Konstruktywna indukcja i rozmywanie dla poprawy jakości reguł z liniowymi konkluzjami

Marek Sikora[1]

Streszczenie: W artykule przestawiono metody poprawy jakości zbioru reguł z liniowymi konkluzjami otrzymanymi za pomocą algorytmy M5. Algorytm M5 pozwala na uzyskiwanie w bardzo krótkim czasie, regułowego modelu predykcyjnego. W pracy przedstawiono architekturę rozmytej sieci neuronowej, która zwiększa czytelność wyznaczonego przez M5 modelu danych oraz wykorzystano konstruktywną indukcję w celu poprawy zdolności predykcyjnych tego algorytmu.

Słowa kluczowe: predykcja, reguły, zdolności opisowe, rozmyta sieć neuronowa, konstruktywna indukcja.

1. Wstęp

Metody stosowane do analizy szeregów czasowych to m.in. podejście klasyczne wykorzystujące aparat statystyki matematycznej (Box, Jenkins, 1994), metody wykorzystujące logikę rozmytą (Wang, 1994; Yager, Filev 1994) oraz sieci neuronowe (Shavlik, Optiz 1997; Tadeusiewicz 1993).

Bardzo efektywne z punktu widzenia osiąganych wyników są algorytmy hybrydowe, tworzące modele rozmyto-neuronowe (Czogała, Łęski, 2002; Oh, Pedrycz, 2004; Rutkowski, Cpałka 2003). Algorytmy te zazwyczaj budują sieć w oparciu o wynik grupowania danych. Liczba grup oraz liczba zbiorów rozmytych dzielących dziedzinę każdej cechy ustalane są arbitralnie lub adaptacyjnie. Zazwyczaj rozmyta sieć neuronowa zbudowana jest z reguł, w przesłankach których znajdują się singeltony lub koniunkcje wszystkich zmiennych niezależnych. W konkluzji reguł znajduje się zmienna zależna określona za pomocą zbioru rozmytego lub kombinacji liniowej zmiennych niezależnych. Wadą rozbudowanych architektur rozmytych sieci neuronowych jest to, że użytkownikowi trudno w prosty i przejrzysty sposób zintepretować uzyskany model danych.

Na tym polu stosunkowo małą popularnością cieszy się metoda zaproponowana przez Quinlana (1993), polegająca na wyznaczeniu zbioru reguł, w konkluzjach których znajdują się modele liniowe opisujące lokalne zależności pomiędzy danymi (konkretyzacją metody jest algorytm M5). Algorytm M5 wraz z dołączoną do niego procedurą wygładzania (ang. smoothing) uzyskuje bardzo dobre wyniki w zakresie dokładności predykcji i czasu działania. Zastosowanie wygładzania dramatycznie pogarsza jednak zdolności opisowe uzyskanych reguł, gdyż proces ustalana wartości zmiennej zależnej nie przebiega zgodnie z zależnościami wyrażanymi przez reguły, które widzi użytkownik.

[1] Instytut Informatyki, Politechnika Śląska, Akademicka 16, 44-100 Gliwice
e-mail: Marek.Sikora@polsl.pl

W dalszej części artykułu przedstawiono zarys algorytmu M5 oraz dwie propozycje umożliwiające polepszenie zarówno zdolności opisowych (które mają szczególne znaczenie w eksploracji danych), jak i predykcyjnych reguł wyznaczanych przez M5. W celu polepszenia zdolności opisowych proponujemy zamianę wyznaczonych przez M5 reguł w sieć rozmyto-neuronową, w której zbyteczna jest procedura wygładzania. Aby poprawić zdolności predykcyjne proponujemy zastosowanie schematu konstruktywnej indukcji sterowanej hipotezami (Wnek, Michalski, 1994). W pracy przedstawiono także wyniki przeprowadzonych eksperymentów.

2. Reguły o liniowych konkluzjach

Idea algorytmu M5 została zaczerpnięta z tzw. drzew regresji i klasyfikacji CART (Breiman i in., 1994) oraz z algorytmu C4.5 dokonującego indukcji drzew decyzyjnych (Quinlan, 1994). Algorytm M5 analizuje zbiór treningowy $Tr = (U, A \cup \{y\})$ i pozwala na tworzenie reguł w postaci (1)

$$\text{IF } a_{i1} \in V_{a_{i1}} \text{ and} \ldots \text{and } a_{ij} \in V_{a_{ij}} \text{ THEN } y = b_{i1}w_{i1} + \cdots + b_{il}w_{il} + w_i, \quad (1)$$

gdzie $\{a_{i1}, \ldots, a_{ij}\} \subseteq A$; $V_{a_{i1}} \subseteq D_{a_{i1}}, \ldots, V_{a_{ij}} \subseteq D_{a_{ij}}$; $\{b_{i1}, \ldots, b_{il}\} \subseteq A$; $w_i, w_{i1}, \ldots, b_{il} \in \mathbf{R}$.

Wyrażenie $a \in V_a$ nazywane jest deskryptorem warunkowym. Zmienne znajdujące się w przesłance reguły (1) mogą być różne od tych znajdujących się w konkluzji. W konkluzji reguły występować mogą jedynie zmienne typ numerycznego. Elementy zbioru A nazywamy zmiennymi niezależnymi, które potraktować można jako funkcje $a\colon U \to D_a$, gdzie D_a jest dziedziną zmiennej a. Zmienną y nazywamy zmienną zależną, a elementy zbioru U przykładami.

