Neuronowe metody odkrywania wiedzy w danych

Krzysztof Grąbczewski, Włodzisław Duch, Rafał Adamczak.
Katedra Metod Komputerowych, Uniwersytet Mikołaja Kopernika,
Grudziądzka 5, 87-100 Toruń,
E-mail: kgrabcze,duch,raad@phys.uni.torun.pl

Streszczenie

Odkrywanie wiedzy w bazach danych jest trudnym i bardzo interesującym zagadnieniem. Omówione zostały formy reprezentacji wiedzy, neuronowe metody ekstrakcji reguł logicznych znane z literatury, rola zmiennych lingwistycznych, a także optymalizacja i rozmywanie reguł logicznych.

1 WPROWADZENIE

Problem odkrywania wiedzy w różnego rodzaju bazach danych, nazywany również dogłębną analizą danych lub drążeniem danych (data mining), staje się aktualnie częstym przedmiotem badań ośrodków naukowych zajmujących się sztuczną inteligencją. Istnieje bardzo wiele różnych sposobów podejścia do tego problemu i wiele różnych systemów, które są używane do takich badań. Oryginalność niektórych podejść sprawia, że bardzo trudno jest ocenić ich możliwości, gdyż nie można porównać otrzymanych za pomocą tych metod wyników z innymi.

Przykładem problemów precyzyjnie zdefiniowanych, które pozwalają porównać możliwości różnych metod są problemy klasyfikacji danych. Należy w nich nie tylko przypisywać nieznanym obiektom klasy z jak największą poprawnością, lecz również w jakiś sposób uzasadnić podejmowaną decyzję. Pozwala to na odkrycie tej części wiedzy ekspertów, która miała największy wpływ na podejmowane decyzje pomimo tego, że często sami eksperci nie są w stanie przedstawić podobnego uzasadnienia swojej diagnozy. Odkrywanie wiedzy w danych dla problemów klasyfikacji może też polegać na wyszukiwaniu tych cech, które najlepiej odróżniają od siebie różne klasy. Na przykład w medycynie bardzo istotnym jest (zarówno z punktu widzenia czasu dochodzenia do właściwej diagnozy jak i koniecznych do poniesienia nakładów finansowych) kierowanie pacjentów na te badania, które potrafią jak najszybciej, najtrafniej, najtaniej i możliwie bezinwazyjnie doprowadzić do właściwej diagnozy.

Poniżej przedstawiono formy reprezentacji wiedzy poszukiwanej w bazach danych i dokonano krótkiego przeglądu neuronowych metod, stosowanych do ekstrakcji reguł logicznych. Następnie omówiono wprowadzone przez nas kontekstowe zmienne lingwistyczne i dwie metody szukania reguł logicznych za pomocą typowych perceptronów wielowarstwowych. Optymalizacja i rozmywanie tak otrzymanych reguł stanowi temat kolejnych rozdziałów. Na zakończenie przedstawiono

przykładowe zastosowanie tych metod i dyskusję perspektyw tego typu podejścia do odkrywania wiedzy w bazach danych.

2 Różne formy reprezentacji wiedzy

Wiedzę odkrytą w danych można przedstawić na wiele różnych sposobów. Od tego, jakiej formy reprezentacji wiedzy potrzebujemy, będzie w dużym stopniu zależało jakiego typu systemów użyjemy. Najbardziej celowym wydaje się szukanie reguł logicznych opisujących dane. Istnieją różne typy reguł logicznych, a najbardziej zrozumiałe dla człowieka wydają się być reguły logiki klasycznej pierwszego rzędu. Reguły rozmyte [Halgamuge i Glesner, 1994, Hayashi, 1991, Kasabov et al, 1998, Nauck et al, 1996, Żurada i Łozowski, 1996] są często zbyt liczne, by mogły być łatwo zrozumiane przez człowieka, a i sama „rozmytość" reguł nie zawsze jest intuicyjnie zrozumiała.

W niektórych przypadkach warto jest użyć reguł w formie **M-of-N** [Towell i Shavlik, 1993] tzn. takich, których przesłanki zawierają zdania typu „*M spośród N podanych warunków jest spełnionych*". Takie reguły potrafią znacznie uprościć zapis, przy czym pozostają dość łatwe w interpretacji.

Oczywiście reprezentacja koncepcji w postaci reguł logicznych jest jedynie najprostszą formą reprezentacji wiedzy. W bardziej złożonych przypadkach dobra klasyfikacja w oparciu o podane cechy może nie być możliwa - należy wówczas poszukać przydatnych cech, zredukować ich liczbę stosując analizę czynników głównych lub niezależnych, lub znaleźć nieliniowe transformacje prowadzące do użytecznych cech. Zakładamy tutaj, że problem opisany jest w wektorowej przestrzeni cech w taki sposób, że poddaje się klasyfikacji za pomocą sieci neuronowych lub innych systemów.

3 Ekstrakcja reguł logicznych

Problem ekstrakcji reguł logicznych z danych lub z sieci neuronowych rozwiązać można na wiele sposobów. Opracowano taksonomię neuronowych algorytmów ekstrakcji reguł [Tickle et al, 1998], która charakteryzuje je ze względu na rodzaj reguł, jakość generowanych reguł (dokładność klasyfikacji, liczbę, zwartość), czytelność reguł, złożoność algorytmu ekstrakcji, czy sposób analizy sieci neuronowej. Do tej taksonomii warto jeszcze dodać sposób użycia zmiennych lingwistycznych, koniecznych do sformułowania reguł.

W najprostszym przypadku sygnały wejściowe i wyjściowe wszystkich neuronów w sieci są binarne. Wówczas po wytrenowaniu sieci można z niej z łatwością „wyczytać" zestaw reguł: wystarczy sprawdzić wyjście dla wszystkich możliwych kombinacji wejść tworząc dla każdej z nich po jednej regule. Przy założeniu, że dana cecha może się pojawiać w regule wprost, pojawiać się zanegowana albo w ogóle nie pojawiać się dla n binarnych cech mamy do sprawdzenia 3^n różnych reguł. Ze wzrostem n to zadanie szybko może się okazać zbyt kosztowne obliczeniowo, więc często szuka się metod ograniczania przestrzeni poszukiwań. Można na przykład ograniczyć liczbę przesłanek, które mogą się pojawiać w regule. Saito i Nakano [Saito i nakano, 1988] ograniczają

głębokość drzewa przeszukiwań pozwalając tylko na takie kombinacje literałów, które występowały w danych treningowych. Wadę takiego rozwiązania, polegającą na akceptowaniu zbyt ogólnych reguł, wyeliminował Gallant [Gallant, 1993], zawężając reguły przez dokładanie cech w nich nie występujących i sprawdzanie ich wszystkich możliwych wartości.

VIA (ang. Validity Interval Analysis) jest systemem opracowanym przez Thurna [Thrun, 1995] operującym na przedziałach walidacyjnych, które przedstawiają zakresy maksymalnych wzbudzeń neuronów. Można ich szukać standardowymi metodami programowania liniowego, można też konstruować sieci neuronowe, w których są one propagowane w przód lub wstecz.

Powyższe metody są przykładami metod globalnych tzn. analizujących jednocześnie wyjścia dla całej sieci i dla wszystkich próbek danych. Metody lokalne analizują fragmenty sieci (często pojedyncze neurony ukryte) w poszukiwaniu reguł opisujących ich zachowanie. Wykorzystywane tutaj sieci używają sigmoidalnych albo zlokalizowanych funkcji transferu. Reguły opisujące działanie całej sieci są tworzone jako stosowne kombinacje reguł odpowiadających poszczególnym węzłom.

Lokalne metody ekstrakcji reguł były przedstawiane m.in. przez Lin Min Fu [Fu, 1991, Fu, 1993, Fu, 1994a, Fu, 1994b] oraz Gallanta [Gallant, 1993]. Podobnie jak w przypadku metod globalnych można tutaj ograniczać głębokość szukania (Sethi i Yoo [Sethi i Yoo, 1994]). Towell i Shavlik w algorytmie *Subset* używają heurystyki polegającej na analizowaniu wag w porządku malejącym, przez co najpierw znajduje się najbardziej ogólne reguły, a potem coraz bardziej szczegółowe. Hayashi [Hayashi, 1991] opracował wersję tej metody generującą reguły rozmyte.

Towell i Shavlik [Towell i Shavlik, 1993] przedstawili algorytm uczący sieć neuronową tak, by łatwo było wygenerować z niej zestaw reguł typu M-of-N. Zbierają oni w grupy połączenia ze zbliżonymi do siebie wagami i zastępują te wagi średnimi wartościami dla całej grupy, eliminując przy tym niepotrzebne wagi. Każda z grup może być opisana przez jedną przesłankę typu M-of-N. Do tego typu metody dodać można stosowną modyfikację wag po porównaniu wektora wag z wektorami wzorcowymi (odpowiadającymi wzorcowym regułom) [McMillan et al, 1992]. Metoda *RuleNet* również wykorzystuje takie wzorce i potrafi wyszukiwać najlepsze reguły typu M-of-N w $O(n^2)$ kroków i najlepsze zbiory zagnieżdżonych reguł w $O(n^3)$ kroków [Alexander i Mozer, 1995]. Ta metoda operuje jednak tylko na danych dyskretnych, a więc cechy o wartościach ciągłych muszą być najpierw zdyskretyzowane.

