Sieci Neuronowe i Uczenie Maszynowe: próba integracji

Włodzisław Duch & Karol Grudziński
Katedra Metod Komputerowych, Uniwersytet Mikołaja Kopernika, ul. Grudziądzka 5, 87-100 Toruń
E-mail:{duch, kagru}@phys.uni.torun.pl

Abstrakt

Metody oparte na ocenie podobieństwa klasyfikowanych przypadków do przypadków wzorcowych dają często doskonałe rezultaty i mają liczne zalety. Ogólny schemat działania metod opartych na podobieństwie oparty jest na parametryzacji prawdopodobieństwa klasyfikacji i uwzględnia wybór wektorów referencyjnych, optymalizację ich położenia i skalowanie ich wpływu, wybór funkcji oceniającej podobieństwo, funkcji kosztu, sposobu uśredniania wyników lokalnych modeli, selekcję cech i wiele innych procedur. Zarówno sieci neuronowe typu radialnych funkcji bazowych jak i typowe perceptrony wielowarstwowe stanowią szczególne przypadki tego schematu. Schemat ten pozwala również na tworzenie nowych metod klasyfikacji i aproksymacji, miedzy innymi metod posiadających naturalną realizację w postaci sieci neuropodobnych.

1. Wprowadzenie

Klasyfikacja jest jednym z głównych zastosowań systemów inteligentnych. Można do niej sprowadzić wiele innych zastosowań, takich jak rozpoznawanie obiektów, diagnozowanie, auto i hetero-asocjację a nawet aproksymację (w granicy nieskończonej liczby klas). Badania empiryczne wielu klasyfikatorów (statystycznych, indukcji reguł, drzew decyzji, sieci neuronowych, minimalnoodległościowych itd.) w zastosowaniu do licznych baz danych, wykonane np. w ramach projektu STATLOG (Michie i inni 1994) lub w pracy Rhowera i Morcinca (1996), pokazują, że nie ma jednego, uniwersalnego klasyfikatora najlepszego dla wszystkich danych – dla każdego klasyfikatora można znaleźć zbiór danych, dla którego rezultaty będą doskonałe, jak i taki, dla którego wyniki będą słabe. Wskazuje to na konieczność opracowania metodologii oceny systemów klasyfikujących, gdyż często rezultaty podawane są dla jednego, wybranego zbioru danych, dla którego uzyskano dobre wyniki. Często metody oparte na podobieństwie, do których należy metoda najbliższych sąsiadów (k-NN) działają lepiej od innych. Np. w projekcie STATLOG (Michie i inni 1994) w jednej trzeciej przypadków metoda k-NN daje najlepsze lub prawie najlepsze rezultaty i to dla k=1 bez optymalizacji k. Algorytm k-NN jest zatem cenną metodą i ze względu na swoja prostotę powinien być używany jako metoda referencyjna dla innych klasyfikatorów. Tymczasem trudno jest znaleźć dobry program realizujący tą metodę, nie brakuje natomiast darmowych i komercyjnych symulatorów neuronowych.

Metoda najbliższych sąsiadów ma liczne zastosowania w rozpoznawaniu wzorców (pattern recognition) i budowie systemów ekspertowych opartych na prototypach (case-based reasoning expert systems). Algorytm k-NN jest tak popularny, że w sztucznej inteligencji podejście

oparte na tym modelu nazywane jest *instance-based reasoning* (prototypy w bazie danych mają postać wektorów liczb, oblicza się odległości od wszystkich wektorów – metoda brute-force), *memory-based reasoning* (w tym podejściu kładzie się nacisk na zorganizowanie bazy treningowej w pamięci w postaci drzewa i na szukanie obiektów w pamięci w celu przyspieszenia działania systemu, posługując się równoległymi technikami programowania) oraz *case-based reasoning* (prototypy mają postać skomplikowanych struktur danych – najczęściej obiektów i są przechowywane w wyspecjalizowanych bazach danych, wykorzystuje się wiedzę ogólną o naturze problemu - domain knowledge - oraz w bardziej złożonych systemach słabo jeszcze rozwiniętą technikę adaptacji prototypu najbardziej podobnego do przypadku nieznanego).

Metody oparte na podobieństwie (Similarity-Based Methods, SBM) rozwijały się dotychczas niezależnie od sieci neuronowych i innych metod przydatnych do klasyfikacji. W znacznej mierze wynika to z odmiennych źródeł, z jakich wywodzą się te metody, jednakże próba zrozumienia związków pomiędzy nimi i integracji różnych metod uczenia się z danych wydaje się być bardzo obiecująca. Ponieważ nie udało się dotychczas znaleźć takiej charakterystyki danych, która pozwoliłaby na określenie, jakie metody będą się najlepiej nadawać do ich klasyfikacji, należy zaproponować inne podejście, gwarantujące wyniki, które powinny być nierozróżnialne (przy statystycznej ocenie dokładności) od najlepszych możliwych do osiągnięcia w danym przypadku wyników. Proponowana tu metodologia polega na użyciu schematu metod SBM o wzrastającym stopniu komplikacji. W przestrzeni wszystkich metod, jakie można wygenerować w ramach danego schematu, należy poszukiwać najprostszego rozszerzenia aktualnie używanej metody, które znacząco poprawia wyniki. Startując np. z metody najbliższego sąsiada można dokonać optymalizacji liczby sąsiadów lub używanej funkcji odległości. Takie postępowanie równoważne jest gradientowej metodzie przeszukiwania, czyli metodzie „pierwszy najlepszy". Jeszcze lepsze rezultaty dać może bardziej kosztowne obliczeniowo przeszukiwanie wiązką.

Optymalizacja jakościowo różnych parametrów adaptacyjnych powstającej w efekcie końcowym metody daje gwarancję dobrania odpowiedniej metody dla analizowanych danych. Pomimo tego, że sieci MLP z funkcjami sigmoidalnymi są uniwersalnymi aproksymatorami złożoność sieci, liczba parametrów adaptacyjnych koniecznych do rozwiązania zadania, a tym samym i czas uczenia, może być nawet w prostych przypadkach niepotrzebnie duża. Na przykład wydzielenie jednego rozkładu o gaussowskim charakterze z tła wymaga jednego prototypu z odpowiednią funkcją odległości, lub pojedynczej funkcji Gaussa w sieciach z radialnymi funkcjami bazowymi, czyli około $2N$ parametrów w N-wymiarowej przestrzeni danych. Natomiast najprostsza sieć MLP, która rozwiąże to zadanie musi mieć przynajmniej $N+1$ neuronów, czyli około $(N+1)^2$ parametrów. Wskazuje to na ważną rolę, jaką grać może wybór odpowiednich funkcji transferu (Duch i Jankowski 1999).

W następnym rozdziale przedstawimy ogólny schemat metod opartych na podobieństwie, omawiając możliwości parametryzacji i procedury optymalizacji parametrów różnego typu, w tym zupełnie nową metodę typu „konsylium ekspertów", łączącą wyniki wielu modeli ze sobą. Następnie przedstawimy kilka przykładów znanych i nowych metod klasyfikacji, będących szczególnymi przypadkami schematu opartego na podobieństwie. Zaproponujemy również naturalny sposób integracji metod typu SBM i sieci neuronowych, oraz sieciowe realizacje niektórych algorytmów SBM. Chociaż dotychczas przetestowana została jedynie niewielka część wszystkich możliwości rezultaty obliczeń na prawdziwych i sztucznych bazach danych, przedstawione w kolejnym rozdziale, są bardzo zachęcające.