Przyjmijmy, że prawa strona wyrażenia (1) może zostać zapisana jako $y = f(\mathbf{x})$, gdzie $\mathbf{x} = (x_1, x_2, .., x_k)$, oraz x_i jest pewną wartością zmiennej $a_i \in A$.

Algorytm M5 wykorzystuje rekurencyjną procedurę budowy tzw. drzewa modeli. Na każdym etapie tworzenia drzewa wywoływana jest procedura sprawdzająca która zmienna $a \in A$ oraz jaka wartość graniczna q dokona najlepszego podziału zbioru przykładów związanego z danym węzłem.

W każdym węźle drzewa występuje zatem podział zbioru obiektów związanych z tym węzłem na te, dla których wartość atrybutu a jest większa od q oraz te, dla których jest mniejsza od q. Dowolna wartość q ma taką własność, że dla ustalonej zmie-nnej a, istnieją takie wartości $v_1, v_2 \in D_a$, że $v_1 < v_2$ oraz $q = (v_2 - v_1)/2$. W przypadku atrybutów symbolicznych stosowana jest procedura wyczerpująca polegająca na badaniu wszystkich możliwych podzbiorów zbioru wartości atrybutu symbolicznego.

Oznaczmy przez P zbiór przykładów związanych z węzłem drzewa. W każdym węźle najlepszym atrybutem i punktem granicznym jest taki punkt, dla którego podział zbioru P na podzbiory $P_{<q}$ i $P_{>q}$ minimalizuje oczekiwaną wariancję zmiennej zależnej (tzn. maksymalizuje wartość wyrażenia (2)).

$$\Delta err = V(P) - \Big(\frac{|P_{<q}|}{|P|}V(P_{<q}) + \frac{|P_{>q}|}{|P|}V(P_{>q})\Big), \quad (2)$$

gdzie $V(P)$ oznacza wariancję zmiennej zależnej w zbiorze przykładów P.

W każdym węźle (nie tylko w liściach) wyznaczana jest funkcja liniowa f zdefiniowana na zbiorze A, której zadaniem jest predykcja zmiennej zależnej w danym węźle drzewa.

Jeśli w danym węźle najlepszy z możliwych podziałów nie zmniejsza już oczekiwanej wariancji zmiennej zależnej, to procedura rozbudowy drzewa zatrzymuje się (węzeł staje się liściem).

Po utworzeniu drzewa, uruchamiana jest procedura obcięcia. Jeśli błąd bezwzględny utworzonego w danym węźle modelu liniowego jest mniejszy niż błąd w węzłach lub liściach znajdujących się poniżej, to drzewo jest przycinane i rozpatrywany węzeł staje się liściem.

Po utworzeniu drzewa struktura ta zamieniana jest na zbiór reguł. Reguły uzyskujemy idąc od korzenia drzewa do każdego liścia (reguł będzie tyle ile jest liści). W ten sposób powstaje część przesłankowa reguły. Konkluzją reguły staje się funkcja f znajdująca się w liściu. Użytkownik nie widzi zatem modeli liniowych tworzonych w kolejnych węzłach nie będących liśćmi.

Przedstawiony algorytm jest niezwykle szybki, a uzyskane reguły w sposób prosty opisują lokalne zależności pomiędzy zmiennymi niezależnymi i zmienną zależną. Zdolność predykcji tak uzyskanego zbioru reguł nie jest jednak najlepsza (Quinlan, 1994; Sikora, Kozielski, 2006). Aby znacząco poprawić zdolności predykcyjne uzyskanych reguł algorytm M5 stosuje procedurę wygładzania.

Procedura wygładzania polega na dostrojeniu wartości zmiennej zależnej przewidywanej przez regułę za pomocą tzw. reguł częściowych (program realizujący algorytm M5 pamięta kolejność dodawania deskryptorów warunkowych do reguły) Procedura wygładzania przebiega w następujący sposób:

- wartość zmiennej zależnej określana jest dla wszystkich reguł częściowych (począwszy od reguły wyjściowej r, poprzez reguły $r_{-1}, r_{-2}, \ldots$, aż do reguły r_{root}, która jest regułą bez jakichkolwiek deskryptorów warunkowych),
- wartość przekazywana do reguły r_{-1} jest wartością określoną przez regułę r,
- załóżmy, że dane są dwie reguły częściowe r_{-i} i r_{-i-1}, wartość przekazywana z reguły r_{-i} do reguły r_{-i-1} określana jest w sposób następujący:

$$PV(r_{-i-1}) = \frac{n_{-i}PV(r_{-i}) + kM(r_{-i-1})}{n_{-i} + k} \tag{3}$$

  gdzie: $n_{-i} = |match(r_{-i})|$, $match(r_{-i})$ jest liczbą obiektów ze zbioru U, które spełniają część warunkową reguły r_{-i}; k jest pewną ustaloną stałą; $M(r_{-i-1})$ jest wartością przewidywaną przez regułę częściową r_{-i-1}; $PV(r_{-i}), PV(r_{-i-1})$ są wartościami przekazywanymi do reguł częściowych r_{-i}, r_{-i-1}.

- ostatecznie, wartością zmiennej zależnej przewidywaną przez regułę r jest wartość zwracana przez regułę częściową r_{root}.

