06 !!!	100
mi09et64	0.99	6.4	3165	100	89	89	0.08 !!!	100
mi09et77	0.99	7.7	2470	78	78	78	0.36	100

Z powyższej tabeli widać, że najlepsze rezultaty osiągano przy współczynniku uczenia 6.4. Jego przekroczenie już do 0.77 wszędzie powodowało brak zbieżności, zaś wartość poniżej 3.2 gorsze rezultaty uczenia. Natomiast wartość współczynnika wygładzenia nie miała istotniejszego wpływu na proces uczenia.

5.5. Wielowymiarowa optymalizacja automatyczna

Definiując wartości minimalne, maksymalne oraz przyrost dla różnych parametrów uczenia i architektury sieci, otrzymujemy pewną przestrzeń dyskretną, której wymiarami są po-

szczególne parametry. Zdefiniujmy nad tą przestrzenią, jako funkcję celu, funkcję daną wzorem:

$$f = 1 - \frac{\sqrt{\sum_p \sum_i (T_i^{(p)} - O_i^{(p)})^2}}{N_o N_f} \tag{18}$$

gdzie: N_o- liczba neuronów wyjściowych,

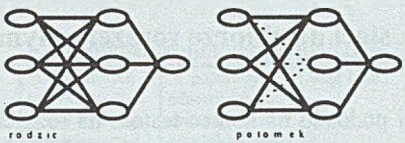
N_f- liczba próbek w zbiorze testowym,

(i zmienia się od 1 do N_o, p zmienia się od 1 do N_f).

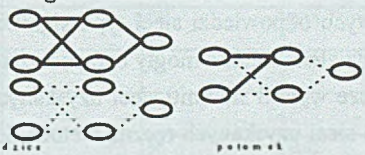
Obliczając wartość funkcji f dla wszystkich punktów przestrzeni, i pamiętając wyniki, możemy znaleźć zestaw parametrów, przy których sieć generuje najmniejszy błąd średniokwadratowy dla zbioru testowego. Stosując jednak tę metodę należy zwrócić uwagę, iż wykorzystuje się zbiór testowy do wyboru najlepszej sieci. W ten sposób dane testowe stają się danymi, które zadecydowały o wyborze konkretnej sieci z setek innych, przez co ich wiarygodność, jako testu, maleje. Dlatego utworzono dodatkowy zbiór testowy rozszerzony, w którym oprócz danych ze zbioru testowego, dodano dane o innych pacjentach, które to dane nie zostały wykorzystane w żaden sposób przy wyborze najlepszej sieci. O tym, że było to konieczne, niech świadczy, że błąd średniokwadratowy najlepszej sieci (o nazwie bestopt), dla zbioru testowego wyniósł zaledwie 0.02 i osiągnięto 100% zgodność odpowiedzi przy tolerancji testowej 0.1, podczas gdy dla zbioru testowego rozszerzonego błąd średniokwadratowy wynosił aż 0.33, a zgodność odpowiedzi dla tej samej tolerancji zmalała do 83% (tabela 6.). W optymalizacji tej jako jeden z wymiarów wybrano wielkość warstwy neuronów ukrytych (od 7 do 13). Sieć wybrana jako najlepsza posiada ich 11, potwierdzając prawidłowość wyboru opisanego w rozdz. 5.3.2.

5.6. Optymalizacja genetyczna

Odmienne podejście do problemu optymalizacji sieci, ale przy ustalonej jej architekturze, jest możliwe przy zastosowaniu algorytmów genetycznych.



Rys. 3. Operacja mutacji sieci neuronowej
Fig. 3. Mutation of neural network



Rys. 4. Operacja krzyżowania sieci neuronowej
Fig. 4. Crossover operation of neural network

Informacja w sieci, zapisana w postaci wartości synaps, ma dwa wymiary, natomiast algorytmy genetyczne operują na jednowymiarowych ciągach kodowych [8]. Naturalne wydaje się skojarzenie wag dochodzących do danego neuronu z samym neuronem (gdyż to ich suma ważona decyduje o zachowaniu neuronu) i traktowanie sieci jako ciągów kodowych, których elementami są neurony wraz ze „swymi" wagami. Rysunki 3 i 4 przedstawiają operację genetyczną mutacji oraz krzyżowania w odniesieniu do sieci neuronowych. Wyniki optymalizacji genetycznej zamieszczono w tabeli 4. Na uwagę zasługuje fakt, iż przy odpowiednio dużej liczbie pokoleń, parametry procesu ewolucji praktycznie nie mają wpływu na jakość wygenerowanych sieci (por. zwłaszcza tabelę 6). Sieci gen4 i gen5 powstanie swoje zawdzięczają tylko czystemu „doborowi naturalnemu", gdyż początkowe wartości ich wag miały wartości przypadkowe, a ponadto zrezygnowano z tzw. szlifowania sieci potomnych algorytmem z wsteczną propagacją błędów. W tym przypadku bardziej adekwatnie byłoby nazwać proces treningiem genetycznym, a nie li tylko optymalizacją. Zdumiewający był fakt dobrej jakości generowanych odpowiedzi dla tych sieci oraz to, iż jakość ta praktycznie nie zależała od drastycznie różnych parametrów krzyżowania. Sieć gen1 powstała na bazie przeciętnie prognozującej sieci, a sieć gen3 z najlepszej otrzymanej w wyniku wielowymiarowej optymalizacji automatycznej.