2. Schemat metod opartych na podobieństwie

Zagadnienie klasyfikacji można sformułować w następujący sposób: mając dany zbiór wektorów treningowych $\{\mathbf{X}^p, \mathbf{C}(\mathbf{X}^p)\}$, $p=1 .. n$, gdzie $\mathbf{C}(\mathbf{X}^p)$ jest etykietą klasy wektora $\mathbf{X}^p$ i wektor $\mathbf{X}$ należący do nieznanej klasy, należy określić prawdopodobieństwo $p(\mathbf{C}_i,|\mathbf{X};M)$ zakwalifikowania wektora $\mathbf{X}$ do zadanych klas $\mathbf{C}_i$, gdzie M jest modelem klasyfikacyjnym, opisującym wszystkie parametry i wykonywane procedury. Nie jest to najbardziej ogólne sformułowanie, gdyż zakłada jednakową długość wektorów $\mathbf{X}$. Ogólny model systemu adaptacyjnego używanego dla potrzeb klasyfikacji może zawierać niektóre lub wszystkie z parametrów danych poniżej:

$$M = \left\{k, d(\cdot; r, \rho), G(d(\cdot)), \{\mathbf{R}^p\}, E[\cdot], K(\cdot)\right\} \tag{1}$$

gdzie:

- k jest liczbą uwzględnianych wektorów referencyjnych w otoczeniu wektora $\mathbf{X}$;
- $d(\cdot;\ r,\rho)$ jest funkcją odległości (podobieństwa), r jest maksymalnym promieniem sfery obejmującej otoczenie wektora $\mathbf{X}$, ρ to pozostałe parametry adaptacyjne.
- $G(d(\mathbf{X}, \mathbf{R}^p))$ jest funkcją ważącą wpływ wektorów referencyjnych $\mathbf{R}^p$ na prawdopodobieństwo klasyfikacji;
- $\{\mathbf{R}^p\}$ jest zbiorem wektorów referencyjnych, wygenerowanym ze zbioru wektorów treningowych $\{\mathbf{X}^p\}$;
- $E[\cdot]$ jest całkowitą funkcją kosztu minimalizowaną podczas uczenia;
- $K[\cdot]$ jest funkcją skalującą wpływ błędu dla danego prototypu z bazy treningowej na całkowitą funkcję kosztu.

Oprócz wymienionych powyżej funkcji i parametrów metody klasyfikacji mogą różnić się procedurami selekcji cech, sposobem traktowania wartości brakujących, algorytmami minimalizacji, używanymi w celu znalezienia optymalnych parametrów oraz sposobami realizacji w postaci sieci neuropodobnych. Można też zbudować więcej niż jeden model i używać różnych procedur łączenia wyników. Omówimy teraz dokładniej wszystkie te możliwości.

Wybór k i prawdopodobieństwa aprioryczne

W najprostszym przypadku prawdopodobieństwo klasyfikacji $p(\mathbf{C}_i,|\mathbf{X};\ M)$ jest sparametryzowane przez $p(\mathbf{C}_i,|\mathbf{X};\ k,\ d(\cdot),\{\mathbf{X}^p\})$, co prowadzi do metody k najbliższych sąsiadów, k-NN, w której cały zbiór treningowy jest użyty jako zbiór referencyjny. Ustalając jakąś metrykę $d(\cdot)$, np. metrykę Euklidesową, liczba najbliższych sąsiadów k jest jedynym parametrem podlegającym optymalizacji. Zwykle jest ona stosunkowo mała i zwiększając k dokonujemy wyboru patrząc na liczbę błędów klasyfikacji otrzymywanych dla kolejnych k. W metodzie k-NN koszt obliczeniowy oceny dokładności klasyfikacji za pomocą procedury testowej, określającej dla wszystkich n wektorów zbioru treningowego klasę danego wektora, biorąc za zbiór referencyjny $\{\mathbf{X}^p\}$ pozostałe n-1 wektorów (test *leave-one-out*), równy jest kosztowi jednokrotnej klasyfikacji wszystkich n wektorów. Ten niewielki koszt umożliwia dobry wybór k z punktu widzenia najlepszej generalizacji metody. Prawdopodobieństwo klasyfikacji $p(\mathbf{C}_i|\mathbf{X};k) = k_i /k$ równe jest liczbie wektorów referencyjnych z klasy $\mathbf{C}_i$ dzielonej przez liczbę wszystkich wektorów uwzględnianych w ocenie prawdopodobieństwa.

Pewne subtelności dotyczą sposobu obliczania błędu klasyfikacji dla przypadków, w których wszystkie prawdopodobieństwa $p(\mathbf{C}_i|\mathbf{X};k)$ są jednakowe. Często zwiększa się w takich przy-

padkach *k* by wybrać jedną z klas. Załóżmy, że mamy *C* klas i uporządkujmy je tak, by ostatnią z nich była klasa większościowa. Interesującą możliwością, zapewniającą przy okazji lepszą zgodność wyników klasyfikacji z prawdopodobieństwami *aprioryczymi*, jest wprowadzenie dodatkowych parametrów κ_i dla pierwszych *C*-1 klas:

$$p_i = \kappa_i p(C_i \mid \mathbf{X}; M), \;\; p_C = 1 - \sum_{i=1}^{C-1} p_i \;, \;\; \kappa_i \in [0,1] \qquad (2)$$

Jeśli κ_i =0 to otrzymamy zawsze p_c =1 dla klasy większościowej, a wiec wyniki nie mogą być gorsze od wyników klasyfikatora większościowego; dla κ_i =1 otrzymamy poprzednie wyniki, $p(C_i|\mathbf{X};M)$. Optymalizując parametry κ_i (dla dwóch klas jest to jednoparametrowa optymalizacja) możemy jedynie polepszyć otrzymane wyniki.

Metoda *k*-NN ma liczne zalety:

1) Liczba klas nie jest ograniczona.
2) System może pracować jako pamięć heteroasocjacyjna, wydzielając dowolną podprzestrzeń znanych cech i przewidując pozostałe elementy wektora, a nie tylko klasę, chociaż dla każdej podprzestrzeni należy niezależnie zoptymalizować *k*.
3) Łatwo jest dodać interpolację liniową, pozwalającą na lokalne uśrednienie wartości i zastosowanie tej metody do aproksymacji.
4) Stabilność: jak pokazał Breiman (1996) sieci neuronowe, drzewa decyzji i wiele innych klasyfikatorów to systemy niestabilne – niewielka zmiana zbioru uczącego może spowodować drastyczną zmianę struktury klasyfikatora i realizowanych przez niego granic decyzji. Metoda najbliższych sąsiadów jest natomiast metodą stabilną (Breiman 1998), gdyż lokalne zmiany danych zmieniają granice decyzji tylko w niewielkim stopniu.

Parametryzacja miary podobieństwa

Większość stosowanych miar podobieństwa wywodzi się z miar odległości które, przekształcone za pomocą funkcji ważącej $G(d(\mathbf{X}, \mathbf{R}^p))$, pozwalają ocenić podobieństwo. Wiele miar podobieństwa opisanych zostało w pracach (Duch i Jankowski 1999) oraz (Wilson i Martinez 1997). W metodach minimalnoodległościowych najczęściej stosuje się metrykę Euklidesową dla danych ciągłych i miarę Hamminga dla cech binarnych. Dodatkowe parametry, które mogą podlegać optymalizacji są albo globalne (takie same w całej przestrzeni) albo lokalne (różne dla każdego wektora referencyjnego). Metryka Minkowskiego posiada jeden globalny parametr adaptacyjny α. Uwzględniając czynniki skalujące (wagi) dla tej miary otrzymujemy:

$$d(\mathbf{A},\mathbf{B};\mathbf{s})^{\alpha} = \sum_{i=1}^{N} s_i d(A_i, B_i)^{\alpha} \qquad (3)$$

gdzie za $d(A_i,B_i)$ przyjmuje się zwykle różnicę $|A_i-B_i|$. Dla α = 2 otrzymujemy metrykę Euklidesową a dla α = 1 miarę Manhattan. Ponieważ kształt granic decyzji jest odmienny dla różnych wartości wykładnika α warto go więc zoptymalizować.

W rozwijanym przez nas systemie zaimplementowaliśmy dwie grupy metod, które automatycznie dobierają czynniki skalujące s_i. Do pierwszej grupy należą metody oparte na minimalizacji funkcji kosztu, do drugiej zaś metody oparte na szukaniu typu „najpierw najlepszy" zdyskretyzowanych wartości czynników skalujących. Szczególnym przypadkiem takiego szukania jest uwzględnienie tylko binarnych wartości czynników skalujących – odpowiada to selekcji cech, omówionej poniżej.