Widać zatem, że w czasie stosowania procedury wygładzania ustalanie wartości zmiennej zależnej następuje według innych kryteriów niż wynika to z zależności zawartych w wyznaczonych regułach, gdyż użytkownik nie zna konkluzji reguł $r_{-1}, r_{-2}, \ldots, r_{root}$.

Przykład 1
Załóżmy, że dana jest funkcja syntetyczna $y = 3x^5 - x^4 - x$, określona na przedziale $[-1, 1]$. Algorytm M5 wygenerował trzy reguły:

$r1$: if $x < -0.7073$ then $y = 7.39948 + 10.31x$
$r2$: if $x \in (-0.7073, 0.5121)$ then $y = -0.0545261 - 0.57x$
$r3$: if $x > 0.5121$ then $y = -4.71678 + 5.7x$

Dla obiektu testowego $x = 0.561$ zapaliła się reguła $r3$, która powinna zwrócić wartość $y = -1.519$, tymczasem M5 z wygładzaniem zwrócił wartość $y = -0.474$, a więc znacząco bliżej rzeczywistej wartości y, która obliczona ze wzoru funkcji powinna wynosić -0.493. Zauważmy, że dla reguły $r3$, reguła $r3_{-1} = r3_{root}$. Konkluzja $r3_{root}$ obliczona przez M5 jest stałą $y = -0.219822$. Regułę $r3$ rozpoznawały trzy obiekty. W procedurze wygładzania wykorzystano stałą $k = 7$, zatem ostateczna wartość zmiennej y obliczona została jako $((3\cdot(-1.519+7)\cdot(-0.219822))/(3+7)) = -0.474$, co jak widać znacząco odbiega od tego co proponuje reguła $r3$ bez procedury wygładzania.

Modyfikacje algorytmu M5 objemowały do chwili obecnej interaktywne tworzenie drzewa (Solomatine i in., 2004) oraz możliwość stosowania algorytmu dla danych niekompletnych. Na algorytmach M5 oraz MODLEM (Stefanowski 2001) bazuje również algorytm tworzący reguły, w konkluzjach których znajduje się pewien przedział zmiennej zależnej (Sikora, Kozielski 2006).

3. Konstruktywna indukcja - większa dokładność kosztem czytelności reguł

Konstruktywna indukcja (CK) polega na wprowadzeniu do zbioru zmiennych niezależnych nowej zmiennej, której wartości zależą funkcyjnie (CK sterowana danymi) lub logicznie (CK sterowana hipotezami (Wnek, Michalski, 1994)) od pierwotnego zbioru zmiennych.

Obecnie do zapisania przykładów treningowych wykorzystamy zapis wektorowy, na przykład $\mathbf{x}_i$ ma postać $[x_1, x_2, \ldots, x_k]$, gdzie $\forall i \in 1, 2, .., k,\ x_i \in D_{a_i}$. Każdą regułę postaci (1) możemy przedstawić jako $\varphi \rightarrow y = f(\mathbf{x})$, gdzie $f\colon A \rightarrow R$. Powiemy, że przykład $\mathbf{x}_i$ rozpoznaje regułę $\varphi \rightarrow y = f(\mathbf{x})$, wtedy i tylko wtedy, gdy dla każdego deskryptora $a \in V_a$, występującego w przesłance reguły $a(\mathbf{x}_i) \in V_a$.

Z algorytmu konstrukcji drzewa wynika, że zawsze jedna i tylko jedna reguła ma wpływ na ustalenie wartości przewidywanej przez model. Proponowany schemat wykorzystania konstruktywnej indukcji w algorytmie M5 jest następujący:

1. wyznacz zbiór reguł na podstawie zbioru treningowego,
2. począwszy od drugiego przykładu trenującego, wyznacz różnicę $\Delta y_i = y_{i-1} - f(\mathbf{x}_{i-1})$ rzeczywistej wartości zmiennej zależnej i odpowiedzi modelu; w przedstawionej różnicy wyrażenie $f(\mathbf{x}_{i-1})$ jest modelem znajdującym się w konkluzji reguły, która została rozpoznana przez i-ty przykład,
3. usuń pierwszy przykład ze zbioru treningowego oraz dodaj zmienną Δy do zbioru cech niezależnych,
4. wyznacz zbiór reguł dla nowego zbioru przykładów treningowych.

Z powyższego schematu wynika, że nowa zmienna mierzy błąd popełniany przez reguły otrzymane w pierwszym kroku. Wprowadzając zmienną Δy do zbioru zmiennych niezależnych mamy nadzieję na lepsze dostrojenie modelu do danych treningowych. Aby uniknąć efektu nadmiernego dopasowania (przeuczenia), przedstawiony schemat stosujemy tylko raz.

Algorytm M5 operuje jedynie na wektorze zmiennych zawartych w zbiorze A. Uwzględnienie opóźnień zmiennych występujących w tym zbiorze możliwe jest jedynie poprzez wprowadzenie do niego nowych zmiennych, opóźnionych w stosunku do zmiennych już istniejących. Przedstawiony schemat konstruktywnej indukcji tworzy właśnie taką możliwość.