Rule Extraction As Learning (REAL) jest ogólną techniką stopniowego budowania zestawu reguł zaprezentowaną przez Cravena i Shavlika [Craven i Shavlik, 1994]. Dla nowego przypadku, który jest błędnie klasyfikowany przez dotychczasowy zestaw reguł tworzy się nową regułę i sprawdza wierność powiększonego zestawu z odpowiedziami z sieci neuronowej. Na podobnej zasadzie działa system *RULEENG* [Andrews et al, 1995, Pop et al, 1996].

W metodzie *BRAINNE* [Sestito i Dillon, 1994] sieć o m wejściach i n wyjściach jest przekształcana w sieć o $m+n$ wejściach i n wyjściach i ponownie trenowana. Wejścia dla których wagi nieco się zmieniają po restrukturyzacji sieci są najbardziej istotnymi i są wykorzystywane do budowania reguł.

Podejmowano także próby ekstrakcji reguł logicznych poprzez samoorganizujące się modele typu ART [Healy i Caudell, 1997] i rozmyte ARTMAP [Tan, 1994]. Te ostatnie dają dodatkowo współczynniki pewności dla reguł. Prostsze architektury samoorganizujące się były także uży-

wane dla celów ekstrakcji reguł [Ultsch, 1993], ale dawały raczej mierne wyniki w problemach klasyfikacyjnych.

Algorytm *DEDEC* [Andrews et al, 1995, Tickle et al, 1994] generuje reguły szukając minimalnego zestawu cech wystarczającego z punktu widzenia sieci neuronowej do rozróżnienia zadanego wzorca od innych. Nowy zbiór danych treningowych jest generowany przez zastępowanie oryginalnych przypadków całymi grupami, a wejścia są uporządkowane wg. ich wpływu na klasyfikację. Tylko najważniejsze wejścia biorą udział w tworzeniu reguł, znajdowanych metodami sprawdzania różnych kombinacji wejściowych.

Sieci neuronowe oparte o separowalne zlokalizowane funkcje transferu są równoważne systemom logiki rozmytej [Jang i Sun, 1993] jako, że funkcja transferu każdego węzła może być wprost zapisana w języku logiki rozmytej. Ogólną propozycję systemu neurorozmytego opartego o separowalne funkcje przedstawiono w pracach [Duch, 1994, Duch et al, 1995]. Ogólną dyskusję na temat ekstrakcji reguł przy użyciu zlokalizowanych funkcji transferu przeprowadzili w swojej pracy [Andrews i Geva, 1996] Andrews i Geva. Takie systemy neurorozmyte powinny mieć zdecydowaną przewagę w zastosowaniach do ekstrakcji reguł, ponieważ reguły logiki klasycznej są podzbiorem reguł rozmytych. Znanych jest wiele takich metod [Duch et al, 1995, Nauck i Kruse, 1996, Nauck et al, 1996, Halgamuge i Glesner, 1994, Żurada i Łozowski, 1996] i teoretycznie powinny one dać bardzo dobre wyniki. W praktyce rzadko wykorzystuje się je do ekstrakcji klasycznych reguł. Główną przyczyną są trudności ze znalezieniem optymalnego rozwiązania dla licznych parametrów adaptacyjnych [Kasabov, 1996, Kasabov et al, 1998]. Funkcja błędu dla klasycznych reguł logicznych wydaje się mieć wiele minimów lokalnych, w których metody gradientowe łatwo grzęzną.

Systemy wykorzystujące teorię zbiorów przybliżonych [Pawlak, 1998] prowadzą z natury do zbioru reguł, jednak potrzebują one dodatkowych procedur dyskretyzacyjnych dla ciągłych atrybutów i zwykle dają bardzo dużą liczbę reguł.

Zamiast szukać bezpośrednio logicznego opisu danych można próbować opisać za pomocą reguł logicznych działanie dowolnego klasyfikatora. Jeśli udało się nam stworzyć dobry klasyfikator (np. sieć neuronową) można go wykorzystać do odpowiedzi na wiele pytań (nazywa się go często „wyrocznią"). Jednym z bardzo efektywnych systemów działających w taki sposób jest *TREPAN* stworzony przez Cravena i Shavlika [Craven, 1996], który generuje drzewo decyzji w oparciu o analizę odpowiedzi sieci neuronowej dla przedstawionych jej próbek danych. Podstawową zaletą takiego rozwiązania jest uniezależnienie systemu ekstrakcji reguł od zbioru danych. Nowe próbki mogą być generowane tak, by zapewnić wystarczającą liczbę przypadków w tych obszarach, o których same dane dostarczają niewiele informacji. Można w ten sposób poszukiwać logicznego opisu działania każdego systemu klasyfikującego (np. korzystającego z metod opartych na podobieństwie, metod statystycznych itp.). Istotną wadą takiego podejścia jest fakt, że tak powstałe reguły mogą znacznie odbiegać od wyjściowych danych ponieważ nakładają się tutaj dwa różne błędy uczenia - pierwszy podczas uczenia badanego klasyfikatora, a drugi podczas próby opisu jego działania. Z tego powodu bardziej uzasadnionym podejściem do celu ekstrakcji reguł logicznych z surowych danych wydaje się być modyfikowanie algorytmów uczenia systemów sztucznej inteligencji w taki sposób, by bezpośrednio po nauczeniu systemu móc z łatwością opisać jego działanie przez zbiór reguł logicznych. Stosunkowo łatwo jest opisać działanie sieci neuronowej

typu MLP (ang. Multilayer Perceptron - wielowarstwowy perceptron) regułami rozmytymi. Jednak zwykle liczba reguł jest duża a ich zrozumienie niemożliwe, co sprawia, że mimo opisu regułowego wciąż nie mamy wiedzy w postaci zrozumiałej dla człowieka. Zdecydowanie bardziej przydatne mogą być klasyczne reguły, które dla odróżnienia od rozmytych będziemy tutaj nazywali ostrymi.

Rodzina systemów, które polegają na modyfikacji algorytmów uczenia tak, by ułatwić ekstrakcję reguł jest również dość liczna. Setiono i Liu [Setiono i Liu, 1995] używają członu regularyzacyjnego w funkcji kosztów dla eliminowania małych wag. Podobna idea przyświeca metodzie *Successive Regularization* opracowanej przez Ishikawę [Ishikawa, 1996], gdzie kładzie się nacisk na to, by neurony ukryte były w pełni wzbudzone albo całkowicie nieaktywne, przy czym stosowny człon regularyzacyjny dba o eliminację wag mniejszych od pewnego progu (metodę nazwano mianem „selektywnego zapominania" *selective forgetting*). Inną metodę należącą do tej grupy zastosowali Geczy i Usui [Geczy i Usui, 1997]: wagi sieci typu MLP są tutaj po zakończeniu procesu uczenia przekształcane w 0, $+1$ lub -1, co znacznie ułatwia szukanie reguł. Andrews i Geva stworzyli metodę *RULEX* [Andrews i Geva, 1994] wykorzystującą sieci neuronowe typu MLP z liniowymi kombinacjami par funkcji sigmoidalnych o niezerowych wartościach w przedziałach, z których potem można wprost wyczytać reguły.

Większość z opisanych powyżej systemów nie ma możliwości kontrolowania zbioru reguł pod względem dokładności i czytelności. Użytecznym byłoby dysponować kilkoma zbiorami reguł: od najprostszych w formie i najbardziej ogólnych do bardziej szczegółowych i dokładnych. Dość istotną sprawą (zwłaszcza w zastosowaniach medycznych) jest również kwestia wiarygodności reguł, która zwykle może być osiągnięta kosztem ich dokładności.

Kilka metod odkrywania ostrych i rozmytych reguł logicznych opisujących dane oraz wyboru najistotniejszych (najbardziej informatywnych) cech opisujących dane powstało w naszym zespole. Niektóre z nich oparte są na sieciach neuronowych. Dwa z naszych algorytmów są modyfikacjami metody wstecznej propagacji błędu, pozwalając na przekształcenie wielowarstwowych perceptronów (MLP) w sieci, których działanie można łatwo zinterpretować w postaci formuł logicznych (LN, logical networks). Stąd pochodzą nazwy tych metod, **MLP2LN** i w wersji konstruktywistycznej **C-MLP2LN**. Innym rodzajem sieci neuronowej jest model **FSM** (ang. Feature Space Mapping). Jest to system neurorozmyty, który potrafi generować zarówno reguły rozmyte jak i ostre (por. opis w tym tomie i [Duch et al, 1995]). Interesujące wyniki można uzyskać także stosując powszechnie znane metody szukania w celu znalezienia optymalnej sieci MLP (w odpowiednio ograniczonym obszarze).

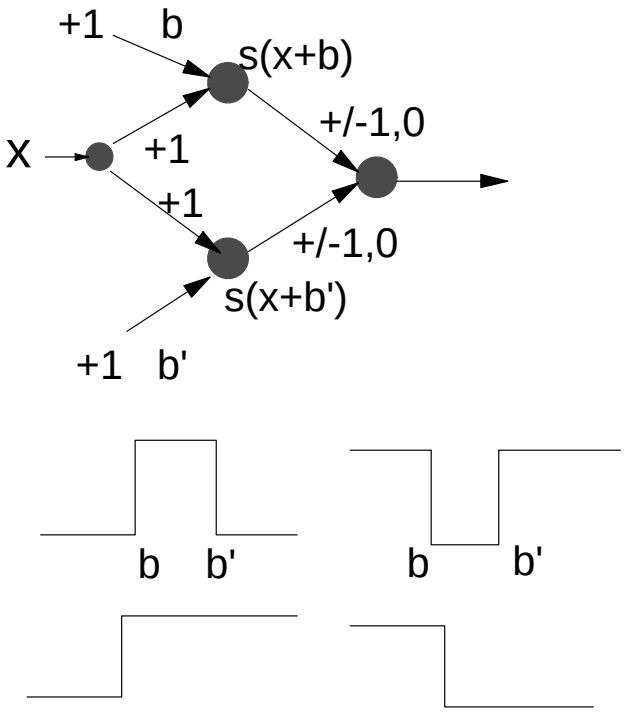
3.1 MLP2LN i C-MLP2LN

Aby sieć typu MLP mogła zostać przekonwertowana do postaci reguł logiki klasycznej, wykorzystywane funkcje aktywacji muszą powodować maksymalne wzbudzanie neuronów albo całkowity brak wzbudzenia. Dlatego neurony, z których zbudowane są sieci MLP2LN i C-MLP2LN realizują funkcje sigmoidalne wzbogacone o parametr s (zwany skosem), którym można zmieniać stromość funkcji tak by w granicy przy s zmierzającym do nieskończoności funkcje te były równoważne funkcjom progowym:

$$f(x) = \frac{1}{1+e^{-s(Wx+b)}} \tag{1}$$

Zatem dla dużych wartości skosu funkcje te dają się wprost przekładać na język logiki, a mniejsze skosy pozwalają efektywnie wykorzystywać metody gradientowe do trenowania sieci.