Tabela 4

Wyniki optymalizacji genetycznej

Nazwa sieci	Występuje wytrenowanie	Ilość przebiegów szlifujących	Ilość pokoleń	Współczynnik mutacji [%]	Współczynnik krzyżowania [%]	Początkowy błąd średniokwadratowy dla zbioru uczącego	Końcowy błąd średniokwadratowy dla zbioru uczącego	Końcowy błąd średniokwadratowy dla zbioru testowego	Końcowy błąd średni dla zbioru testowego
gen1	tak	50	6000	2	50	0.049	0.014	0.10	0.05
gen2	nie	25	12000	3	50	0.58	0.005	0.09	0.05
gen3	tak	50	6000	2	50	0.024	0.010	0.06	0.03
gen4	nie	0	20000	3	50	0.58	0.012	0.26	0.10
gen5	nie	0	20000	5	0	0.58	0.004	0.27	0.15
gen6	nie	25	12000	5	50	0.58	0.004	0.15	0.07
gen7	nie	25	12000	10	50	0.58	0.003	0.08	0.04

6. Testy wybranych sieci na zbiorze rozszerzonym

Powstałe wcześniej sieci poddano na koniec testom na rozszerzonym zbiorze. Poniższe tabele zawierają wyniki tych testów. Uwagę zwraca ogólnie lepsza zdolność do generalizacji sieci z 11 neuronami ukrytymi niż z 20 neuronami ukrytymi. Ponadto potwierdzona została lepsza zdolność do prawidłowych odpowiedzi sieci typu sigmtres od sigmsigm, jednakże z powodów rozważanych w rozdziale 5.3.2. nie mogły być one użyte w praktyce.

Tabela 7 prezentuje zbiorcze wyniki siedmiu sieci uzyskanych metodą optymalizacji genetycznej, sześciu najlepszych sieci uzyskanych ręczną metodą doboru różnych współczynników uczenia (przy tym doborze kierowano się wynikami dla parametrów wcześniej przetestowanych) oraz sieci otrzymanej w wyniku automatycznej optymalizacji.

Tabela 5

Wyniki poszerzonego testu dla sieci z 20 neuronami ukrytymi

Nazwa sieci	Moment modyfikacji	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.4	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.2	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.1	Błąd średni	Błąd średnio-kwadratowy	% poprawnych odpowiedzi dla zbioru uczącego przy tolerancji 0.1
linesigm	ciąg	3	3	3	0.48	0.70	1
linestep	ciąg	0	0	0	0.53	0.73	4
linetres	ciąg	0	0	0	0.50	0.70	0
sigmsigm	ciąg	83	70	70	0.18	0.36	100
sigmtres	ciąg	83	83	83	0.13	0.35	100
stepsigm	ciąg	90	90	70	0.12	0.28	57
steptres	ciąg	97	97	97	0.03	0.18	84
trestres	ciąg	50	50	50	0.32	0.56	76
linesigm	próbka	90	90	90	0.09	0.28	78
linestep	próbka	90	90	90	0.08	0.29	88
linetres	próbka	90	90	90	0.10	0.32	75
sigmsigm	próbka	73	73	67	0.22	0.42	100
sigmtres	próbka	87	80	80	0.15	0.35	100
stepsigm	próbka	77	77	77	0.19	0.41	100
steptres	próbka	83	83	83	0.10	0.31	81
trestres	próbka	97	97	97	0.03	0.18	77

Tabela 6

Wyniki poszerzonego testu dla najlepszych sieci z 11 neuronami ukrytymi

Nazwa sieci	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.4	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.2	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.1	Błąd średni	Błąd średnio-kwadratowy	% poprawnych odpowiedzi dla zbioru uczącego przy tolerancji 0.1
gen1	87	80	77	0.14	0.31	100
gen2	87	77	77	0.13	0.30	100
gen3	83	83	80	0.15	0.35	100
gen4	83	80	77	0.14	0.33	100
gen5	83	83	77	0.11	0.28	100
gen6	80	73	73	0.19	0.40	100
gen7	90	87	83	0.09	0.26	100
bestopt	87	83	83	0.13	0.33	100
mi00et48	93	87	83	0.09	0.25	100
mi02et64	93	93	90	0.06	0.20	100
mi04et64	90	87	87	0.11	0.29	100
mi08et64	87	87	87	0.13	0.34	100
mi09et32	83	80	77	0.19	0.40	100
mi09et64	97	93	93	0.05	0.18	100