Celem CK jest poprawa zdolności predykcyjnych, dzieje się to jednak kosztem prostoty uzyskanych reguł. Po zastosowaniu CK możemy uzyskać cztery postaci reguł:

1. $\varphi \rightarrow y = f(\mathbf{x})$, brak nowej zmiennej w regule,
2. $\varphi \wedge \Delta y \rightarrow y = f(\mathbf{x})$, nowa zmienna występuje w przesłance reguły,
3. $\varphi \rightarrow y = f(\mathbf{x}')$, gdzie $\mathbf{x}' = [x_1, x_2, \ldots, x, \Delta y]$, nowa zmienna występuje w konkluzji reguły,
4. $\varphi \wedge \Delta y \rightarrow y = f(\mathbf{x}')$, nowa zmienna występuje zarówno w przesłance, jak i w konkluzji reguły.

W przypadku pierwszym, uzyskana reguła może być tak samo prosta jak reguły wyznaczane bez konstruktywnej indukcji, gdyż zmienna Δy nigdzie w regule tej nie występuje. We wszystkich pozostałych przypadkach reguły stają się trudniejsze do interpretacji. Jeśli reguła ma postać taką jak w punktcie 2, to zmienna Δy występująca w przesłance reguły jest zmienną typu numerycznego, deskryptor $\Delta y \in V_{\Delta y}$ możemy zapisać jako $\Delta y \in (q_{\Delta y1}, q_{\Delta y2})$. Rozważając kolejno wszystkie przykłady rozpoznające tę regułę możemy, stosując odpowiednie podstawienia, uzyskać jawną (bez zmiennej Δy) postać przesłanek w tej regule. Dla przykładu $\mathbf{x}_i$ rozpoznającego rozważaną regułę, za wyrażenie $\Delta y_i \in (q_{\Delta y1}, q_{\Delta y2})$ możemy podstawić $(y_{i-1} - f(\mathbf{x}_{i-1})) \in (q_{\Delta y1}, q_{\Delta y2})$. W podstawieniu tym, wyrażenie $f(\mathbf{x}_{i-1})$ jest funkcją występującą w konkluzji jakiejś reguły uzyskanej przed zastosowaniem konstruktywnej indukcji. Reguła ta rozpoznawana jest przez przykład $\mathbf{x}_{i-1}$.

Jeśli reguła ma taką postać jak w punkcie 3, to w jej konkluzji występuje wyrażenie $w\Delta y$. Rozważając kolejno wszystkie przykłady rozpoznające tą regułę możemy, stosując odpowiednie podstawienia, uzyskać jawną (bez zmiennej Δy) postać konkluzji tej reguły. Dla przykładu $\mathbf{x}_i$ rozpoznającego rozważaną regułę za wyrażenie $w\Delta y$ możemy podstawić $w(y_{i-1} - f(\mathbf{x}_{i-1}))$, gdzie $f(\mathbf{x}_{i-1})$ spełnia warunki identycz-ne jak w opisie przypadku 2.

Powyższe rozumowanie pokazuje, że reguły uzyskane poprzez zastosowanie CK stają się mniej czytelne. Dodatkowo można zauważyć, że jeśli założymy, iż przykład $\mathbf{x}_i$ rozpoznaje reguły opisane w punktach 2, 3, 4, to przykład x_{i-1} może rozpoznawać dowolną z reguł wyznaczonych przez zastosowaniem konstruktywnej indukcji. Zapiszmy to bardziej formalnie, przez $supp(r)$ oznaczmy zbiór przykładów rozpoznających regułę r. Jeśli reguła r ma jedną z postaci opisanych w punktach 2, 3, 4, to dla każdego przykładu $\mathbf{x}_i \in supp(r)$ istnieje przykład $\mathbf{x}_{i-1}$, który rozpoznaje

pewną regułę $\varphi \rightarrow y = f(\mathbf{x})$, wyznaczoną w pierwotnej przestrzeni cech. Funkcja $f(\mathbf{x})$ z konkluzji reguły $\varphi \rightarrow y = f(\mathbf{x})$ przechodzi do deskryptora (przypadek 2, 4) lub staje się częścią konkluzji (przypadek 3, 4) reguły r. Można zauważyć, że zbiór $supp(r)$ determinuje również zbiór $supp_{-1}(r)$, do którego należą wszystkie wspomniane przykłady $\mathbf{x}_{i-1}$. Ważne jest to, że obiekty należące do zbioru $supp_{-1}(r)$ mogą rozpoznawać więcej niż jedną z reguł wyznaczonych w wejściowej przestrzeni cech. Powoduje to dalsze komplikacje w interpretacji uzyskanego modelu danych, gdyż deskryptor $\Delta y \in V_{\Delta y}$ i wyrażenie $w\Delta y$ mogą mieć w jednej regule wiele postaci.

Wniosek z przeprowadzonego rozumowania jest taki, że jeśli w zbiorze reguł wyznaczonych w pierwotnej przestrzeni cech występują reguły, dla których różnica $y - f(\mathbf{x})$ jest mała, to reguły te należy uznać za dobre. Konsekwencją przyjęcia takiego rozwiązania będzie usunięcie ze zbioru treningowego przekazywanego do CK wszystkich przykładów rozpoznających reguły dobre w wejściowej przestrzeni cech.

Przykład 2
Zadaniem jest predykcja zmiennej y na podstawie opóźnień rejestrowanych wartości y_{-1}, u_{-3} (Box i in., 1994). W rozważanym przypadku algorytm M5 wygenerował jedną regułę dla całego zbioru treningowego, $y = 0.76y_{-1} - 0.87u_{-3} + 12.8$. Po zastsowaniu CK uzyskano również jedną regułę $y = 0.73y_{-1} - 0.95u_{-3} + 14.41 + 0.8\Delta y$. Stosując odpowiednie podstawienia uzyskano $y = 0.73y_{-1} - 0.95u_{-3} + 14.41 + 0.8(y_{-1} - (0.76y_{-2} - 0.87u_{-6} + 12.8))$ i, po uproszczeniu, $y = 1.53y_{-1} - 0.608y_{-2} - 0.95u_{-3} + 0.696u_{-6} + 4.17$. Jak widać zastosowanie CK wprowadziło do modelu opróżnienia y_{-2} oraz u_{-6}.