Kontekstowe zmienne lingwistyczne Lingwistyczne jednostki neuronów (nazywane jednostkami L) automatycznie analizują wejścia i produkują zmienne lingwistyczne [Duch et al, 1999a]. Pomysł oparty jest na „funkcjach okienkowych", które można uzyskać z kombinacji dwóch neuronów z funkcjami sigmoidalnymi o różnych wartościach progów b i b'. Różnice dwóch sigmoid reprezentują typową zmienną lingwistyczną równoważną warunkowi $x \in [b, b']$ lub jego zaprzeczeniu. Pojedyncze sigmoidy realizują przedziały jednostronnie nieskończone. Wartości progów są parametrami sieci, które podlegają procesowi adaptacji. Wszystkie sigmoidy w końcowym etapie uczenia stają się bardzo strome, dzięki czemu wiernie reprezentują przedziały.



Rysunek 1: Schemat jednostki L.

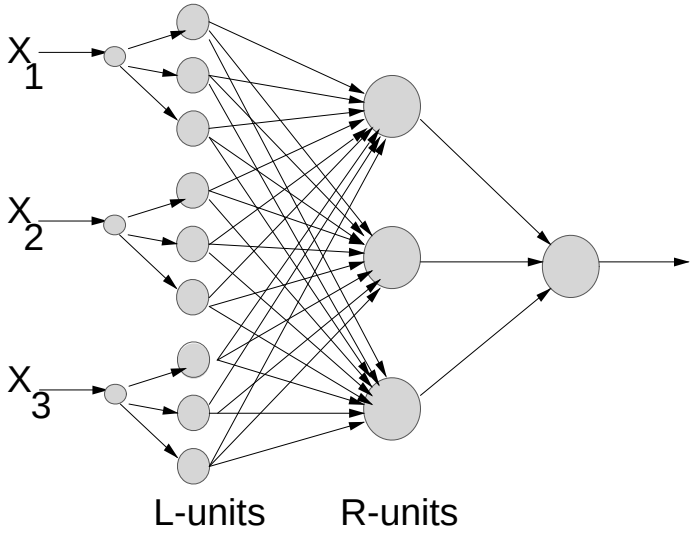
Schemat jednostki L jest pokazany na rysunku 1. Składa się ona z neuronu wejściowego, który jest połączony wagami ustawionymi na 1 i nie podlegającymi uczeniu z dwoma neuronami wewnętrznymi, które z kolei połączone są z pojedynczym neuronem wyjściowym jednostki. Wagi dochodzące do neuronu wyjściowego mogą przybierać wartości 0, $+1$ lub -1, co daje możliwość realizacji przedziałów skończonych jak i lewostronnie i prawostronnie nieskończonych.

Można oczywiście zamiast jednostek L użyć metod dyskretyzacji danych tak, by sieć nie dostawała na wejściu sygnałów ciągłych, ale rozbudowanie sieci o jednostki L daje możliwość realizacji celu szukania zmiennych lingwistycznych w kontekście powstających reguł, a co za tym

idzie większe szanse na zwarte i skuteczne reguły. Dodatkowo użycie jednostek *L* sprowadza etapy wyboru zmiennych lingwistycznych i tworzenia reguł do jednego zadania.

Kiedy używamy zdyskretyzowanych danych pojedyncze wejście dla cechy o ciągłych wartościach musi zostać zamienione na wektor elementów wejściowych, składających się z wartości ± 1. Wektor ten ma wymiar równy liczbie możliwych (dyskretnych) wartości przyjmowanych przez daną cechę wejściową. Na przykład cecha, która może przyjmować trzy wartości lingwistyczne: *mały*, *średni* i *duży*, musi zostać wprowadzona do sieci jako trzy niezależne wejścia odpowiadające tym wartościom lingwistycznym. Jeśli cecha przyjmuje wartość *mały*, to na pierwszym z tych trzech wejść postawimy wartość $+1$ a na pozostałych -1, co pozwoli łatwo odnajdywać także reguły z negacjami. A zatem nasze trzy wartości lingwistyczne zostaną zakodowane następująco jako wektory wejściowe: mały=$[+1,-1,-1]$, średni=$[-1,+1,-1]$ oraz duży=$[-1,-1,+1]$.

Struktura sieci. Sieć MLP2LN składa się z trzech warstw: wejściowej, ukrytej i wyjściowej (kiedy używamy jednostek *L* mamy w rzeczywistości większą liczbę warstw, ale dla uproszczenia opisu traktujemy je jako część warstwy wejściowej). Liczba węzłów w warstwie wyjściowej równa jest liczbie klas w zbiorze treningowym, natomiast w warstwie wejściowej liczbie zmiennych lingwistycznych. Każdy z neuronów w warstwie ukrytej jest połączony ze wszystkimi węzłami z warstwy wejściowej i z jednym neuronem wyjściowym (będzie realizował reguły klasyfikujące do klasy odpowiadającej temu wyjściu). Proces uczenia odbywa się dla każdego wyjścia (klasy) niezależnie. Można więc powiedzieć, że tworzymy dla każdej klasy osobną sieć. Schemat takiej sieci przedstawia rysunek 2. Początkowa liczba węzłów w warstwie ukrytej jest zależna od tego



Rysunek 2: Struktura sieci MLP2LN.

czy stosujemy standardową wersję algorytmu czy konstruktywistyczną. Podczas procesu uczenia wymuszane są wagi zerowe, $+1$ lub -1. Analizując wagi i próg dla neuronu z warstwy ukrytej otrzymujemy reguły odnoszące się do klasy, z którą ten neuron jest połączony (połączenie z

określonym węzłem wyjściowym). Jeśli waga połączenia jest równa $+1$ to otrzymujemy reguły dla danej klasy, jeśli natomiast waga jest równa -1 to wyjątki, czyli reguły opisujące przypadki błędnie klasyfikowane przez istniejące ogólne reguły dla tej klasy. Węzły w warstwie wyjściowej dokonują jedynie sumowania aktywacji odpowiednich węzłów z warstwy ukrytej, więc w związku z tym, że na wyjściu oczekujemy wartości 0 lub 1, to sytuacja, gdy dwa węzły klasyfikują ten sam wektor traktowana jest jako błąd. Dzięki temu otrzymujemy z różnych węzłów reguły, które są rozłączne, czyli nie klasyfikują tych samych wektorów.

Algorytm uczenia. Logiczna interpretacja węzłów w sieci MLP jest w ogólności trudna, dlatego algorytm MLP2LN używa funkcji sigmoidalnych o stopniowo (w czasie uczenia) wzrastającym nachyleniu. W czasie uczenia wymuszane są wartości wag równe $0,+1,-1$. Wartość 0 oznacza że zmienna wejściowa połączona tą wagą jest nieistotna, $+1$ oznacza, że dana wartość cechy musi wystąpić oraz -1, że nie może wystąpić. Można to osiągnąć poprzez modyfikacje funkcji błędu stosowanej dla algorytmu wstecznej propagacji:

$$E(W) = \frac{1}{2}\sum_p \sum_k \left(Y_k^{(p)} - \mathbf{A}_W\left(X^{(p)}\right)_k\right)^2 + \frac{\lambda_1}{2}\sum_{i>j} W_{ij}^2 + \frac{\lambda_2}{2}\sum_{i>j} W_{ij}^2 (W_{ij}-1)^2 (W_{ij}+1)^2 \tag{2}$$