Interesującą funkcję odległości (Duch i Jankowski 1999) otrzymamy przez kombinację kilku funkcji sigmoidalnych (Rys. 1) daną wzorem:

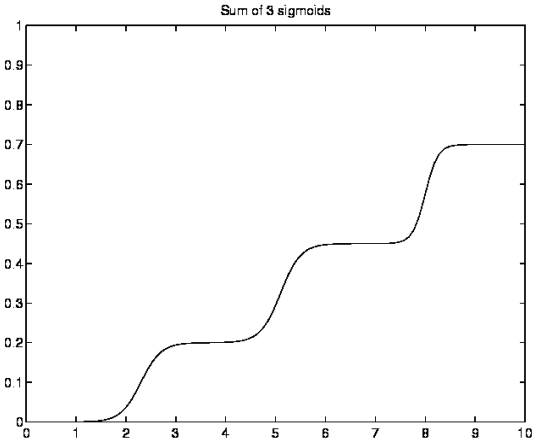
$$d(A_i, B_i) = \sum_j \alpha_{ij} \sigma\left(\beta_{ij} |A_i - B_i| - \gamma_{ij}\right); \;\; d(A,B)^\alpha = \sum_{i=1}^{N} d(A_i, B_i)^\alpha \qquad (4)$$

Parametry α, β, γ pozwalają na minimalizację odległości wewnątrz klas i maksymalizację odległości pomiędzy klasami. Granice decyzji przy użyciu tak sparametryzowanych funkcji mogą stać się prostokątne, pozwalając na interpretację działania klasyfikatora za pomocą reguł logiki klasycznej.

Ważną kategorią funkcji odległości są miary probabilistyczne. Można je zastosować dla danych o cechach symbolicznych bez zamiany tych cech na wartości numeryczne. Miara znana jako VDM (*Value Difference Metric*) dana jest przez:

$$D_V^q(A,B) = \sum_{j=1}^{N} \sum_{i=1}^{C} \left| p\left(C_i | A_j\right) - p\left(C_i | B_j\right) \right|^q, \;\; p\left(C_i | A_j\right) = N(C_i | A_j) / N(A_j) \qquad (5)$$

gdzie $N(C_i|A_j)$ jest liczbą wystąpień wartości atrybutu A_j w wektorach należących do klasy C_i a $N(A_j)$ jest liczbą wystąpień we wszystkich klasach. Uogólnienie tej miary dla wartości ciągłych możliwe jest po zdefiniowaniu funkcji rozkładu gęstości prawdopodobieństwa. Miarę VDM i jej modyfikacje (Wilson i Martinez 1997) można też połączyć z innymi miarami, zastosowanymi do atrybutów o wartościach numerycznych. W szczególności zamieniając symboliczne wartości A_j na zbiór prawdopodobieństw $p(C_i|A_j)$ można odtworzyć odległości VDM korzystając z wartości numerycznych. Taką metodę można zastosować dla dowolnego klasyfikatora.



Rys. 1. Suma 3 funkcji sigmoidalnych – użyteczna funkcja odległości.

Selekcja i skalowanie cech

Istnieje bardzo wiele metod selekcji cech (por. Hernandez, Fernandez 1999), które mogą znaleźć zastosowanie w metodach opartych na podobieństwie. Teoretycznie rzecz biorąc minimalizacja funkcji kosztu ze względu na czynniki s_i skalujące wkład od poszczególnych cech powinna nadać cechom zbędnym małe wartości. Jedną z możliwości jest ustalenie wartości czynnika skalującego dla najważniejszej cechy na 1 i dodanie do funkcji kosztu członu regularyzującego, np. sumy kwadratów pozostałych czynników skalujących, w celu otrzymania jak największej liczby zerowych s_i. Jeśli w danej metodzie klasyfikacji można użyć minimalizacji gradientowej takie podejście może dać dobre wyniki, w przeciwnym przypadku trzeba

stosować metody szukające globalnego minimum bez wykorzystania gradientu (Duch i Korczak 1999), co jest czasochłonne. Dlatego lepiej jest najpierw przeprowadzić selekcję cech i wystartować z binarnych czynników skalujących s_i.

Poniżej opisane zostały proste metody selekcji cech oparte na wariantach szukania typu „najpierw najlepszy" (Duch i Grudziński, 1999a). Algorytmy selekcji cech powinny odrzucić te cechy, które niewiele wnoszą do oceny podobieństwa. Dobrym testem takich metod jest dodanie nowych cech, zawierających jedynie szum (powinny zostać odrzucone), oraz dodanie cechy będącej liniową funkcją klasy dla danego wektora (powinna zostać jako jedyna, pozwalająca na 100% klasyfikację). Taki test powinien być użyty dla wszystkich metod selekcji cech.

Wstępną ocenę cech, nazywaną dalej rankingiem, dokonać można sprawdzając dokładność klasyfikacji przy użyciu wszystkich cech, a następnie stosując kroswalidację na zbiorze treningowym wyłączać kolejne cechy jedną po drugiej, zapamiętując za każdym razem dokładność klasyfikacji i wariancję. Najważniejszą cechą jest ta, która po eliminacji powoduje największą degradację dokładności klasyfikacji. Następnie startując od jednoelementowych wektorów zawierających cechę uznaną za najważniejszą, sprawdza się dokładność klasyfikacji powiększając liczbę elementów wektorów o kolejne, coraz mniej ważne cechy. Tylko te cechy, które po dodaniu do rosnącej liczby cech powodują wzrost dokładności klasyfikacji, są akceptowane. Podczas dodawania cech można optymalizować parametry klasyfikatora, co jest oczywiście bardziej kosztowne.

Dokładniejszy wybór czynników skalujących s_i dokonać można za pomocą metod opartych na szukaniu, które często są szybsze i znajdują lepsze rozwiązania niż metody oparte na minimalizacji. Rozwinęliśmy trzy takie metody, oznaczane jako S0, S1 i „metoda dostrajania wag". Ta ostatnia startuje z rozwiązań otrzymanych przez metody minimalizacyjne lub metody S0 czy S1 i jest bardzo szybka. Stosując metody szukania należy przyjąć dyskretne wartości czynników skalujących, można to jednak zrobić stopniowo zwiększając ich precyzję.

W metodzie S0 najpierw wybieramy najważniejszą cechę analizując rezultaty otrzymane z kroswalidacji na zbiorze testowym z włączoną tylko jedną cechą. Cechy, dla których osiągnięto najwyższą dokładność klasyfikacji są dobrymi kandydatami do uwzględnienia w pierwszym rzędzie. Ich indeksy są pamiętane w tablicy w kolejności od najlepszej do najgorszej. Dla najważniejszej cechy ustala się s_1=1, definiując tym samym względną skalę odległości. Czynnik skalujący dla następnej cechy wyznaczany jest na podstawie rezultatów otrzymanych przy dwóch cechach włączonych, zmieniając s_2 z krokiem Δ (np. Δ=0.05) tak, że $s_2 = m\Delta \in [0,1]$. Najlepsza wartość s_2 i dokładność klasyfikacji jest zapamiętywana i opisany proces powtarzamy dla kolejnej cechy wybranej zgodnie ze wstępnym rankingiem przy ustalonych wartościach s_1 i s_2. Po zakończeniu uczenia dokładność klasyfikacji sprawdzana jest na zbiorze testowym ze skalowaną funkcją odległości.

Może się oczywiście okazać, że testowanie pojedynczych cech nie da dobrego wyniku, bo dopiero kombinacja kilku cech pozwala na klasyfikację. Jeśli algorytm S0 nie daje dobrych rezultatów należy zastosować nieco bardziej kosztowny obliczeniowo algorytm S1, który startuje od wszystkich cech po kolei je odrzucając. W początkowym rankingu cech wszystkie wagi ustawiane są na s_i =1, i=1..N, gdzie N jest liczbą cech, po czym ranking dokonuje się przez kolejne wyłączanie pojedynczej cechy. W następnym etapie czynnik skalujący najważniejszej cechy ustala się na 1, a optymalny czynnik skalujący dla drugiej cechy znajdowany jest przez poszukiwanie minimum dla dyskretnych wartości s_i, utrzymując wartości pozostałych czynników jako 1. Takie szukanie czynników skalujących powtarza się dla kolejnych cech z rankingu podczas gdy pozostałe cechy mają wagi równe 1. Odmiana tej metody startuje z binarnego wektora $s_i \in \{0, 1\}$ otrzymanego za pomocą jakiegoś algorytmu selekcji cech.