4. Sieć rozmyto-neuronowa na podstawie reguł z liniowymi konkluzjami

Zbiory $supp(r)$ związane z regułami utworzonymi przez M5 potraktować można jako grupy, przy czym podobieństwo obiektów należących do danej grupy polega na tym, że przy spełnieniu warunków występujących w przesłance reguły, minimalizują one wartość wyrażenia (2).

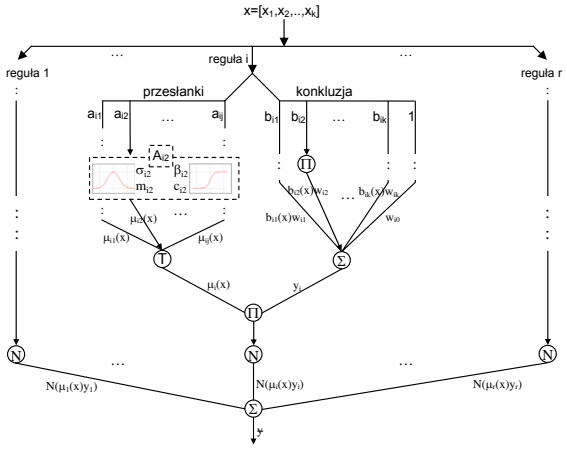
Grupowanie danych (Jain i in.,1988) stanowi podstawę do identyfikacji architektur wielu sieci rozmyto-neuronowych (Czogała, Łęski, 2000; Oh, Pedrycz, 2004). We wszystkich przypadkach zdolności uogólniania utworzonej sieci zależą od liczby zdefiniowanych grup oraz sposobu podziału dziedzin zmiennych na zbiory rozmyte. Do ustalenia liczby grup wykorzystywane są zazwyczaj współczynniki jakości grupowania (Jain i in., 1988, Czogała, Łęski 2000), ale ostatecznie liczba grup definiująca architekturę sieci rozmyto-neuronowej ustalana jest zazwyczaj drogą eksperymentalną.

Wykorzystując doświadczenia w konstrukcji rozmytych sieci neuronowych, można zbiory $supp(r)$, generowane przez reguły wyznaczone za pomocą algorytmu M5, potraktować jako grupy obiektów podobnych, a deskryptory występujące we wszystkich regułach przekształcić w zbiory rozmyte.

Przedstawione w dalszej części niniejszego rozdziału propozycje mają na celu:

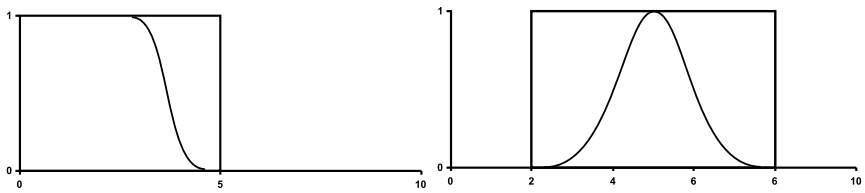
zaproponowanie schematu rozmytej sieci neuronowej FNM5, której architektura, funkcje przynależności przesłanek reguł oraz parametry ich konkluzji zdeterminowane są przez algorytm M5; uzyskanie, bez uczenia, takiego samego jak M5 błędu predykcji sieci FNM5; uzyskanie błędu predykcji na zbiorze testowym zbliżonego do błędu M5 z włączoną opcją wygładzania.

Rysunek pierwszy zawiera schemat sieci FNM5 spełniającej pierwsze dwa z przedstawionych warunków. Spełnianie trzeciego warunku sprawdzone może być jedynie eksperymentalnie ze względu na indukcyjny charakter problemu uczenia się sieci. Na schemacie przedstawiono przepływ danych przez każdą z reguł tworzących sieć (na przykładzie i-tej reguły) oraz ostateczną odpowiedź sieci uwzględnia-jącą wyniki wypracowane przez każdą z reguł.



Rysunek 1. Schemat sieci FNM5.

W utworzonej sieci, ostre postaci deskryptorów zostały zastąpione zbiorami rozmytymi. Jeśli deskryptor jest postaci $a \in (q_1, q_2)$, to przynależność do przedziału zastępujemy gaussowską funkcją przynależności $a(x) = e^{-(\frac{x-m}{\sigma})^2}$, przy czym $m = q_1 + (q_2 - q_1)/2$ oraz σ jest dobierana tak, aby wartości funkcji przynależności w punktach q_2 i q_1 były bliskie zera (zazwyczaj $(q_2 - q_1)/5$) (rys. 2)). Jeśli deskryptor jest postaci $a > q$ (lub $a < q$), to przynależność do przedziału $(q, \pm\infty)$, zastępowana jest sigmoidalną funkcją przynależności $a(x) = \frac{1}{1+e^{-\alpha\beta(x-c)}}$, przy czym $\alpha = \pm 1$, a β jest parametrem definiowanym przez użytkownika decydującym o prędkości narastania (zmniejszania się) wartości funkcji przynależności. Dla ustalonych α, β, parametr c dobierany jest tak, aby funkcja przynależności przyjmowała wartość bliską zero ($\alpha = 1$) lub bliską jeden ($\alpha = -1$) w punkcie q. W opisanych eksperymentach słowo *bliskie* rozumiane jest jako dokładność co najmniej do jednego procenta w stosunku do danych oryginalnych. Ostatecznie, deskryptor $a \in (q_1, q_2)$ zastępowany jest deskryptorem rozmytym (a *is* A), gdzie



Rysunek 2. Zamiana deskryptorów ostrych na zbiory rozmyte.