Można zastosować również człony kary w innej postaci [Duch et al, w druku], np. prostszym członem trzeciego stopnia:

$$|W_{ij}||W_{ij}^2 - 1| \tag{3}$$

Z dotychczasowego doświadczenia trudno jest stwierdzić różnice w zastosowaniu między tymi dwoma członami. Nową postać przyjmuje również gradient, mamy więc dodatkowe człony we wzorze na zmianę parametrów sieci. Dla członu szóstego stopnia jest to:

$$\lambda_1 W_{ij} + \lambda_2 W_{ij}(W_{ij}^2 - 1)(3W_{ij}^2 - 1) \tag{4}$$

Pierwszy dodatkowy człon wymusza podczas uczenia małe wartości wag przez co prowadzi do eliminacji cech zbędnych, natomiast człon drugi wymusza dla wag wartości $-1,0,+1$ umożliwiając późniejszą logiczną interpretację sieci. Za pomocą parametrów λ_1, λ_2 możemy zwiększać lub też zmniejszać dominację odpowiednich członów. Ustalenie dominacji któregoś z członów wyznacza granicę miedzy prostotą a dokładnością reguł otrzymanych z sieci. Jeżeli chcemy uzyskać bardzo prostą sieć, a co za tym idzie proste reguły dające przybliżony opis danych, to pierwszy człon powinien być tak duży, jak to tylko jest możliwe, przy akceptowalnym jeszcze błędzie. Na początku procesu uczenia parametr λ_2 jest równy zero natomiast λ_1 jest mały (ma wartość 0.00001). Z takimi parametrami uczymy sieć tak długo, jak długo maleje błąd. Następnie zwiększamy wartość parametru λ_1 (np. do 0.10) i ponownie uczymy. Z reguły po zwiększeniu parametru λ_1 następuje wzrost błędu SSE – można go nieco zmniejszyć przez zwiększenie skosów. Tę procedurę uczenia powtarzamy tak długo, aż zaobserwujemy, że większość wag ma wartość zero lub też nastąpił

bardzo duży skok błędu. W tym momencie usuwamy zbędne połączenia, człon odpowiedzialny za wymuszanie małych wartości wag przestaje być ważny i uaktywniamy człon drugi. Wartość parametru λ_2 jest równa lub też trochę większa od ostatniej wartości parametru λ_1. Jednocześnie nadal zwiększamy nachylenie funkcji sigmoidalnych, realizowanych przez węzły sieci. W celu dalszego zmniejszenia wag można jednocześnie z niezerowym parametrem λ_2 utrzymywać niezerową wartość parametru λ_1. Wówczas wartość parametru λ_1, początkowo istotnie większa od λ_2, stopniowo maleje w miarę zwiększania λ_2 i w końcu zmierza do zera.

W przypadku trudniejszych danych warto spróbować kilku strategii zmiany parametrów po to, by uzyskać najprostsze reguły. Kontynuujemy proces uczenia zwiększając wartość parametru λ_2 oraz nachylenia sigmoid. Wraz ze wzrostem wartości parametru λ_2 wagi stają się coraz bliższe docelowych wartości. Parametr λ_2 nie powinien przekraczać wartości 1. Jeśli parametr ten osiągnął już swoją maksymalną wartość a parametr uczenia, powoli zmniejszany w procesie uczenia, nie jest jeszcze bardzo mały, tzn. > 0.00001, to nie zmieniamy już wartości λ_2, a jedynie zwiększamy skos i zmniejszamy parametr uczenia, aż osiągnie wartość minimalną (np. 0).

W końcowym etapie zwiększamy nachylenie sigmoid do bardzo dużych wartości (1000) przez co uzyskujemy ostre granice decyzyjne. Początkowe wymuszenie małych wartości wag umożliwia w późniejszym etapie wyzerowanie tych wag zupełnie, pozostałe wagi natomiast będą zbliżały swoją wartość do ± 1 dzięki drugiemu członowi. Może się zdarzyć, że na skutek zbyt mocnego wymuszenia wag o małych wartościach w pierwszej fazie, w końcowym etapie uzyskuje się węzeł, który posiada wszystkie wagi zerowe. W takim przypadku trzeba ponownie węzeł zainicjować i powtórzyć proces uczenia utrzymując mniejszą wartość parametru λ_1. Pomimo tego, że z nauczonego węzła ukrytego otrzymujemy proste reguły, dobrze jest spróbować ponownie nauczyć sieć, ale z jeszcze większym wymuszeniem początkowych zer. Często zdarza się, że taka procedura prowadzi do jeszcze prostszej postaci sieci. Liczba reguł, która zostanie utworzona z danego węzła bardzo mocno zależy od liczby niezerowych wag, dlatego też etap pierwszy (wymuszanie małych wag) jest bardzo istotny. Cała procedura wymuszania wartości na wagach w sieci dotyczy tylko i wyłącznie wag, wszystkie progi w sieci mogą przyjmować dowolne wartości.

Mimo, że dodatkowe człony w funkcji błędu nie zmieniają MLP dokładnie w sieć logiczną, to ułatwiają w znaczny sposób logiczną interpretację końcowej sieci.

Interpretacja węzłów ukrytych. Wszystkie sygnały wejściowe oraz wagi pomiędzy warstwą wejściową a wyjściową mają wartości $+1,-1,0$, dlatego też sygnał wpływający do węzła ukrytego ma wartości całkowite. Ponieważ sigmoidy w węzłach mają bardzo duży skos (są bardzo ostre, końcowy skos jest równy 1000) to funkcja aktywacji ma wartość $+1,0$. Na podstawie analizy aktywacji oraz progu sigmoidy możemy określić w jakich przypadkach węzeł może się wzbudzić. Wzbudzenie może nastąpić tylko wtedy, gdy wartość aktywacji przekroczy wartość progu. Ponieważ funkcja aktywacji ma postać 1, to $f(x) = 1$ gdy $e^{-s(Wx+b)} = 0$, a więc gdy $Wx + b > 0$ przy założeniu, że skos jest bardzo duży. Żeby utworzyć reguły wystarczy analizować przypadki w których $Wx > -b$. Rozpatrzmy następujący przykład (dla prostoty analizujemy węzeł ukryty, który połączony jest tylko z jednym wejściem lingwistycznym): $W = [+1,0,0,-1]$, $b = -2$, sygnał wejściowy $x \in [x_1,x_2,x_3,x_4]$ gdzie $x_1 = [+1,-1,-1,-1]$, $x_2 = [-1,+1,-1,-1]$, $x_3 = [-1,-1,+1,-1]$, $x_4 = [-1,-1,-1,+1]$, są interpretowane jako x_1 =mały, x_2 =średni, x_3 =duży

i x_4=bardzo duży. Mamy zatem: $Wx_1 = 2, Wx_2 = 0, Wx_3 = 0\ Wx_4 = -2$, czyli tylko w pierwszym przypadku $Wx > -b$ więc reguła, która opisuje działanie takiego neuronu ukrytego ma postać:

$$\text{IF } x = \text{mały THEN Klasa 1}$$

Gdybyśmy natomiast mieli $b = 0$, to otrzymalibyśmy regułę

$$\text{IF } x = \text{mały} \vee x = \text{średni} \vee x = \text{duży THEN Klasa 1}$$

co można zapisać w prostszej formie jako:

$$\text{IF } \neg\, x = \text{bardzo duży THEN Klasa 1}$$

C-MLP2LN. Ponieważ liczebność zbioru reguł, które powstaną z sieci jest w dużej mierze zależna od liczby węzłów ukrytych, to problem ustalania tej liczby (poważny dla sieci typu MLP) nabiera tutaj jeszcze większego znaczenia. Problem ten można skutecznie rozwiązać stosując konstruktywistyczną wersję metody MLP (stąd nazwa C-MLP2LN). Na początku w sieci dla danej klasy istnieje tylko jeden neuron ukryty, który trenowany jest na wszystkich wektorach z ciągu treningowego. Do zmiany wag stosujemy standardową procedurę wstecznej propagacji z momentem. Po zakończeniu uczenia dostawiany jest nowy neuron do warstwy ukrytej, połączony z tą samą klasą. Poprzedni neuron jest natomiast zamrażany, tzn. wagi tego neuronu podczas dalszego uczenia nie będą się zmieniały. Dzięki temu wektory wejściowe, które są poprawnie klasyfikowane przez neuron zamrożony, nie dają już wkładu do funkcji błędu. Uczymy sieć ponownie i w razie konieczności dołączamy następny neuron.

Jeśli zamrożone neurony popełniają błędy dołączamy neuron z wagą -1 połączenia z jednostką wyjściową. Oznacza to, że szukamy wyjątków od działania już zbudowanego fragmentu sieci, czyli staramy się znaleźć regułę, która opisze wektory dotychczas błędnie klasyfikowane. Całą procedurę powtarzamy tak długo, aż uzyskamy wystarczająco mały błąd, albo aż reguły, które powstają podczas analizy ostatnio nauczonego węzła staną się zbyt szczegółowe, lub jest ich zbyt dużo. Ponieważ pierwsze węzły w sieci obejmują cały zbiór treningowy to reguły, które się z nich otrzymuje są najbardziej ogólne. Kolejne neurony dają coraz bardziej szczegółowe reguły, aż wreszcie otrzymuje się reguły opisujące pojedyncze wektory. Takie reguły, opisujące niewielką liczbę wektorów treningowych, powinny być odrzucane, ponieważ psują generalizację. A zatem reguły generowane są w porządku od najbardziej ogólnych do coraz bardziej szczegółowych. Proces uczenia jest bardzo szybki, ponieważ w danej chwili uczony jest tylko jeden węzeł sieci.

3.2 Szukanie optymalnego MLP.

Metody minimalizacji i metody szukania mają wspólny cel polegający na znalezieniu minimalnej wartości funkcji kosztów. Dlatego też można zastąpić metody gradientowe odpowiednimi metodami szukania. W praktyce, aby takie przeszukiwanie trwało sensownie krótko, musi ono ograniczać się do stosunkowo małego podzbioru przestrzeni wszystkich możliwych rozwiązań. W przypadku szukania parametrów sieci MLP (zwłaszcza, kiedy jesteśmy zainteresowani uzyskaniem sieci,

którą będzie można zinterpretować logicznie) można znacznie przyspieszyć szukanie ograniczając wartości wag do zbioru liczb całkowitych. W takim przypadku już proste wyszukiwanie algorytmem „najpierw najlepszy” (ang. best first search) jest w stanie szybko znaleźć bardzo dobre rozwiązanie.