Po wyznaczeniu przybliżonego wektora optymalnych czynników skalujących można je „dostroić" szukając lepszego rozwiązania wokół wektora już znalezionego. Do każdego czynnika skalującego dodawana jest lub odejmowana stała wartość δ zadana z góry przez użytkownika (np. 0.5): $s_i \leftarrow s_i \pm\delta$. Jeśli zmiana wagi prowadzi do polepszenia działania klasyfikatora, czynnik ten zostaje uaktualniony, w przeciwnym przypadku pozostaje bez zmian. Gdy wszystkie wagi zostaną sprawdzone i żadna zmiana o czynnik δ nie poprawia rezultatu powtarza się procedurę z $\delta \leftarrow \delta/2$. Procedura kończy się jeśli różnica w dokładności klasyfikacji w dwóch kolejnych iteracjach jest mniejsza od zadanej wartości.

Zaimplementowano różne warianty opisanych powyżej metod, np. zawężając procedurę szukania tylko do podzbioru cech zidentyfikowanych jako istotne podczas rankingu. Chociaż są to proste metody w praktyce działają równie dobrze jak bardziej wyrafinowane (por. empiryczne oceny algorytmów selekcji cech, Fernandez, Hernandez 1999).

Parametryzacja funkcji ważących wpływ wektorów referencyjnych

Wektory referencyjne, które są najbardziej podobne do przypadku testowego **X**, powinny mieć większy wpływ na prawdopodobieństwo klasyfikacji od tych, które leżą dalej. Uwzględnianie dokładnie *k* sąsiadów odpowiada funkcjom transferu typu „twarda sfera". Zamiana tych funkcji na bardziej gładkie pozwala na uwzględnienie takiego skalowania. Na przykład weźmy stożkową radialną funkcję przynależności znaną z logiki rozmytej: przyjmuje ona wartość zero na zewnątrz promienia *r* oraz rośnie liniowo do wartości 1 wewnątrz promienia. Prawdopodobieństwo klasyfikacji jest obliczane przez neurony wyjściowe korzystając ze wzoru:

$$p(\mathbf{C}_i \mid \mathbf{X}; r) = \frac{\sum_{p\in\mathbf{C}_i} G\left(\mathbf{X};\mathbf{R}^p, r\right)}{\sum_p G\left(\mathbf{X};\mathbf{R}^p, r\right)}$$

$$G\left(\mathbf{X};\mathbf{R}, r\right) = \max\left(0, 1-\frac{d\left(\mathbf{X};\mathbf{R}\right)}{r}\right) \qquad (6)$$

$G(\mathbf{X}; \mathbf{R}, r)$ gra tu rolę wagi skalującej wpływ na prawdopodobieństwo klasyfikacji wektorów referencyjnych umieszczonych w odległości $d(\mathbf{X};\mathbf{R})$. W przypadku algorytmu *r*-NN parametr *r* jest optymalizowany. Sieci typu RBF o gaussowskich funkcjach transferu stanowią szczególny przypadek takiej skalowanej metody SBM. Inną użyteczną funkcją skalującą jest kombinacja dwóch funkcji sigmoidalnych:

$$G(d) = \sigma\left(d\left(\mathbf{X}-\mathbf{R}^n\right)-r\right)-\sigma\left(d\left(\mathbf{X}-\mathbf{R}^n\right)+r\right) \qquad (7)$$

Jako funkcji skalującej wektory wewnątrz promienia *r* można użyć zmiennego *r* równego odległości do *k*-tego sąsiada. Jeśli przez r_k oznaczyć odległość do *k*-tego sąsiada i $r_k \geq r_i$, dla $i=1...k-1$, wtedy stożkowa funkcja skalująca $G(d) = 1-d/\alpha r_k$, $\alpha > 1$, przyjmuje wartości $G(0)=1$ oraz $G(r_k)=1-1/\alpha$. Dla dużego α stożek jest bardzo szeroki i wszystkie wektory wewnątrz są uwzględnione jednakowo; dla α zmierzającego do wartości 1 sąsiad najdalej położony ma wagę bliską zeru. Metoda opisana powyżej musi działać nie gorzej niż konwencjonalny *k*-NN.

Zamiast wymuszać dokładnie *k* sąsiadów, można użyć promienia sfery *r* wokół wektorów testowych **X** jako parametru adaptacyjnego. W tym przypadku liczba błędów klasyfikacji lub prawdopodobieństwo klasyfikacji $p(\mathbf{C}_i|\mathbf{X};r) = N_i / \sum_j N_j$ jest optymalizowane na zbiorze treningowym za pomocą kroswalidacji. W neuronowej wersji tego algorytmu należy użyć funk-

cji transferu typu „twarda sfera". Jeśli żaden wektor referencyjny nie wpada do sfery o promieniu r lub osiągnięto takie samo prawdopodobieństwo klasyfikacji $p(\mathbf{C}_i|\mathbf{X};r)$ dla wszystkich klas, wektor testowy $\mathbf{X}$ może być przesunięty do zbioru przypadków niemożliwych do sklasyfikowania. Aby uniknąć takich problemów r powinno być zwiększona aż zostanie przełamany impas. Można też wprowadzić zmienne promienie r_i dla każdego wektora referencyjnego co znacznie powiększa liczbę parametrów adaptacyjnych.

Selekcja i parametryzacja wektorów referencyjnych

W najprostszym przypadku wszystkie wektory treningowe można przyjąć jako wektory referencyjne. Z kilku względów może to być nienajlepsze rozwiązanie:

1) Jeśli danych jest bardzo dużo większość wektorów i tak nic nie wnosi, spowalniając niepotrzebnie poszukiwanie najbardziej podobnych kandydatów.

2) Jeśli danych jest mało warto dostawić wirtualne wektory referencyjne lub zoptymalizować położenie istniejących.

3) Jeśli dane są zaszumione lub zawierają błędy zmniejszenie liczby wektorów referencyjnych może polepszyć wyniki.

Dodatkową zaletą redukcji liczby wektorów referencyjnych jest możliwość identyfikacji ciekawych prototypów, pozwalająca lepiej zrozumieć strukturę danych. Wyboru wektorów referencyjnych dokonać można na wiele sposobów. Pierwsza grupa metod, działająca dobrze dla dużych baz treningowych, oparta jest na metodach klasteryzacji. Należy wybrać względnie mały zbiór wektorów referencyjnych spośród wektorów leżących blisko centrów klastrów. Następnie sprawdza się dokładność klasyfikacji na zbiorze treningowym. Za każdym razem, kiedy system popełni błąd przy klasyfikacji wektora treningowego jest on przesuwany do zbioru referencyjnego.

Wariantem metod klasteryzacyjnych jest stopniowe zmniejszanie rozdzielczości danych. Przyjmując rozdzielczość r nie rozróżniamy danych w przedziale $[a, a+r)$. Używając funkcji zaokrąglania możemy otrzymać rozdzielczość C+1 przedziałów za pomocą wzoru

$$X_r(X) = X_{\min} + r\,\mathrm{Round}\left(\left(X - X_{\min}\right)/r\right);\; r = \left(X_{\max} - X_{\min}\right)/C$$

Dla C=1 wszystkie wartości X_r przechodzą w X_{min} lub X_{max} a dla dużego C stają się coraz bardziej podobne do pierwotnych wartości, pozwalając na utworzenie większej liczby klastrów.

Druga grupa metod opiera się na usuwaniu z całego zbioru treningowego tych wektorów, dla których k najbliższych sąsiadów jest z tej samej klasy. Te wektory leżą daleko od brzegów obszarów decyzji i dlatego wszystkie nowe wektory z ich sąsiedztwa będą i tak jednoznacznie klasyfikowane. Dobry prototyp leżący w środku klastra można znaleźć poszukując wektorów, które mają największą liczbę sąsiadów z tej samej klasy, dopuszczając pewien procent błędów.