A jest zbiorem rozmytym o funkcji przynależności zdefiniowanej w taki sposób, jak przedstawiono wcześniej.

Wektor danych wejściowych $\mathbf{x}$ przekazywany jest do każdej z reguł, następnie ustalane są poziomy zapłonu każdego deskryptora występującego w i-tej regule $\mu_{i.}(\mathbf{x})$ i poziom zapłonu całej reguły $\mu_i(\mathbf{x})$, utożsamiany z operacją T-normy. Można zauważyć, że w konkluzjach reguł sieci FNM5 występują wszystkie zmienne niezależne. W przypadku zmiennych rzeczywiście występujących w konkluzji i-tej reguły, współczynniki $w_{i.}$ przyjmują takie wartości, jak występują w regułach wyznaczonych przez M5, w pozostałych przypadkach wartości $w_{i.}$ są równe zero. Z konkluzji reguły uzyskujemy odpowiedź y_i, wartość ta mnożona jest przez stopień aktywacji reguły, uzyskujemy zatem wartość $\mu_i(x)y_i$. Następnie dokonujemy norma-lizacji odpowiedzi i-tej reguły $N(\mu_i(\mathbf{x})y_i) = \frac{\mu_i(\mathbf{x})y_i}{\sum_{i=1}^{r}\mu_i(\mathbf{x})}$. Ostateczną odpowiedzią sieci jest suma znormalizowanych odpowiedzi każdej z reguł o dodatnim stopniu aktywacji.

Po zamianie każdej z reguł z postaci ostrej na rozmytą uzyskujemy reguły zgodne z modelem Takagi-Sugeno-Kanga (Yager, Filev, 1994), których przesłanki zbudowane mogą być wokół różnych zmiennych niezależnych, jest to cecha odróżniająca przedstawioną architekturę od dotychczas istniejących sieci. Cechą charakterystyczną przedstawionej architektury sieci jest również to, że początkowy podział dziedziny danej zmiennej niezależnej dokonuje się w taki sposób, że funkcja przynależności definiująca iloczyn zbiorów rozmytych dzielących tą dziedzinę przyjmuje zawsze wartość zero. Przedstawiona sieć uzyska zatem na zbiorze treningowym identyczne odpowiedzi jak reguły otrzymane algorytmem M5. Wynika to z faktu, że dla każdego przykładu treningowego zapala się zawsze jedna reguła. W sieci FNM5 wynik pochodzący z konkluzji reguły przemnożony zostaje przez poziom aktywacji reguły, który nie musi być równy jeden, ale z uwagi na fakt normalizacji otrzymamy $(\mu_i(\mathbf{x})y_i)/\mu_i(\mathbf{x})$, czyli y_i, a więc odpowiedź identyczną jak reguły przed rozmyciem.

W przedstawionej architekturze sieci, dostrojeniu podlegają parametry funkcji przynależności zbiorów rozmytych znajdujących się w przesłankach (m, σ, c) jak również współczynniki znajdujące się w konkluzjach reguł $(w_{i.})$. Do dostrojenia parametrów sieci po prezentacji każdego przykładu treningowego stosowany jest algorytm wstecznej propagacji błędu, w którym jako funkcję celu wykorzystano tzw. błąd euklidesowy $E = (y - \bar{y})^2/2$, gdzie y jest rzeczywistą wartością zmiennej zależnej, a $\bar{y}$ odpowiedzią sieci. Dowolny z parametrów p zmienia się według

następującej zasady $new(p) = old(p) + \Delta p$, gdzie $\Delta p = \eta(-\frac{\partial E}{\partial p})$. Nowa wartość parametru p modyfikowana jest dodatkowo o wartość tzw. współczynnika momentum $m(p)$ wzmacniającego efekt uczenia $m(p) = \varepsilon(new(p) - old(p))$, gdzie $\varepsilon \simeq 0.95$. We wzorze na Δp występuje tzw. współczynnik uczenia, którego wartość mocno determinuje zbieżność i szybkość uczenia. W sieci FNM5 wartość współczynnika uczenia zmniejsza się zgodnie ze wzorem $\eta = \eta_0 \sqrt[-0.02]{e}$, gdzie η_0 jest początkową wartością współczynnika uczenia, a e jest numerem epoki. Przykłady biorące udział w procesie uczenia sieci pochodzą z tzw. zbioru strojącego, który jest rozłączny ze zbiorem testowym.