Algorytm, który stosowaliśmy [Duch i Grąbczewski, 1999] rozpoczyna działanie przez zainicjowanie wartości wszystkich wag $W_{ij} = 0$ i progów $\theta_i = -0.5$. Takie ustawienia powodują, że neuron wyjściowy nie wzbudzi się dla żadnej kombinacji wejść. Następnie należy ustalić wartość kroku Δ, o który będą się zmieniały wagi i progi. W każdej iteracji procesu szukania rozpatruje się wszystkie możliwe zmiany wag i progów ($W_{ij} \pm \Delta,\ \theta_i \pm \Delta$) i ocenia wpływ tych zmian na błąd klasyfikacji. Można tu zastosować wiele różnych algorytmów np. „najpierw najlepszy” albo „przeszukiwanie wiązką” (ang. beam search) [Kanal i Kumar, 1988] dla pojedynczych zmian wag w każdym kroku. Nie zawsze jednak tak proste metody szukania są w stanie znaleźć satysfakcjonujące minimum lokalne funkcji błędu. Dlatego też można zastosować bardziej złożone obliczeniowo metody (np. zmieniając w pojedynczym kroku dowolną parę parametrów). Aby przyspieszyć działanie metody i jednocześnie uzyskiwać dobre wyniki stosowaliśmy szukanie dwuetapowe: w pierwszym etapie zmienialiśmy wagi i progi pojedynczo i zaznaczaliśmy te z nich, które dają najlepszą zmianę błędu klasyfikacji, a w drugim rozpatrywaliśmy wszystkie możliwe pary (a nawet podzbiory) złożone z parametrów zaznaczonych w pierwszym etapie.

Nakładając na metodę dodatkowe ograniczenia możemy tworzyć sieci, które z łatwością będzie można opisać ostrymi albo rozmytymi regułami logicznymi. Jeśli na przykład wszystkie wagi w sieci będą liczbami całkowitymi (co otrzymamy stosując $\Delta = 1$) i funkcje realizowane przez neurony ukryte są wystarczająco stromymi sigmoidami, to stworzona sieć może być opisana przez zbiór reguł typu *M*-of-*N*. Reguły tworzone są w wyniku analizy wszystkich możliwych kombinacji sygnałów wejściowych (tak samo jak w przypadku metody MLP2LN). Aby zagwarantować sobie małą liczbę reguł logicznych można dodatkowo ograniczyć przeszukiwaną przestrzeń przez wyłączenie ze zmian wartości progów i automatyczne ustawianie każdego z nich po każdej zmianie wag tak, by był równy sumie wszystkich wag połączeń dochodzących do jego neuronu pomniejszonej o 0.5 ($\theta_i = \sum_j W_{ij} - 0.5$). W takim przypadku każdy z neuronów ukrytych będzie mógł być opisany pojedynczą regułą jako, że tylko jedna kombinacja wejść da w sumie wartość przewyższającą wartość progu.

W tej metodzie można też zastosować dodatkową technikę ułatwiającą dochodzenie do optymalnych rozwiązań przez stopniowe zwiększanie rozdzielczości, w której „oglądamy” przestrzeń (tzn. startujemy ze stosunkowo dużą wartością Δ i w trakcie procesu szukania stopniowo ją zmniejszamy). Taki sposób szukania optymalnej sieci można porównać do stopniowego zmniejszania parametru uczenia w metodzie propagacji wstecznej błędu, czy też do technik „stopniowego schładzania”.

4 Optymalizacja reguł

Ponieważ metody gradientowe odnajdują zwykle lokalne minima funkcji błędu, zbiór reguł „wyczytany” ze struktury sieci neuronowej nie musi być optymalnym opisem danych treningowych. Jednak mając tak stworzony opis regułowy można próbować go udoskonalać modyfikując grani-

ce przedziałów pojawiających się w regułach przy użyciu globalnych metod optymalizacji. Takie modyfikacje można przeprowadzać na wiele różnych sposobów [Duch et al, 1999b], np. można maksymalizować ślad macierzy rozrzutu $P(C_i,C_j|M)$ aby uzyskać maksymalną poprawność klasyfikacji. Można też minimalizować liczbę pomyłek klasyfikacji kosztem częściej udzielanych przez reguły odpowiedzi „nie wiem", aby w ten sposób zwiększać wiarygodność klasyfikacji dla tych przypadków, które spełniają przesłanki otrzymanych reguł. Dobre wyniki daje kombinacja tych dwóch metod, czyli optymalizacja funkcji błędu dla klasyfikatora regułowego:

$$E(M) = \gamma \sum_{i \neq j} P(C_i,C_j|M) - \mathrm{Tr}\, P(C_i,C_j|M) \geq -n \tag{5}$$

gdzie n to liczba klasyfikowanych wektorów, M to parametry modelu (dla reguł przedziały zmiennych lingwistycznych), zaś γ określa balans pomiędzy poziomem zaufania do reguł a liczbą wektorów odrzucanych jako nieznane. Można też stworzyć hierarchiczny system reguł tak, aby przy użyciu pewnych zestawów reguł dawać bardzo wiarygodne odpowiedzi, a przy użyciu innych mniej wiarygodne, ale obejmujące coraz większe części przestrzeni danych [Duch et al, 1999b, Duch et al, w druku].

5 ROZMYWANIE REGUŁ

Stosowanie hierarchicznych systemów reguł daje możliwość oceny prawdopodobieństwa poprawności klasyfikacji. Czasami jednak nie potrzebujemy wielu zestawów reguł, aby móc przypisać klasyfikacji regułowej pewien współczynnik zaufania. Najprostszą metodą jest uznanie niedokładności danych i uwzględnienie rozkładu błędów przy obliczaniu prawdopodobieństwa przynależności danego przypadku do poszczególnych reguł klasyfikacji i do poszczególnych klas. Zastępując wartość x danej cechy gaussowskim rozkładem $G(y;x,s_x)$, można obliczyć prawdopodobieństwo przynależności wartości tej cechy do przedziału występującego w regule:

$$P(x \in (a,b)) = \frac{1}{2}\left[\mathrm{erf}\left(\frac{b-x}{s_x\sqrt{2}}\right) - \mathrm{erf}\left(\frac{a-x}{s_x\sqrt{2}}\right)\right] \tag{6}$$

Jest ono dane przez funkcję błędu, która bardzo przypomina funkcje logistyczne, używane w sieciach neuronowych. Prawdopodobieństwo spełniania reguły przez dany wektor można wyliczyć jako iloczyn prawdopodobieństw przynależności wartości poszczególnych cech do przedziałów zadanych regułą. W efekcie połączenie rozmytych gaussowsko danych z regułami logiki klasycznej daje ten sam efekt, co użycie ostro określonych danych z rozmytymi regułami, których funkcje przynależności mają kształt okienek zdefiniowanych jako różnica albo iloczyn stosownych sigmoid.

Uznanie danych za rozmyte i stosowanie prawdopodobieństw zamiast binarnych decyzji może samo w sobie przynieść poprawę klasyfikacji, daje więcej informacji w przypadku trudnych do sklasyfikowania przypadków, a także daje możliwość zastosowania metod gradientowych do optymalizacji zbiorów reguł [Duch et al, w druku]. Parametry rozmyć s_x można dla niektórych danych ocenić na podstawie dokładności pomiarów lub uznać za parametry adaptacyjne minimalizacji funkcji 5.

6 WYNIKI

Nasze neuronowe metody ekstrakcji reguł logicznych zastosowaliśmy do wielu zbiorów danych, między innymi do powszechnie znanych zbiorów zawartych w bazie gromadzonej w UCI (University of California at Irvine). Otrzymane wyniki pozwalają wysoko ocenić efektywność tych metod. Najczęściej wyniki plasują się w ścisłej czołówce rankingu poprawności klasyfikacji, a logiczny opis struktury danych daje możliwość zrozumienia decyzji podejmowanych przez system i w ten sposób wydobycia z danych wiedzy, która może się okazać przydatna ekspertom w danej dziedzinie. Szczególnie przydatne mogą być regułowe opisy diagnoz medycznych.

Przedstawimy tutaj kilka przykładów dla zilustrowania opisywanych metod i porównania wyników z osiąganymi przez inne systemy. Zestawy reguł dla większej liczby baz danych znaleźć można w pracach [Duch et al, 1999b, Duch et al, 1999a, Duch et al, w druku] oraz pod internetowym adresem *http://www.phys.uni.torun.pl/kmk/projects/rules.html*.

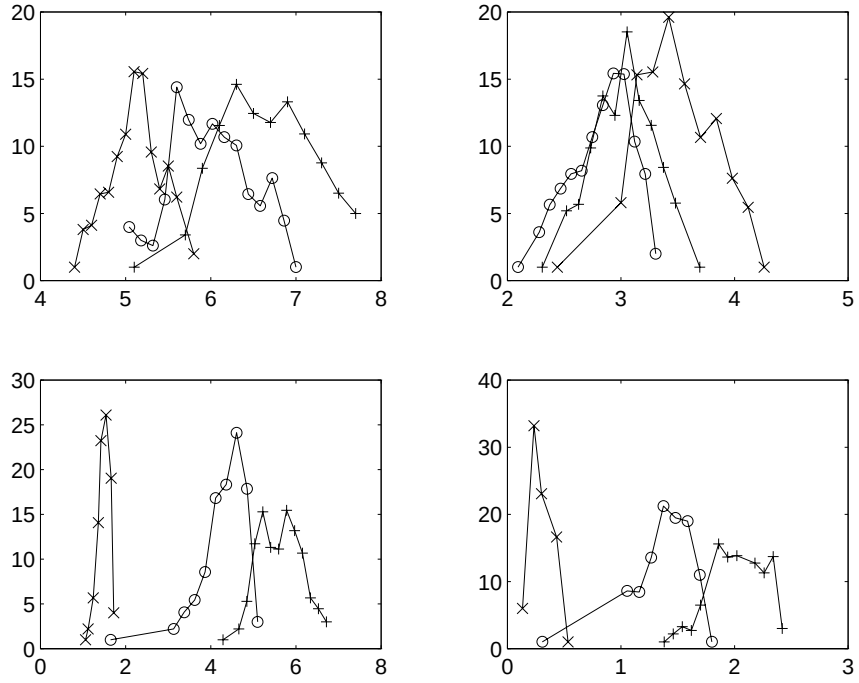
Iris - ilustracja opisywanych metod. Dobrym przykładem ilustrującym metodologię jest przypadek danych o irysach. Baza zawiera opisy 150 kwiatów klasyfikowanych do trzech grup: iris-setosa, iris-versicolor i iris-virginica. Każdy kwiat opisany jest czterema liczbami: długością x_1 i szerokością x_2 listka oraz długością x_3 i szerokością x_4 płatka.