Metody z trzeciej grupy eliminują ze zbioru referencyjnego kolejne wektory treningowe za każdym razem testując klasyfikator na całym zbiorze treningowym. Jeśli $K \le \delta$, gdzie K to osiągnięta dokładność klasyfikacji wyrażona w procentach, a $0 < \delta \le 100$ jest dokładnością wymaganą przez użytkownika, obliczoną np. za pomocą testu *leave-one-out* na całym zbiorze treningowym, to wybrany wektor musi pozostać i służyć jako wektor referencyjny, w przeciwnym przypadku wektor taki można odrzucić. Dobierając odpowiednio wartość δ można w przybliżony sposób regulować liczbę wektorów referencyjnych (im mniejsze δ tym w zbiorze referencyjnym pozostaje mniej wektorów). Wadą tej metody jest to, że jest dość kosztowna,

ponieważ trzeba przeprowadzić tyle ocen błędu ile jest wektorów treningowych. Do zalet zaliczyć można to, że metoda generuje bardzo małe zbiory treningowe przy minimalnej degradacji dokładności klasyfikacji, czasami nawet poprawiając generalizację.

Selekcja wektorów referencyjnych pozwala w znaczny sposób zmniejszyć zbiór treningowy, a co za tym idzie przyspieszyć obliczenia. Można pójść jeszcze dalej i optymalizować położenia wybranych wektorów referencyjnych, co powinno zmniejszyć błąd uczenia (Laaksonen, Oja 1996). Prowadzi to do adaptacyjnych metod kwantyzacji wektorowej (LVQ). Położenie wektora referencyjnego **R** znajdującego się w otoczeniu wektora treningowego **X** powinno być uaktualniane zgodnie ze wzorem:

$$\mathbf{R}_{nowy} = \mathbf{R}_{stary} + \eta\left(2\delta\left(\mathbf{C}(\mathbf{X}), \mathbf{C}\left(\mathbf{R}_{stary}\right)\right) - 1\right)\left(\mathbf{X} - \mathbf{R}_{stary}\right) \tag{8}$$

gdzie η jest stałą uczenia, która może być stopniowo zmniejszana. Jest ona mnożona przez (+1) jeśli **X** i $\mathbf{R}_{stary}$ należą do tej samej klasy, lub (-1) w przeciwnym przypadku. Więcej na temat różnych wersji metody LVQ znaleźć można w pracy (Laaksonen, Oja 1996). Błędy w wybranych obszarach przestrzeni cech można zlikwidować wprowadzając dodatkowe, „wirtualne" wektory referencyjne, pomagające wyostrzyć granice decyzji.

Funkcja kosztu i metody minimalizacji

Najprostszą funkcją kosztu jest liczba popełnianych błędów. Jest to oczywiście funkcja zmieniająca się skokowo, co wyklucza stosowanie szybkich, gradientowych metod minimalizacji. Jeśli obliczone prawdopodobieństwa klasyfikacji zmieniają się w sposób ciągły możemy stosować funkcję oceniająca ryzyko klasyfikacji:

$$E_R(\{\mathbf{X}\}; M) = \sum_{i,p} R\left(C_i, C(\mathbf{X}^p)\right)\left[p\left(C_i|\mathbf{X}^p; M\right) - \delta\left(C_i, C(\mathbf{X}^p)\right)\right]^2 \tag{9}$$

Należy jednak pamiętać, że minimum tej funkcji nie musi być identyczne z minimum błędu klasyfikacji. Sieci neuronowe uczone są najczęściej metodą wstecznej propagacji, która minimalizuje funkcję mierzącą średni błąd kwadratowy (MSE), ale metody gradientowe nie gwarantują znalezienia minimalnego błędu klasyfikacji, którego ocena podawana jest jako wynik końcowy. To właśnie z tego powodu dla niektórych baz danych lepsze rezultaty osiągnąć można za pomocą prostej dyskryminacji liniowej, chociaż mogło by się wydawać, że te same rezultaty powinna dać sieć MLP z jednym neuronem (Duch i Adamczak 1998). Inicjalizacja sieci wagami otrzymanymi z liniowej dyskryminacji (LDA) daje równie dobre rezultaty co LDA tylko dla dużych skosów funkcji sigmoidalnych. W wyniku uczenia neuronu błąd MSE może maleć przy rosnącym błędzie klasyfikacji.

W lokalnej regresji opartej na minimalno-odległościowych ocenach za funkcję błędu przyjmuje się:

$$E_R(\{\mathbf{X}\}; M) = \sum_{i,p} K\left(d\left(\mathbf{R}^i, \mathbf{X}^p\right)\right)\left[F\left(\mathbf{X}^p; M\right) - Y^p\right]^2 \tag{10}$$

gdzie Y^p jest pożądaną wartością funkcji w punkcie $\mathbf{X}^p$, a funkcja K(d) ocenia wpływ okolicznych wektorów referencyjnych na miarę błędu. Jeśli funkcja ta ma ostre maksimum wokół d=0 to wymusza ona przechodzenie dokładnie przez punkty (X^p,Y^p); jeśli jest ona stała to wszystkie punktu traktowane są jednakowo. W przypadku klasyfikacji można w ten sposób wymusić rozmiary sąsiedztwa wokół wektorów referencyjnych, w których żądamy zwiększonej dokładności klasyfikacji.

Metody gradientowe są podstawą uczenia sieci neuronowych, jednakże nie nadają się do minimalizacji nieciągłych funkcji kosztu. W tym celu używaliśmy trzech metod minimalizacji:

lokalnej metody sympleksów (Nelder i Mead 1965), adaptacyjnej metody symulowanego wyżarzania ASA (Ingberg 1996; jest to globalna metoda minimalizacji) oraz globalnej metody multisympleks (Duan i inni 1993). Najszybszą metodą jest oczywiście lokalna metoda sympleksów, która zapewnia względnie szybką zbieżność nawet dla dużych baz danych. Jest to cenna metoda, gdyż pozwala ocenić czy dla danej bazy danych skalowanie ma szansę działać i czy warto stosować kosztowniejsze metody optymalizacji. Jej wadą jest duża wariancja wyników, co wiąże się z lokalnością metody. Dla baz danych z osobno wydzielonym zbiorem testowym metody ASA i multisimpleks dają podobne rezultaty dla wielu uruchomień systemu (zaleta globalnej minimalizacji), są to jednak metody bardzo kosztowne.

Mieszanie wielu modeli

Stabilizację różnych metod osiągnąć można na kilka sposobów: przez metody statystycznego próbkowania danych, takie jak *arcing, bagging, stacking*, pozwalające wygenerować wiele modeli dla tego samego klasyfikatora uczonego na różnych podzbiorach danych, lub przez stworzenie „komitetu ekspertów", których wyniki uśredniane są dla wielu modeli. Zwykle przyjmuje się opinię większości ekspertów lub stosuje bardziej złożone metody kombinacji wyników (Merz 1999).

Przy tworzeniu modeli korzystających z wyników różnych klasyfikatorów należy wykorzystać wszystkie informacje, które mamy do dyspozycji. W przypadku modeli opartych na podobieństwie informacją taką może być ocena kompetencji danego modelu w pobliżu każdego z wektorów referencyjnych. Dlatego zamiast zwykłej kombinacji liniowej wyników wielu modeli należy zastosować kombinację ze współczynnikami pomnożonymi przez funkcję oceny kompetencji w danym obszarze.

$$p\left(C_i \mid \mathbf{X}; M\right) = \sum_l W_l g_l\left(\mathbf{X}; \mathbf{R}\right) p\left(C_i \mid \mathbf{X}; M_l\right) \tag{11}$$

gdzie funkcja $g_l(\mathbf{X};\mathbf{R})$ przyjmuje małe wartości w pobliżu tych wektorów referencyjnych, które nie są klasyfikowane poprawnie, a w pozostałych obszarach jest stała. Współczynniki W_l dopasować można metodą najmniejszych kwadratów.