5. Eksperymenty z danymi

Do eksperymentów wykorzystano trzy zbiory danych syntetycznych i dwa zbiory pochodzące z systemów monitorowania stosowanych w przemyśle wydobywczym:

- synth1 - to zbiór generowany przez funkcję postaci $y = 3x^5 - x^4 - x$, analizowano dane z przedziału $[-1, 1]$,
- synth2 - to zbiór generowany przez funkcję postaci $y = (1 + x1^{0.5} + x2^{-1} + x3^{-1.5})^2$, analizowano dane z przedziału $(0, 20]$,
- gas furnace - to zbiór benchmarkowy opisany przez Boxa (1994), zmienna zależna y oznaczająca stężenie dwutlenku węgla na wyjściu z pieca gazowego zależy od wartości y_{-1} oraz u_{-3} (u - tempo dopływu metanu do pieca),
- Ec - to zbiór danych opisujący proces urabiania skał stożkowymi nożami obrotowymi, zmienna zależna to jednostkowa energia skrawania Ec, zmiennymi niezależnymi są: parametry technologiczne pracy ostrza oraz parametry geometryczne ostrza,
- CO2 - to zbiór opisujący stężenie dwutlenku węgla w stacji odwadniania kopalń; zmiennymi niezależnymi są m.in.: stężenie dwutlenku węgla w stacji, ciśnienie atmosferyczne, wilgotność otoczenia, temperatura; wartością przewidywaną jest CO2pred oznaczającą stężenie dwutlenku węgla z 6-minutowym wyprzedzeniem.

Tablica 1. Wyniki eksperymentów

Zbiór danych	M5	NFM5	ANNBFIS	CKM5
synth1	0.11 0.16	0.09 0.14	–	0.6 0.14
synth2	1.13 439	1.07 437	4.42 439	0.51 445
gas furnance	0.38 0.40	0.38 0.40	0.33 0.41	0.26 0.29
Ec	3.89 3.78	3.89 3.79	3.93 3.80	0.78 4.45
CO2	0.28 0.26	0.28 0.26	0.40 0.34	0.12 0.16

W tabeli pierwszej zaprezentowano wyniki przeprowadzonych eksperymentów, podano wyniki uzyskane na zbiorze treningowym oraz na zbiorze testowym. Efektywność utworzonych modeli porównywano obliczając błąd RMS (pierwiastek z błędu średniokwadratowego). Znaczenie kolumn jest następujące: M5 - wyniki uzyskane algorytmem M5+wygładzanie, NFM5 - wyniki sieci FNM5, ANNBFIS

- wyniki sieci zaproponowanej przez Czogałę i Łęskiego (2000), CKM5 - wyniki uzyskane po zastosowaniu CK.

We wszystkich eksperymentach jako operator ustalający stopień aktywacji reguły rozmytej wykorzystano T-normę Zadeha (minimum). Sieć poddawano uczeniu przez maksymalnie 70 epok (dla zbioru synth1 200 epok). Zbiory danych synth1 i synth2 składały się z 40 obiektów, zbiór gas furnace zawiera 293 obiekty, Ec zawiera 582 obiekty, CO2 zawiera 5370 obiektów. Podobnie jak w pracach Czogały i Łęskiego (2000) oraz Oha i Pedrycza (2004) zbiory treningowe zawierały od 40 do 70 procent losowo wybranych obiektów. Liczba reguł utworzona przez algorytm M5, wynosiła: synth1 - 3 reguły, synth2 - 3 reguły, gas furnace - 1 reguła, Ec - 5 reguł, CO2 - 10 reguł.

W przypadku sieci ANNBFIS testowano różne rozwiązania (od 2 do 5 grup, do 200 epok uczenia), w tablicy zamieszczono najlepsze z uzyskanych wyników.

Czas działania poszczególnych algorytmów to kolejny argument przemawiający za zastosowaniem algorytmu M5 i sieci FNM5. We wszystkich (również w przypadku CO2) zbiorach danych czas uczenia reguł trwał poniżej jednej sekundy, czas uczenia sieci FNM5 dla wszystkich zbiorów poza Ec i Co2 wynosił poniżej trzech sekund (Ec - 9 s., CO2 - 26 s.). Przykładowo, czas uczenia sieci ANNBFIS (implementacja w środowisku Matlab - 70 epok) dla zbiorów gas furnace i CO2 wynosił odpowiednio 20 s. (3 reguły) i 35 s. (5 reguł). Dla 10 reguł czas uczenia wynosił dla zbioru CO2, 3.5 minuty.

6. Analiza wyników i wnioski

Zaproponowany w artykule schemat wykorzystania CK pozwolił na zmniejszenie błędu popełnianego w czasie predykcji. Działo się tak we wszystkich przypadkach poza zbiorem Ec, w którym błąd na zbiorze testowym zwiększył się. Wytłumaczenie tego faktu jest stosukowo proste, można zauważyć, że zastosowanie CK doprowadziło do efektu przeuczenia, uzyskany model niemal idealnie dopasował się do danych treningowych i strojących, tracąc zdolności uogólniania. Duże wartość błędu popełniane na testowym zbiorze synth2 wynikają z postaci funkcji syntetycznej (gwałtowny spadek wartości funkcji w przedziale (0,2)) oraz rozkładu przykładów w zbiorach treningowym strojącym i testowym. Poprawa zdolności predykcyjnych odbywała się kosztem czytelności modelu, gdyż w pewnych regułach pojawiła się nowa zmienna (metazmienna) Δy.

Dla zbioru treningowego gas furnace sieć FNM5 uzyskała gorsze wnyiki niż ANNBFIS, przyczyną takiego stanu rzeczy jest fakt, że algorytm M5 wygenerował tylko jedną regułę (bez przesłanek), zatem sieć FNM5 nie mogła uzyskać innych wyników, gdyż jakiekolwiek dostrajanie było niemożliwe.