Ponieważ wszystkie cztery cechy opisujące dane mają charakter ciągły, to musimy albo dokonać ich dyskretyzacji przed użyciem metod MLP2LN albo też zastosować wariant sieci z jednostkami L. Najprostszą metodą uzyskania zmiennych lingwistycznych jest arbitralny podział zakresów cech na kilka równych części. Na przykład dzieląc na trzy części możemy uzyskać przedziały reprezentujące zmienne lingwistyczne o wartościach mały (s), średni (m) i duży (l). Oczywiście ostateczne wyniki będą mocno uzależnione od wybranego na początku podziału, więc stosując taką metodę dyskretyzacji mamy małe szanse uzyskać dobre reguły. W przypadku irysów podział na trzy równe części daje całkiem dobre wyniki (bo odpowiada rzeczywistemu rozkładowi danych), ale już podział na cztery czy pięć części prowadzi do większej liczby reguł i mniejszej poprawności klasyfikacji.

Jeśli jednak chcemy dokonać dyskretyzacji ręcznie, to warto do tego celu posłużyć się histogramami, które pokazują rozkład klas w poszczególnych wymiarach. Analiza histogramów dla danych o irysach (rysunek 3) potwierdza, że podział na trzy równe części dość dobrze odzwierciedla rozkład klas. Dzieląc zakresy cech na podstawie sporządzonych histogramów otrzymujemy zmienne lingwistyczne przedstawione w tabeli 1. Dla takiej dyskretyzacji metoda C-MLP2LN

Tabela 1: Zmienne lingwistyczne powstałe w wyniku analizy histogramów.

	s	*m*	*l*
x_1	[4.3,5.5]	(5.5,6.1]	(6.1,7.9]
x_2	[2.0,2,75]	(2.75,3.2]	(3.2,4.4]
x_3	[1.0,2.0]	(2.0,4.93]	(4.93,6.9]
x_4	[0.1,0.6]	(0.6,1.7]	(1.7,2.5]



Rysunek 3: Histogramy dla czterech cech opisujących irysy. Cechy x_3 i x_4 (dolne wykresy) pozwalają lepiej oddzielić różne klasy, niż pierwsze dwie.

tworzy po jednym neuronie dla każdej z klas. Struktura nauczonej sieci przedstawiona jest na rysunku 4. Ponieważ mamy po jednym neuronie ukrytym na klasę, a warstwa wyjściowa wykonuje proste sumowanie, to można powiedzieć, że powstała sieć nie ma warstwy ukrytej, a tylko trzy neurony w warstwie wyjściowej. Trenowanie sieci zajęło 1000 epok, a wagi połączeń w końcowej sieci odbiegają co najwyżej o 0.05 od wartości ± 1 lub od 0. Otrzymaliśmy zestaw wag i progów przedstawiony w tabeli 2 (dla zwiększenia przejrzystości podane są tylko znaki wag). Prosta ana-

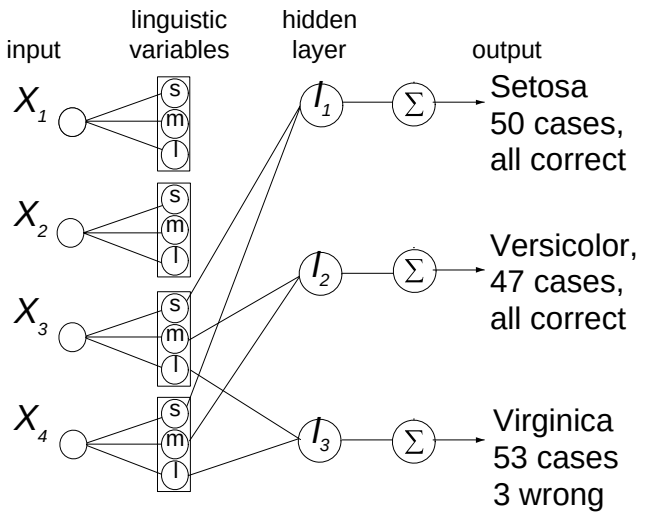
Tabela 2: Wagi i progi sieci C-MLP2LN dla irysów

Setosa	(0,0,0	0,0,0	+,0,0	+,0,0)	$\theta = 1$
Versicolor	(0,0,0	0,0,0	0,+,0	0,+,0)	$\theta = 2$
Virginica	(0,0,0	0,0,0	0,0,+	0,0,+)	$\theta = 1$

liza tych parametrów prowadzi do następującego zestawu reguł:

Iris-setosa if $x_3 = \mathrm{s} \vee x_4 = \mathrm{s}$
Iris-versicolor if $x_3 = \mathrm{m} \wedge x_4 = \mathrm{m}$
Iris-virginica if $x_3 = \mathrm{l} \vee x_4 = \mathrm{l}$

Tylko dwie cechy (x_3 i x_4) są używane w klasyfikacji - pozostałe dwie zostały wyeliminowane



Rysunek 4: Struktura sieci C-MLP2LN dla danych o irysach.

przez sieć. Pierwsza reguła klasyfikuje poprawnie wszystkie wektory z klasy iris-setosa. Cały zestaw reguł klasyfikuje poprawnie 147 przypadków (98%).

Analiza histogramów może być także bardzo przydatna, kiedy stosujemy sieci z jednostkami L. Kiedy startujemy od przypadkowych parametrów jednostek L sieć wymaga długiego okresu uczenia. Znacznie szybciej można znaleźć dobre rozwiązanie, jeśli zainicjujemy parametry jednostek L na podstawie analizy histogramów. W takiej sytuacji proces uczenia sieci poprawi początkowe ustawienia tak, żeby uzyskać stabilne i proste reguły. W zależności od sposobu używania parametrów regularyzacyjnych w trakcie trenowania sieci możemy uzyskać różne zestawy reguł. Na przykład wymuszając jak najprostsze reguły otrzymujemy zestaw, który klasyfikuje z dokładnością 96%:

Iris-setosa if $x_3 \leq 2.56$

Iris-virginica if $x_4 > 1.63$

Iris-versicolor otherwise

Stosując mniejsze wartości parametru odpowiedzialnego za zerowanie wag możemy dostać nieco bardziej złożone, ale i bardziej dokładne zestawy reguł.

Opisy złożoności i dokładności zestawów reguł stworzonych przez różne systemy zostały zebrane w tabeli 3.

Mushrooms. Wzorcowym przypadkiem pokazującym, że reguły generowane są od najbardziej ogólnych do najbardziej szczegółowych jest przykład 8124 danych o grzybach, spośród których 4208 przypadki (51.8%) to grzyby jadalne, a 3916 (48.2%) niejadalne. Każdy wektor opisany jest 22 symbolicznymi atrybutami, przyjmującymi w sumie 122 wartości. Nie trzeba więc tutaj przeprowadzać dyskretyzacji ani konstruować sieci z jednostkami L. Wygenerowane reguły i liczby obejmowanych przypadków prezentuje tabela 4.

Tabela 3: Reguły logiczne dla irysów. Typy reguł F=Fuzzy (rozmyte), C=Crisp (ostre), R=Rough (przybliżone), W=Weighted (ważone).

Metoda	Liczba reguł/przesłanek/cech	Typ	Dokładność
ReFuNN [Kasabov, 1996]	9/26/4	F	95.7
ReFuNN [Kasabov, 1996]	14/28/4	F	95.7
ReFuNN [Kasabov, 1996]	104/368/4	F	95.7
Grobian [Browne et al, 1998]	118/?/4	R	100
GA+NN [Jagielska et al, 1996]	6/6/4	W	100
NEFCLASS[Nauck i Kruse, 1996]	7/28/4	F	96.7
NEFCLASS[Nauck i Kruse, 1996]	3/6/2	F	96.7
FuNe-I[Halgamuge i Glesner, 1994]	7/?/3	F	96.0
C-MLP2LN	2/2/1	C	95.7
C-MLP2LN	2/2/2	C	96.0
C-MLP2LN	2/3/2	C	98.0
SSV	2/2/2	C	98.0

Tabela 4: Reguły dla danych o grzybach z liczbami obejmowanych przypadków

odor = $\neg$(almond $\vee$ anise $\vee$ none)	8004
spore-print-color = green	72
odor=none $\wedge$ stalk-surface-below-ring=scaly $\wedge$ stalk-color-above-ring = $\neg$brown	40
habitat = leaves $\wedge$ cap-color = white	8

Ten zestaw reguł jest jednym z najprostszych opisów badanego zbioru danych i klasyfikuje poprawnie wszystkie przypadki ze zbioru treningowego. Choć zbiór z danymi o grzybach wydaje się być dość łatwym do klasyfikacji (także na losowo wybranej próbce złożonej z 10% wektorów uzyskaliśmy ten sam zestaw reguł), to jednak nie wszystkie systemy tworzące opisy regułowe tworzą tak zwarte i dokładne reguły. Zestawienie znanych nam wyników dla tych danych przedstawia tabela 5.