3. Sieci MLP a metody oparte na podobieństwie

Może się wydawać, że pomiędzy sieciami neuronowymi opartymi na zlokalizowanych funkcjach radialnych, które w naturalny sposób uznać można za wersję metod opartych na podobieństwie, a sieciami typu perceptronów wielowarstwowych istnieje zasadnicza różnica. Działanie sieci pierwszego rodzaju bliższe jest metodom analizy skupień i polega na przybliżaniu rozkładu gęstości prawdopodobieństwa wektorów należących do różnych klastrów, a więc obliczaniu warunkowego prawdopodobieństwa $P(\mathbf{X}|C)$. Sieci drugiego rodzaju bliższe są analizie dyskryminacyjnej, posługują się hiperpłaszczyznami by dokonać podziału przestrzeni cech na rozłączne obszary, obliczając przy tym prawdopodobieństwa *a posteriori* $P(C|\mathbf{X})$. Korzystając z twierdzenia Bayesa:

$$P\left(C \mid \mathbf{X}\right) = \frac{P\left(\mathbf{X} \mid C\right) P(C)}{P(\mathbf{X})}$$

Zakładając dwie klasy C_1 i C_2 możemy to zapisać w postaci:

$$P(C_1 \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C_1) P(C_1)}{P(\mathbf{X} \mid C_1) P(C_1) + P(\mathbf{X} \mid C_2) P(C_2)} = \frac{1}{1 + \frac{P(\mathbf{X} \mid C_2) P(C_2)}{P(\mathbf{X} \mid C_1) P(C_1)}}$$

Przedstawiając teraz

$$\frac{P(\mathbf{X} \mid C_2) P(C_2)}{P(\mathbf{X} \mid C_1) P(C_1)} = \exp\left\{ -\ln \frac{P(\mathbf{X} \mid C_1)}{P(\mathbf{X} \mid C_2)} - \ln \frac{P(C_1)}{P(C_2)} \right\} = \mathrm{e}^{-\chi}$$

możemy formalnie zapisać:

$$P(C_1 \mid \mathbf{X}) = \sigma(\chi)$$

Przyjmując *d*-wymiarową funkcję Gaussa dla prawdopodobieństwa warunkowego, oraz jednakowe macierze kowariancji Σ dla obu klas:

$$P(\mathbf{X} \mid C_i) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(\mathbf{X} - \bar{C}_i)^T \Sigma^{-1} (\mathbf{X} - \bar{C}_i)} = G(\mathbf{X}; \bar{C}_i, \Sigma)$$

łatwo jest zapisać prawdopodobieństwa posterioryczne w postaci funkcji logistycznej:

$$P(C_1 \mid \mathbf{X}) = \sigma\left(\mathbf{W}^T \mathbf{X} - \theta\right)$$

gdzie

$$\mathbf{W} = \Sigma^{-1}\left(\bar{C}_1 - \bar{C}_2\right), \quad \theta = \frac{1}{2} \bar{C}_1^T \Sigma^{-1} \bar{C}_1 - \frac{1}{2} \bar{C}_2^T \Sigma^{-1} \bar{C}_2 + \ln \frac{P(C_1)}{P(C_2)}$$

W tym szczególnym przypadku można więc przejść od funkcji Gaussowskich do funkcji sigmoidalnych. Nietrudno zauważyć, że dla diagonalnych macierzy kowariancji renormalizacja funkcji Gaussowskich prowadzi do funkcji sigmoidalnych:

$$G_R(\mathbf{X}; C, \Sigma) = \frac{G(\mathbf{X}; C, \Sigma)}{G(\mathbf{X}; C, \Sigma) + G(\mathbf{X}; -C, \Sigma)} = \sigma\left(\mathbf{W}^T \mathbf{X}\right)$$

gdzie $W_i = 4C_i / \Sigma_{ii}$. Pary neuronów realizujące funkcję Gaussa mogą więc zastąpić funkcję logistyczną. Możemy się tu posłużyć prostszym i bardziej intuicyjnym argumentem, ukazującym związek metod opartych na podobieństwie i analizy dyskryminacji. Hiperpłaszczyzna, stanowiąca granicę decyzji między klasami, jest równoodległa od dowolnych dwóch punktów, które leżą na prostej do niej prostopadłej. Przyjmując te punkty za wektory prototypowe dla dwóch klas w metodzie najbliższych sąsiadów i obliczając odległość Euklidesową otrzymamy więc równoważny system klasyfikujący. Wybór najlepszej płaszczyzny rozdzielającej dwie klasy za pomocą liniowej analizy dyskryminacyjnej lub uczenia perceptronu jest równoważny

ustaleniu położenia jednego z wektorów prototypowych i poszukiwaniu najlepszego położenia drugiego z tych wektorów.

Jeśli założymy binarne sygnały wejściowe $\{X_i=\pm 1\}$ i wagi $\{W_i=\pm 1\}$, to neuron o N wejściach i progu θ realizuje następującą funkcję:

$$\Theta\left(\sum_i^N W_i X_i - \theta\right) = \begin{cases} 0 & \text{if } \|\mathbf{W}-\mathbf{X}\| > (N-\theta)/2 \\ 1 & \text{if } \|\mathbf{W}-\mathbf{X}\| \le (N-\theta)/2 \end{cases} \tag{12}$$

gdzie norma $\|\cdot\|$ jest zdefiniowana przez odległość Hamminga. Wagi neuronów w pierwszej warstwie ukrytej można interpretować jako adresy wektorów referencyjnych z przestrzeni wejściowej. Neurony progowe są wzbudzane przez wektory wejściowe wpadające w sferę o promieniu θ scentrowaną w **W**. Zmieniając wartości binarne na rzeczywiste oraz progową funkcję transferu na sigmoidalną oraz dokonując normalizacji $\|\mathbf{X}\|=\|\mathbf{W}\|=1$ można uzyskać miękkie wzbudzenie neuronów przez wektory wejściowe bliskie **W** na sferze jednostkowej. W ogólności funkcja transferu neuronów ma postać:

$$\sigma(\mathbf{W}\cdot\mathbf{X}) = \sigma\left(\frac{1}{2}\left(\|\mathbf{W}\|^2 + \|\mathbf{X}\|^2 - \|\mathbf{W}-\mathbf{X}\|^2\right)\right) = \sigma\left(I_{\max} - d(\mathbf{W},\mathbf{X})\right) \tag{13}$$

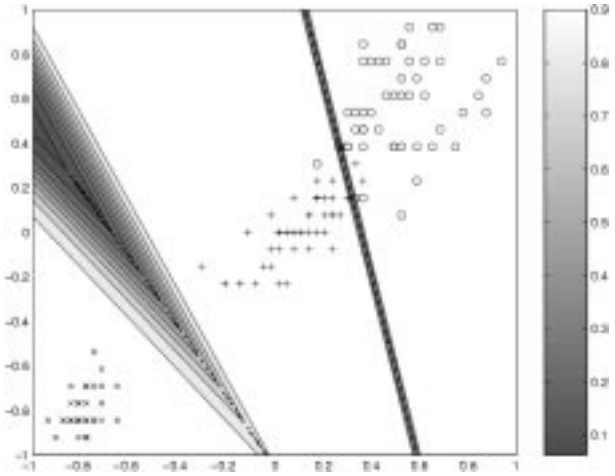
Dla znormalizowanych wektorów wejściowych sigmoidalne (lub inne monotonicznie rosnące) funkcje transferu obliczają wpływ wektorów referencyjnych **W** na prawdopodobieństwo klasyfikacji $p(\mathbf{C}_i|\mathbf{X};\{\mathbf{W},\theta\})$ w zależności od ich odległości $d(\mathbf{W},\mathbf{X})$. Jako funkcja odległości, funkcja transferu $\sigma(I_{\max} - d(\mathbf{W},\mathbf{X}))$ monotonicznie maleje, osiągając wartość 0.5 dla $d(\mathbf{W},\mathbf{X})=I_{\max}$. Dla znormalizowanych wag i wektorów wejściowych

$$\sigma(I_{max} - d(\mathbf{W},\mathbf{X})) \in [0.5, \sigma(1)],$$

przy czym $0 \le d(\mathbf{W},\mathbf{X}) \le 1$, co oznacza, że tylko część zakresu funkcji sigmoidalnej jest wykorzystana. Dla znormalizowanych **X** i dowolnego **W** zakres argumentu sigmoidy leży w przedziale $[-|\mathbf{W}|,+|\mathbf{W}|]$. Sigmoida unipolarna ma maksymalną krzywiznę około wartości ±2.4 i dlatego wagi mniejsze od tej wartości oznaczają, że sieć będzie działać w obszarze prawie liniowym. Metody regularyzacji dodają człony kary do funkcji błędu spychając wagi do małych wartości i tym samym wygładzając dopasowanie sieci do danych treningowych. Dla niektórych danych polepsza to generalizację sieci. Jeśli jednak potrzebne są ostre granice decyzji lub silne nieliniowości regularyzacja może pogorszyć generalizację.