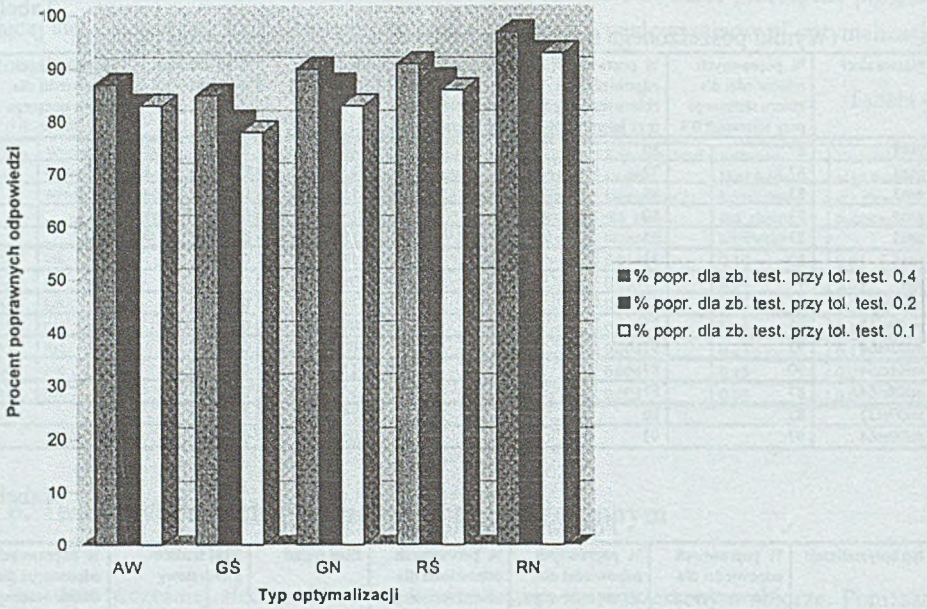
Tabela 7

Porównanie metod optymalizacji

typ optymalizacji	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.4	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.2	% poprawnych odpowiedzi dla zbioru testowego przy tolerancji 0.1	Błąd średni	Błąd średnio-kwadratowy	% poprawnych odpowiedzi dla zbioru uczącego przy tolerancji 0.1
automatyczna wielowymiarowa AW	87	83	83	0.13	0.33	100
genetyczna średnia GS	85	80	78	0.14	0.32	100
genetyczna najlepsza GN	90	87	83	0.09	0.26	100
ręczna średnia RS	91	88	86	0.11	0.28	100
ręczna najlepsza RN	97	93	93	0.05	0.18	100

7. Wnioski

W tabeli 7 widać, że wyniki sieci bestopt otrzymanej w wyniku optymalizacji wielowymiarowej oraz średnie z wyników siedmiu sieci otrzymanych na drodze optymalizacji genetycznej są w zasadzie takie same (współczynniki poprawności rokowań minimalnie lepsze są dla optymalizacji wielowymiarowej - rys.8, podczas gdy błędy średnie i średniokwadratowe dla optymalizacji genetycznej - rys.9). Trzeba jednak podkreślić, iż jeżeli wybierzemy tylko jedną, najlepszą (na podstawie wyników dla zbioru uczącego i testowego) z sieci genetycznych, tj. gen7, to jej wyniki na podstawie testu rozszerzonego przewyższają pod każdym względem wyniki sieci bestopt. Są natomiast porównywalne ze średnimi wynikami sześciu najlepszych sieci optymalizowanych przez ręczny dobór parametrów uczenia. Z kolei najlepsza z nich, sieć mi09et64, uzyskała bezwzględnie najlepsze wyniki, osiągając przy 10%-owej tolerancji błędu poprawność rokowań 93% (błędne rokowania dla 2 z 30 pacjentów.

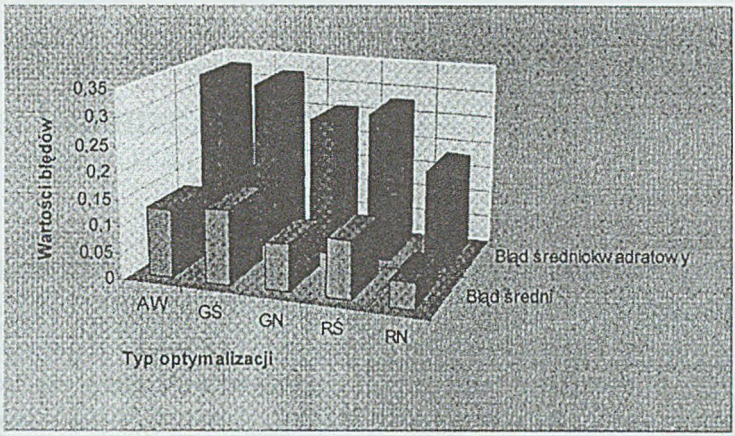


Rys. 8. Współczynniki poprawności rokowań dla różnych typów optymalizacji
Fig. 8. Prognosis correctness coefficients for different optimization types