Poza cehcami charakterystycznymi sieci FNM5 przedstawionymi w rozdziale czwartym, sieć FNM5 posiada naszym zdaniem co najmniej trzy unikalne cechy:

- architektura i parametry sieci ustalane są automatycznie w oparciu o zbiór reguł otrzymanych przez szybki algorytm M5,
- błąd predykcji po niewielkiej liczbie epok jest podobny (a czasem mniejszy - synth1, synth2) do tego, jaki uzyskujemy za pomocą algorytmu M5 z wł-

czoną opcją wygładzania; reguły zawarte w sieci, nawet po rozmyciu, dają lepszą (niż M5+wygładzanie) możliwość interpretacji uzyskanych wyników,

- łączny czas działania algorytmu M5 i trenowania sieci FNM5 są krótkie w porównaniu do innych metod.

Analiza danych Ec i CO2 była niejako główną motywacją wyeliminowania procedury smoothingu z algorytmu M5, gdyż końcowym użytkownikom zależało nie tylko na dokładności predykcji, ale także na czytelności modelu tak, aby porównać go ze swoją wiedzą ekspercką (Sikora i in. 2005).

Dalsze prace koncentrują się na zmianie algorytmu uczenia sieci FNM5, celem tych prac jest próba polepszenia dokładność predykcji. W czasie uczenia modyfkowane będą również parametry l i β, a ich początkowe wartości ustalone zostaną za pomocą algorytmu genetycznego w sposób podobny do proponowanego przez Oha i Czogałę (2004). Każda grupa z modyfikowanych parame-trów będzie posiadała własny współczynnik uczenia. Wykonana zostanie także większa liczba eksperymentów wraz z badaniem statystycznej istotności różnic uzyskanych wyników. Inny kierunek badań to adaptacja na potrzeby FNM5 algorytmów indukcji rozmytych drzew decyzyjnych (Janikov 1996).

Literatura

Box G.E. and Jenkins G.M. (1994) *Time series analysis: forecasting and control.* Prentice Hall, New Jersey.

Breiman L. et al (1994) *Classification and Regression Trees.* Wadsworth, Belmont CA.

Czogała E. and Łęski J. (2000) Fuzzy and Neuro-Fuzzy Intelligent Systems *Studies in Fuzziness and Soft Computing* **47**, Springer-Verlag Company.

Jain A. and Dubes R.C. (1988) *Algorithms for clustering data.* Prentice-Hall, NY.

Janikov C. (1996) *Fuzzy decision trees : issues and methods.* Research report, Dept. of Math. and CS., University of Missouri.

Oh S.K., Pedrycz W. and Park H.S. (2004) Rules based multi-FNN identification with the aid of evolutionary fuzzy granulation *Knowledge-Based Systems* **17**, 1-13.

Rutkowski L. and Cpałka K. (2003) Flexible neuro-fuzzy systems *IEEE Transactions on Neural Network* **14**, 554-574.

Quinlan R. (1992) *C4.5 Programs for Machine Learning.* Morgan Kaufman Publishers, California.

Quinlan R. (1993) Combining instance-based learning and model-based learning. *Proc of the Tenth International Conference on Machine Learning.*

Sikora M. and Kozielski M. (2006) Hybrid data exploration methods to prediction tasks solving *Archives of Theoretical and Applied Informatics* **18**, 57-73.

Sikora M. and Krzykawski D. (2005) Zastosowanie metod eksploracji danych do analizy wydzielania się CO2 w pomieszczeniach stacji odwadniania kopalń. *Mechanizacja i Automatyzacja Górnictwa* **6/413**, 57-67.

Shavlik J.W. and Opitz D.W (1997) Actively searching for a effective neural-network ensamble. *Connection Science* **8**, 337-353.

Solomatine D. and Siek M.B (2004) Flexible and optimal M5 model trees with applications to flow predictions. *Proc of the Sixth International Conference on Hydroinformatics.*

Stefanowski J. (2001) Algorytmy indukcji reguł decyzyjnych w odkrywaniu wiedzy. *Praca habilitacyjna.* Politechnika Poznańska, Poznań.

Tadeusiewicz R. (1993) *Sieci neuronowe.* Akademicka oficyna wydawanicza RM, Warszawa.

Wang L.X. (1994) *Adaptive Fuzzy Systems and Control. Design and Stability Analysis.* Prentice Hall, N.Y.

Wnek J. and Michalski R.S. (1994) Hypothesis-driven constructive induction in AQ17-HCI: A Method and Experiments *Machine Learning* **14**, 139-168.

Yager R.R. and Filev D.P. (1994) *Essential of Fuzzy Modelling and Control.* John Wiley and Sons, N.Y.

Constructive induction and fuzzification for rules with linear conclusions quality improvement

Methods of quality improvement of rules with linear conclusions determined by means of the M5 algorithm proposed by Quinlan are presented in the paper. The M5 algorithm makes possible to get a rule prediction model in very short time. Application of smoothing procedure in the M5 algorithm considerably increases prediction abilities at the cost of meaningful decrease possibilities of model interpretation. A fuzzy-neural network architecture that allows to get results close to the M5 algorithm ones, with smoothing option switch on, is proposed in the paper. The second of proposed me-thods is application of constructive induction in the aim of the M5 algorithm prediction abilities improvement.