W perceptronach wielowarstwowych funkcje sigmoidalne oceniają wpływ wag zależnie od odległości pomiędzy wagami a wektorami treningowymi. W sieciach typu RBF używane są Euklidesowe miary odległości $d(\mathbf{R},\mathbf{X})=\|\mathbf{X}-\mathbf{R}\|$ i eksponencjalne funkcje wagowe $\exp(-d^2)$. Zmieniając funkcję odległości w równaniu (13), np. używając miarę odległości Minkowskiego, można tworzyć nowe typy sieci neuronowych, nazwane przez nas (Duch i inni 1999) D-MLP (od *Distance-based MLPs*).

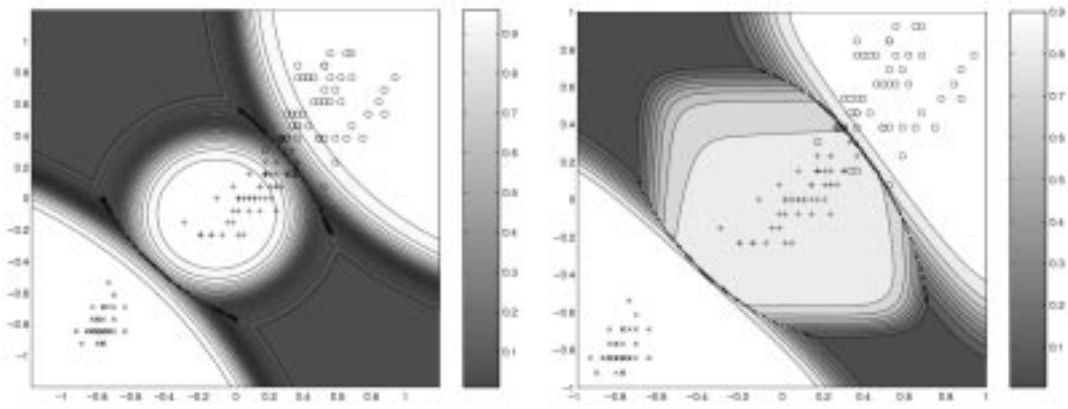
Na rysunkach przedstawiony jest efekt wprowadzenia dodatkowego wymiaru na kształt granic decyzji dla danych Iris. Chociaż te dane mają 4 cechy dla wygody pokazano tylko dwie ostatnie cechy, pozwalające na znalezienie optymalnego (z punktu widzenia generalizacji) rozwiązania, odpowiadającego co najwyżej 3 błędom. Minimalna sieć MLP ma w tym przypadku dwa wejścia, 3 neurony ukryte i 3 wyjścia (po jednym dla klasy) a jej granice decyzji pokazane są na rysunku poniżej.



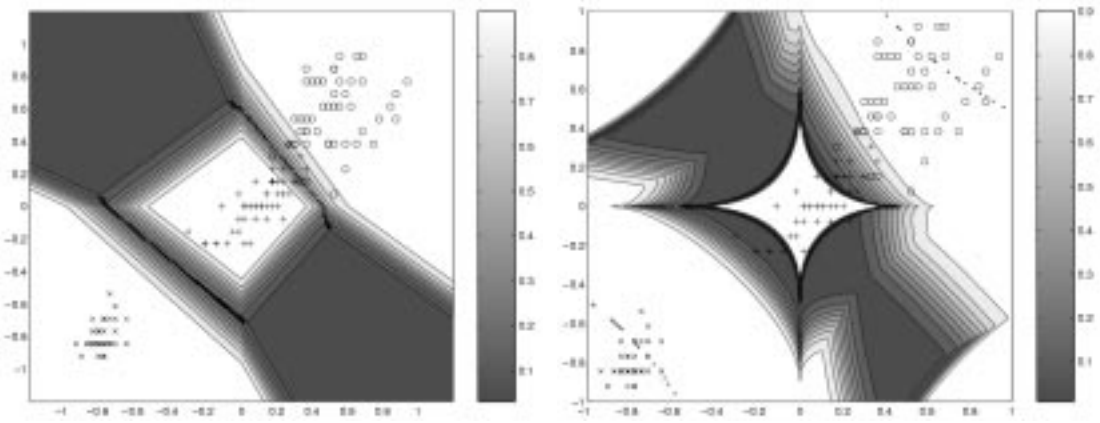
Wprowadzenie dodatkowego wymiaru pozwala zrezygnować z ukrytych neuronów, pozostawiając tylko 3 neurony wejściowe i 3 neurony wyjściowe. Zastosujemy uogólnioną miarę Minkowskiego

$$d\left(\mathbf{A},\mathbf{B};\mathbf{s}\right)^{\alpha}=\sum_{i=1}^{N}s_i d\left(A_i,B_i\right)^{\beta}$$

Dla α = 1 i β = 2 dostajemy w równaniu (13) kwadrat odległości, odpowiadający zwykłym hiperpłaszczyznom; dla $\alpha = \beta = 2$ dostajemy odległość Euklidesową, odpowiadającą powstaniu jednostek kołowych. Dla α =2 a β dużego (np. β = 7) kształt granic decyzji zbliża się do prostokątnego. Te dwa przypadki pokazane zostały na rysunkach poniżej. Dla β =1 lub mniejszego kształt granic decyzji robi się początkowo romboidalny a później przybiera kształt wklęsły, jak pokazano na dwóch kolejnych rysunkach.



Granice decyzji dla $\alpha = \beta = 2$ oraz $\alpha = 2, \beta = 7$



Granice decyzji dla $\alpha = 2$, $\beta = 1$ oraz $\alpha = 2$, $\beta = 0.5$

4. Pamięć asocjacyjna, dopełnianie wzorców i brakujące wartości

Metody należące do schematu SBM, takie jak metody najbliższych sąsiadów, można w naturalny sposób wykorzystać jako pamięć auto lub hetero-asocjacyjną. Mając dany wektor $\mathbf{X}=(\mathbf{X}_d, \mathbf{X}_n)$, w którym $\mathbf{X}_d$ to wartości znane a $\mathbf{X}_n$ to wartości szukane, możemy szukać najbardziej podobnych wektorów w podprzestrzeni znanych cech $\mathbf{X}_d$. Brakujące wartości cech $\mathbf{X}_n$ można określić za pomocą interpolacji biorąc *k* najbliższych wektorów referencyjnych. Jeśli znana jest klasa wektora **X** to można się ograniczyć tylko do wektorów referencyjnych z tej klasy, jeśli klasa nie jest znana to należy wybrać wektory z klasy dominującej w pobliżu $\mathbf{X}_d$. Liczbę sąsiadów *k* można zoptymalizować w podprzestrzeni $\mathbf{X}_d$ wykonując odpowiednie testy.

Uzupełnianie brakujących wartości zrealizować można na kilka sposobów. W wielu przypadkach wektory zawierające brakujące wartości usuwane są ze zbioru treningowego lub wstawiane są uśrednione lub najczęściej spotykane wartości. Prowadzi to do utraty cennej informacji lub wprowadza mylną informację. Na przykład dane „echocardiogram" z repozytorium UCI (Mertz i Murphy 1999) zawierają 132 wektory o 12 cechach, z których tylko cechy 1-9 są użyteczne (druga z nich to klasa). Brakuje 15 wartości dla cechy 6, 11 wartości dla cechy 7 itd. Jeśli cechy zawierające brakujące wartości zostaną zignorowane, używając sieci FSM z gaussowskimi funkcjami transferu (Duch i Diercksen 1995; Duch i inni 1997) 10-krotna stratyfikowana kroswalidacja daje dokładność 87.8% dla średniej liczby neuronów równej 24. W konstruktywistycznym algorytmie, jakim posługuje się FSM, powstają różne liczby neuronów dla różnych podziałów danych. Liczba tworzonych neuronów zależna jest od złożoności danych.