Hypothyroid. Innym interesującym przypadkiem są dane o chorobach tarczycy. Ta baza zawiera wyraźny podział na dane treningowe i testowe, co znacznie ułatwia porównywanie wyników uzyskanych różnymi metodami. Zbiór treningowy zawiera 3772 wektorów, a testowy 3428. Każdy z przypadków jest opisany 22 atrybutami (15 binarnych i 6 ciągłych) i należy do jednej z trzech klas: primary hypothyroid, compensated hypothyroid i normal. Rozkład danych w klasach jest tutaj bardzo nierówny: w zbiorze treningowym klasy mają odpowiednio 93, 191 i 3488 reprezentantów w zbiorze treningowym oraz 73, 177 i 3178 w zbiorze testowym.

Metoda MLP2LN pozwoliła znaleźć zestaw 4 reguł, które klasyfikują poprawnie 99.68% wek-

Tabela 5: Mushroom - zestawienie wyników.

Metoda	Liczba reguł/przesłanek/cech	Dokładność
RULENEG[Sestito i Dillon, 1994]	300/8087	91.0
REAL [Craven i Shavlik, 1994]	155/6603	98.0
DEDEC [Tickle et al, 1994]	26/26	99.8
TREX[Andrews et al, 1995]	3/13	100
C4.5 (decision tree)	3/3	99.8
RULEX[Andrews i Geva, 1994]	1/3/1	98.5
Successive Regulariz.[Duch et al, 1997]	1/4/2	99.4
Successive Regulariz.[Duch et al, 1997]	2/22/4	99.9
Successive Regulariz.[Duch et al, 1997]	3/24/6	100
C-MLP2LN, SSV	1/3/1	98.5
C-MLP2LN, SSV	2/4/2	99.4
C-MLP2LN	3/7/4	99.9
SSV	3/7/4	99.9
C-MLP2LN	4/9/6	100
SSV	4/9/5	100

torów treningowych i 99.07% testowych. Do opisania pierwszej klasy wystarczyły dwie reguły (wszystkie wartości ciągłych atrybutów zostały przemnożone przez 1000):

R_{11}: FTI $< 63 \wedge$ TSH ≥ 29
R_{12}: FTI $< 63 \wedge$ TSH $\in [6.1, 29) \wedge$ T3< 20
Drugą klasę opisuje jedna reguła:

R_2: FTI $\in [63, 180] \wedge$ TSH $\geq 6.1 \wedge$on thyroxine=no $\wedge$ surgery=no
Trzecia klasa jest reprezentowana jako uzupełnienie sumy reguł dla pierwszych dwóch klas.

Po optymalizacji powyższych reguł otrzymujemy nieco dokładniejszy zestaw:

R_{11}: TSH $\geq 30.48 \wedge$ FTI < 64.27
R_{12}: TSH $\in [6.02, 29.53] \wedge$ FTI $< 64.27 \wedge$ T3< 23.22
R_2: TSH $\geq 6.02 \wedge$ FTI $\in [64.27, 186.71] \wedge$ TT4$\in [50, 150.5) \wedge$ on thyroxine=no $\wedge$ surgery=no
Tak poprawione reguły klasyfikują błędnie tylko 4 wektory ze zbioru treningowego (99.89% poprawności) i 22 ze zbioru testowego (99.36% poprawności). Bardzo podobny zestaw znaleźli Weiss i Kapouleas używając heurystycznej wersji metody PVM [Weiss i Kapouleas, 1990].

Różnice w wynikach systemów PVM, CART i C-MLP2LN są bardzo małe (tabela 6), ale inne metody włączając w to optymalizowane MLP (także metodami genetycznymi [Shiffman et al, 1993]) i korelację kaskadową, dają dwukrotnie i więcej razy większy błąd (1.5%) dla zbioru testowego. Fakt ten pokazuje, że w niektórych przypadkach ostre granice decyzji mogą być bardziej skuteczne niż łagodne przejścia pomiędzy klasami realizowane metodami neuronowymi. W takich przypadkach również metody minimalnoodległościowe okazują się bardzo nieskuteczne.

Tabela 6: Poprawności klasyfikacji (w procentach) dla zbioru hypothyroid.

Metoda	zbiór treningowy	zbiór testowy
CART [Weiss i Kapouleas, 1990]	99.79	99.36
PVM [Weiss i Kapouleas, 1990]	99.79	99.33
Cascade correl. [Shiffman et al, 1993]	100.00	98.5
MLP+backprop [Shiffman et al, 1993]	99.60	98.5
3-NN, 3 features used	98.7	97.9
Bayes [Weiss i Kapouleas, 1990]	97.0	96.1
k-NN [Weiss i Kapouleas, 1990]	–	95.3
C-MLP2LN	99.89	99.36
SSV rules	99.79	99.33
FSM 10 rules	99.60	98.90

7 PODSUMOWANIE

Sieci neuronowe znalazły liczne zastosowania do odkrywania wiedzy w bazach danych. Metody te osiągają w wielu przypadkach bardzo dobre wyniki, znajdując reguły działające równie dokładnie jak najlepsze klasyfikatory nieregułowe. Co więcej, dla niektórych danych medycznych reguły logiczne odkryte za pomocą sieci neuronowych są znacznie dokładniejsze niż same sieci oraz wszystkie inne klasyfikatory (np. dla danych „hypothyroid" z UCI). Najprawdopodobniej wynika to z faktu, że lekarze podejmując decyzje kierują się przesłankami, które przybierają właśnie charakter reguł. Znalezienie dobrych rozwiązań o bardzo ostrych granicach decyzji okazuje się trudnym zadaniem dla sieci neuronowych i innych klasyfikatorów. Dodatkowym atutem reguł logicznych jest łatwe ustalenie optymalnej złożoności klasyfikatora po odrzuceniu reguł zbyt specyficznych. Ustalenie odpowiedniej architektury sieci neuronowej jest zadaniem trudniejszym.

Metody neuronowe nie są oczywiście jedynymi metodami przydatnymi do odkrywania wiedzy w bazach danych. Żaden algorytm nie jest lepszy od pozostałych we wszelkich zastosowaniach, więc w konkretnym przypadku należy zwykle sięgnąć po kilka metod i porównać ich wyniki. Niestety bardzo mało jest prac dyskutujących przydatność różnych metod do różnego rodzaju zastosowań, co znacznie ułatwiałoby szukanie najbardziej odpowiedniego systemu do analizy konkretnych danych. Z naszych doświadczeń wynika, że dużą konkurencję dla sieci neuronowych stanowią metody oparte na drzewach decyzji [Grąbczewski i Duch, 1999].

W wielu problemach zastosowanie reguł logicznych jako formy reprezentacji wiedzy może okazać się niewystarczające.

Podziękowania: za wsparcie finansowe jesteśmy wdzięczni Komitetowi Badań Naukowych, grant nr. 8 T11F 014 14.

BIBLIOGRAFIA

[Alexander i Mozer, 1995] J.A. Alexander, M.C. Mozer, “Template-based algorithms for connectionist rule extraction”. In: G. Tesauro, D. Touretzky, T. Leen, eds, Advances in Neural Information Processing Systems 7. MIT Press, Cambridge, MA, 1995

[Andrews et al, 1995] R. Andrews, J. Diederich, A.B. Tickle, “A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks”, Knowledge-Based Systems vol. 8, str. 373–389, 1995.

[Andrews i Geva, 1994] R. Andrews, S. Geva, “Rule extraction from a constrained error back propagation MLP”. Proc. 5th Australian Conference on Neural Networks, Brisbane, Queensland 1994, str. 9-12

[Andrews i Geva, 1996] R. Andrews, S. Geva, “Rules and Local Function Networks”. In: R. Andrews, J. Diederich (Eds), *Rules and Networks*, Proc. of the Rule Extraction From Trained Artificial Neural Networks Workshop, AISB96, Brighton UK, April 1996; R. Andrews, S. Geva, “Refining Expert Knowledge with an Artificial Neural Network". Int. Conf. on Neural Information Processing, New Zealand, Nov.1997, Vol. 2, str. 847-850

[Browne et al, 1998] C. Browne, I. Düntsch, G. Gediga “IRIS revisited: A comparison of discriminant and enhanced rough set data analysis". In: L. Polkowski and A. Skowron, eds. *Rough sets in knowledge discovery*, vol. 2. Physica Verlag, Heidelberg, 1998, pp. 345-368

[Craven, 1996] M. W. Craven “Extracting comprehensible models from trained neural networks”, PhD thesis, 1996

[Craven i Shavlik, 1994] M. W. Craven, J.W. Shavlik, “Using sampling and queries to extract rules from trained neural networks”. In: Proc. of the Eleventh Int. Conference on Machine Learning, New Brunswick, NJ. Morgan Kaufmann 1994, str. 37-45

[Duch, 1994] W. Duch, “Floating Gaussian Mapping: a new model of adaptive systems”. Neural Network World 4:645-654, 1994

[Duch et al, 1999a] W. Duch, R. Adamczak and K. Grąbczewski. *Neural optimization of linguistic variables and membership functions,* ICONIP’99, Perth, Australia

[Duch et al, 1999b] W. Duch, R. Adamczak and K. Grąbczewski. *Methodology of extraction, optimization and application of logical rules*, IIS’99, Ustroń, Poland