Z przedstawionych wyników można wnioskować, że optymalizacja genetyczna warta jest polecenia, jako optymalizacja automatyczna, która dała minimalnie lepsze wyniki niż wielowymiarowa, i co najważniejsze, jest stosunkowo niewrażliwa na wybór parametrów genetycznych. Jej wadą natomiast jest niemożność zaimplementowania w prosty sposób mechani-

zmu szukania liczby neuronów ukrytych (w przeciwieństwie do optymalizacji wielowymiarowej). Najlepsze wszakże rezultaty osiągnięto w optymalizacji nieautomatycznej, w której kolejny zestaw parametrów uczenia ustalano na podstawie „algorytmu" modyfikowanego po otrzymaniu wyników z kroków wcześniejszych.

W pracy potwierdzona została ponadto wiarygodność wzoru (17), wśród wielu empirycznych zależności określającej ilość neuronów w warstwie ukrytej.



Rys. 9. Błędy średnie i średniokwadratowe dla różnych typów optymalizacji
Fig. 9. Average and root mean square errors for different optimization types

LITERATURA

[1] Lamb D. J., Niederberger C. S.: New Statistical Techniques That Predict Medical Outcomes. Advances in Urology, 1996, t. 9, s. 407-426.

[2] Moul J.W., Snow P. B., Fernandez E. B., Maher P. D., Sesterhenn I.A.: Neural Network Analisis of Quantitative Histological Factors to Predict Pathological Stage in Clinical Stage I Nonseminomatous Testicular Cancer. Journal of Urology, 1995, t. 153, nr 5, s. 1674-1677.

[3] Niederberger C.S.: Commentary on the Use of Neural Networks in Clinical Urology. The Journal of Urology, 1995, t. 153, nr 5, s. 1362-1362.

[4] Floyd C.E., Lo J.Y., Joon A., Sullivan D. C., Kornguth Ph. J.: Prediction of Breast Cancer Malignancy Using an Artificial Neural Network. Cancer, 1994, t.74, nr 11, s. 2944-2948.

[5] Lawrence J.: Introduction to Neural Networks - Design, Theory, and Applications. California Scientific Software Press, Nevada City 1994.

[6] Tadeusiewicz R.: Sieci neuronowe. Akademicka Oficyna Wydawnicza RM, Warszawa 1993.

[7] Hertz J., Krogh A., Palmer R. G.: Wstęp do obliczeń neuronowych. WNT, Warszawa 1995.

[8] Goldberg D. E.: Algorytmy genetyczne i ich zastosowania. WNT, Warszawa 1995

Recenzent: Dr inż. Sławomir Skoneczny

Wpłynęło do Redakcji 1 września 1997 r.

Abstract

It has been proved so far, that connectionist approach can be very useful for medical diagnosis and prognosis support systems, and that in practical applications, feed-forward architectures (Fig. 1) with neurons which satisfy equation (1) are still leading over the feedback ones. In article, we present the process of designing such network, used to predict the durable recovery for patients with renal cancer. We used set of 113 patients to create: training, testing and extended testing sets. We describe the problem of defining the proper range of input factors changes. As neural network used in prognosis support systems, must be resistant to lack of some factors describing the health state of the patient, and yet, like all neural networks, they must be feed with any value to every input, we present the proper way of representation for lacking inputs. In medical applications, there is also a critical need to ascertain the certainty of neural network output, in order to know when human judgment is required, so our networks present this ability by applying a mechanism of validity checks and consensus of opinion of many networks. As training rule we used back-propagation algorithm, modified by adding smoothing factor (15). We experimented with different types of neuron transfer functions (Fig. 3-7), and the results are shown in table 1. In process of finding the best networks, we also explored two possibilities in choosing the moment of weight modifying (after each fact, and after entire training set). The results are presented in table 2. To find out the optimal architecture in terms of hidden neurons, we applied testing while training, tolerance tuning and hidden neuron pruning when network is trained. Our experiments confirmed the rule (17). Then the learning rate and smoothing factor were changed (table. 3), to get the best network. Finally, multidimensional and genetic optimization, with different evolution parameters values, were applied, and the best networks were tested using the extended testing set. The results are presented in Fig. 8 and Fig. 9. The best prognosis correctness coefficients, as well as root mean square and average errors were for non-automatic optimization, and genetic optimization results were better that results of multidimensional one.