[Duch et al, w druku] W. Duch, R. Adamczak, K. Grąbczewski, *Methodology of extraction, optimization and application of crisp and fuzzy logical rules*. IEEE Transactions on Neural Networks (w druku)

[Duch et al, 1997] W. Duch, R. Adamczak, K. Grąbczewski, M. Ishikawa, H. Ueda, "Extraction of crisp logical rules using constrained backpropagation networks - comparison of two new approaches", Proc. of the European Symposium on Artificial Neural Networks (ESANN'97), Bruge 16-18.4.1997, pp. 109-114

[Duch et al, 1995] W. Duch, G.H.F. Diercksen, *Feature Space Mapping as a universal adaptive system*, Computer Physics Communication 87: 341–371, 1995

[Duch i Grąbczewski, 1999] W. Duch and K. Grąbczewski, "Searching for optimal MLP". Fourth Conference on Neural Networks and Their Applications, Zakopane, May 1999, pp. 65-70

[Fu, 1991] L.M. Fu, "Rule learning by searching on adapted nets", Proceedings of the Ninth National Conference on Artificial Intelligence (Anaheim CA) (1991) 590-595

[Fu, 1993] L.M. Fu, "Knowledge-based connectionism for revising domain theories", IEEE Transactions on Systems, Man, and Cybernetics, 23 (1993) 173-182

[Fu, 1994a] L.M. Fu, "Neural networks in computer intelligence", McGraw Hill (New York) (1994)

[Fu, 1994b] L.M. Fu, "Rule generation from neural networks", IEEE Transactions on Systems, Man, and Cybernetics 28 (1994) 1114-1124

[Gallant, 1993] S. Gallant, "Neural Network Learning and Expert Systems". MIT Press, Cambridge, MA 1993

[Geczy i Usui, 1997] Geczy P, Usui S, "Rule extraction from trained neural networks". Int. Conf. on Neural Information Processing, New Zealand, Nov.1997, Vol. 2, str. 835-838

[Grąbczewski i Duch, 1999] Grąbczewski K, Duch W, "A general purpose separability criterion for classification systems". 4-ta konferencja sieci neuronowych i ich zastosowań, Zakopane, Maj 1999, str. 203-208

[Halgamuge i Glesner, 1994] Halgamuge S.K, Glesner M, "Neural networks in designing fuzzy systems for real world applications". Fuzzy Sets and Systems 65:1-12, 1994.

[Hayashi, 1991] Y. Hayashi, "A neural expert system with automated extraction of fuzzy if-then rules". In: Lippmann, R., Moody, J., Touretzky, D., eds, Advances in Neural Information Processing Systems (vol. 3). Morgan Kaufmann, San Mateo, CA 1991

[Healy i Caudell, 1997] M.J. Healy, T.P. Caudell, "Acquiring Rule Sets as a Product of Learning in a Logical Neural Architecture", *IEEE Trans. Neural Networks*, vol 8, str. 461–474, 1997

[Ishikawa, 1996] Ishikawa M, "rule extraction by succesive regularization". in: proc. of 1996 ieee int. conf. on neural networks. washington, 1996, str. 1139–1143.

[Jagielska et al, 1996] I. Jagielska, C. Matthews, T. Whitfort, “The application of neural networks, fuzzy logic, genetic algorithms and rough sets to automated knowledge acquisition”. 4th Int. Conf. on Soft Computing, IIZUKA’96, Iizuka, Japan, 1996, vol. 2, pp. 565-569

[Jang i Sun, 1993] J-S. R. Jang, C.T. Sun, “Functional Equivalence Between Radial Basis Function Neural Networks and Fuzzy Inference Systems,” *IEEE Trans. on Neural Networks* 4, no. 1, pp. 156–158, 1993.

[Kanal i Kumar, 1988] L. Kanal, V. Kumar (Eds), *Search in Artificial Intelligence*. Springer Verlag, 1988

[Kasabov, 1996] N. Kasabov, “Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering”, The MIT Press (1996).

[Kasabov et al, 1998] Kasabov N, Kozma R, Duch W, “Rule extraction from linguistic rule Networks and from Fuzzy Neural Networks: Propositional versus Fuzzy Rules”. Fourth Int. Conf. on Neural Networks and their Applications, March 11-13, 1998, Marseille, France, str. 403-406

[McMillan et al, 1992] C. McMillan, M.C. Mozer, P. Smolensky, “Rule induction through integrated symbolic and subsymbolic processing”. In: J. Moody, S. Hanson, R. Lippmann, eds, Advances in Neural Information Processing Systems (vol. 4). Morgan Kaufmann, San Mateo, CA 1992

[Nauck i Kruse, 1996] Nauck D, Kruse R, “Designing neuro-fuzzy systems through backpropagation”. In: W. Pedrycz, ed, Fuzzy Modelling: Paradigms and Practice, pages 203-228. Kluwer, Boston, 1996.

[Nauck et al, 1996] Nauck D, Nauck U, Kruse R, “Generating Classification Rules with the Neuro-Fuzzy System NEFCLASS”. Proc. Biennial Conf. of the North American Fuzzy Information Processing Society (NAFIPS’96), Berkeley, 1996.

[Pawlak, 1998] Z. Pawlak, “Rough sets - theoretical aspects of reasoning about data”, Kluver Academic Publishers 1991; L. Polkowski, A. Skowron (eds.), “Rough Sets in Knowledge Discovery 1. Methodology and Applications”, and “Rough Sets in Knowledge Discovery 2. Applications, Case Studies and Software Systems”, Physica–Verlag, Heidelberg, 1998; L. Polkowski, A. Skowron (eds.), “Rough Sets and Current Trends in Computing”, Lecture Notes in Computer Science 1424, Springer-Verlag, 1998; S. K. Pal, A. Skowron, “Rough Fuzzy Hybridization A New Trend in Decision-Making”, Springer-Verlag, 1999.

[Pop et al, 1996] E. Pop, R. Hayward, J. Diederich, “RULENEG: extracting rules from a trained ANN by stepwise negation”, QUT NRC technical report, December 1994; R. Hayward, C. Ho-Stuart, J. Diederich and E. Pop, “RULENEG: extracting rules from a trained ANN by stepwise negation”, QUT NRC technical report, January 1996

[Saito i nakano, 1988] K. Saito, R. Nakano, “Medical diagnostic expert system based on PDP model”, Proc. of IEEE Int. Conf. on Neural Networks (San Diego CA), Vol 1 (1988) 255-262

[Shiffman et al, 1993] W. Schiffman, M. Joost and R. Werner, “Comparison of optimized backpropagation algorithms”. Proc. of European Symposium on Artificial Neural Networks, De facto Publications, Brussels 1993, pp. 97-104

[Sestito i Dillon, 1994] S. Sestito, T. Dillon, “Automated knowledge acquisition”. Prentice Hall (Australia), 1994

[Sethi i Yoo, 1994] I.K. Sethi, J.H. Yoo, “Symbolic approximation of feedforward neural networks.” In: E.S. Gelsema, L.N. Kanal, eds, Pattern Recognition in Practice (vol. 4). North-Holland, New York, NY 1994.

[Setiono i Liu, 1995] Setiono R, Liu H, “Understanding neural networks via rule extraction”. In: Proc. of the 14th Int. Joint Conference on Artificial Intelligence, Montreal, Quebec. Morgan Kaufmann, 1995, str. 480-485

[Tan, 1994] A-H. Tan, “Rule learning and extraction with self-organizing neural networks”. In: Proc. of the 1993 Connectionist Models Summer School, Hillsdale, NJ. Lawrence Erlbaum Associates 1994, str. 192-199

[Thrun, 1995] S. Thrun, “Extracting rules from artifcial neural networks with distributed representations”. In: G. Tesauro, D. Touretzky, T. Leen, eds, Advances in Neural Information Processing Systems 7. MIT Press, Cambridge, MA, 1995

[Tickle et al, 1998] A. B. Tickle, R. Andrews, M. Golea, and J. Diederich, “The Truth Will Come to Light: Directions and Challenges in Extracting the Knowledge Embedded Within Trained Artificial Neural Networks”. *IEEE Trans. Neural Networks*, vol 9, pp. 1057–1068, 1998

[Tickle et al, 1994] A.B. Tickle, M. Orlowski, J. Diederich, “DEDEC: decision detection by rule extraction from neural networks”, QUT NRC technical report, September 1994

[Towell i Shavlik, 1993] G. Towell, J. Shavlik, “Extracting refined rules from knowledge-based neural networks”. Machine Learning 13 (1993) 71-101

[Towell i Shavlik , 1994] G. Towell, J. Shavlik, “Knowledge-based artificial neural networks.”, Artificial Intelligence 70 (1994) 119-165

[Ultsch, 1993] A. Ultsch, “Knowledge extraction from self-organizing neural networks”, In: O. Opitz and B. Lausen and R. Klar, eds. *Information and Classification*, Springer, Berlin, 1993, p. 301-306

[Weiss i Kapouleas, 1990] S.M. Weiss, I. Kapouleas. “An empirical comparison of pattern recognition, neural nets and machine learning classification methods”, in: *Readings in Machine Learning*, eds. J.W. Shavlik, T.G. Dietterich, Morgan Kauffman Publ, CA 1990

[Żurada i Łozowski, 1996] Żurada J.M, Łozowski A, “Generating Linguistic Rules from Data Using Neuro-Fuzzy Framework”. 4th Int. Conf. on Soft Computing, IIZUKA’96, Iizuka, Japan, 1996, vol. 2, str. 618-621.