Wstawianie wartości średnich dla wszystkich klas zmniejszyło dokładność do 85.5% (przy średnio 20 neuronach), a wstawianie nietypowych wartości (np. –100) obniżyło dokładność jeszcze bardziej do 81.5% (przy średnio 22 neuronach). Dane Hepatitis, z tego samego repozytorium UCI, mają 155 wektorów o 18 cechach, 13 z nich to cechy binarne a pozostałe przyjmują kilka wartości całkowitych. Ostatnia cecha ma 67 brakujących wartości, cecha 16 ma 29 brakujących wartości itd. Ignorowanie cech zawierających brakujące wartości dla 10-krotnej stratyfikowanej kroswalidacji daje dokładność 79.9% (przy średnio 19 neuronach), wstawianie średnich daje 81.0% (12 neuronów) a wstawienie wartości nietypowej 79.1% (16 neuronów). Podobne zachowanie zaobserwować można na innych danych.

Załóżmy, że dwu-wymiarowe wektory skupione są wokół dwóch centrów, (1,1) oraz (2,2), a pierwsze skupienie zawiera dwa razy więcej wektorów niż drugie. Weźmy wektor **X** dla którego pierwsza cecha $X_1 = 1.9$ a druga jest nieznana. Poszukując sąsiadów wektora **X** w pod-

przestrzeni X_1 i interpolując wartość X_2 na podstawie wartości znalezionych w wektorach referencyjnych znajdziemy przybliżoną poprawną wartość (około 2). Używając wartości średnich lub wartości najczęściej występujących otrzymamy nieprawidłową wartość (bliską 1). Mamy tu do czynienia ze znanym w statystyce dylematem: w oparciu o jaką grupę danych można robić przewidywania? Powinny to być dane najbardziej specyficzne, tzn. dane dotyczące lokalnych skupień, a nie średnie lub najczęściej występujące wartości. Dylemat polega na tym, że im bardziej specyficzne skupienia rozpatrujemy, tym mniej liczne są dane, na podstawie których możemy dokonać interpolacji.

W wielu zastosowaniach dane zbierane są w sposób hierarchiczny: początkowa grupa testów pozwala stawiać hipotezy, które są następnie potwierdzane lub odrzucane przez kolejne testy. Odkrycie takiej hierarchicznej struktury na podstawie samych danych jest dużym wyzwaniem. Łączna analiza kwestionariuszy z różnych badań powinna rozróżniać dwie sytuacje: brak odpowiedzi na zadane pytania oraz brak odpowiedzi, ponieważ pytanie nie zostało zadane. W statystyce problem ten znany jest jako „wielokrotna imputacja danych" (*multiple imputation*) i rozpatrywany często w kontekście analizy preferencji politycznych, określanych za pomocą kwestionariuszy przez różne instytucje. Standardowa teoria pochodząca od Rubina (1996) opiera się na założeniach o gaussowskim rozkładzie danych. Założenia te często nie są spełnione.

W kontekście metod opartych na podobieństwie zastosować można uzupełnianie wartości wektorów treningowych oparte na interpolacji wartości pobliskich wektorów. Po uzupełnieniu wektory te wykorzystywane są do uczenia modelu. Najbardziej prawdopodobne nieznane wartości ocenić można maksymalizując:

$$P(\mathbf{X}_u \mid \mathbf{X}_d; M) = \max_{u',i} P(C_i \mid (\mathbf{X}_d, \mathbf{X}_{u'}); M)$$

dla danego modelu *M*, stopniowo doskonalonego dzięki dodawaniu do bazy referencyjnej coraz większej liczby wektorów z uzupełnionymi brakującymi wartościami. Poszukiwania maksimum prawdopodobieństwa dokonywane są w podprzestrzeni cech o nieznanych wartościach, dla ustalonych wartości $\mathbf{X}_d$. Jeśli brakuje tylko jedna wartość można ją uzupełnić dokonując jednowymiarowego przeszukiwania lokalnego maksimum. Początkowy model *M* oparty jest na wektorach nie zawierających brakujących danych. Jeśli takich wektorów nie ma należy znaleźć największy zbiór wektorów treningowych, które mają najwięcej znanych, wspólnych cech. Tak więc cecha, która nie ma dla większości wektorów określonych wartości nie jest brana pod uwagę w konstrukcji początkowego modelu, jednakże na późniejszym etapie znajdywane są jej najbardziej prawdopodobne wartości i model korygowany jest przy użyciu wektorów zawierających wszystkie cechy. Na każdym kroku należy jednak sprawdzać, czy dodawanie nowej cechy istotnie polepsza aktualnie istniejący model wykonując testy kroswalidacyjne, może się bowiem okazać, że nowa cecha nie zawiera istotnych informacji i jej uwzględnienie jedynie pogarsza wyniki.

Dla silnie za sobą sprzężonych cech problem wyboru początkowych cech, jak i porządku, w którym uzupełniane są brakujące wartości, może być bardzo złożony. Nie ma więc gwarancji, że uda się znaleźć model optymalny. W praktyce sieciowa realizacja modeli SBM znacznie ułatwia poszukiwanie maksymalnie prawdopodobnych wartości, gdyż zamiast kosztownych procesów szukania w wielowymiarowych przestrzeniach wystarczy sprawdzić wielkość pobudzeń poszczególnych węzłów sieci i użyć gradientowych metod w obszarze bliskim centrum węzła najbardziej wzbudzonego. Ponieważ sieci oferują analityczne przybliżenie do wyrażeń na gęstości prawdopodobieństwa można też zastosować statystyczne metody próbkowania danych (Neal 1996).

Zastosowanie opisanej powyżej metody maksymalizacji prawdopodobieństwa dla echokardiogramów poprawiło wyniki testów kroswalidacyjnych, dając dokładność 90.2% i zmniejszając liczbę neuronów do 18. Dla danych hepatitis otrzymujemy 83.4% dla średnio 10 neuronów. Zarówno dokładność, jak i złożoność systemu uległy wyraźnej poprawie.

5. Implementacja i realizacja sieciowa

Liczba możliwych wariantów metod opartych na podobieństwie jest bardzo duża i na razie jedynie część z opisanych tu algorytmów została zaimplementowana w rozwijanym przez nas systemie. W najprostszej z nich, metodzie najbliższych sąsiadów, oblicza się wszystkie odległości od wektora testowego do wektorów treningowych, wybierając *k* najbliższych. Taki system potrzebuje niewiele czasu na optymalizację parametrów (w najprostszym przypadku jest to tylko liczba sąsiadów *k*), ale jest stosunkowo wolny na etapie klasyfikacji. Chociaż Laaksonen i Oja (1996) twierdzą, że „... dla baz danych charakteryzujących się niezbyt dużą wymiarowością danych trudno znaleźć wariant metody *k*-NN, który byłby znacząco szybszy niż metoda *brute force*", różne usprawnienia przyspieszające działanie tego algorytmu zostały opisane w literaturze (np. Daelemans 1998).

W naszym systemie stosujemy buforowanie tablicy odległości (planujemy zaimplementować również inne usprawnienia). Pamiętana jest tablica z obliczonymi odległościami między wektorami treningowymi (dla potrzeb testu leave-one-out lub kroswalidacji) oraz, jeśli plik testowy jest wydzielony, tablica odległości między wektorami treningowymi a testowymi. W przypadku wielokrotnych obliczeń na tej samej bazie danych (np. w czasie optymalizacji parametrów klasyfikatora) już dla kilkudziesięciu atrybutów można znacznie zredukować czas obliczeń. Jeśli klasyfikacji będziemy dokonywać często na tych samych danych, innym rozwiązaniem jest zmniejszenie bazy treningowej za pomocą selekcji wektorów referencyjnych.

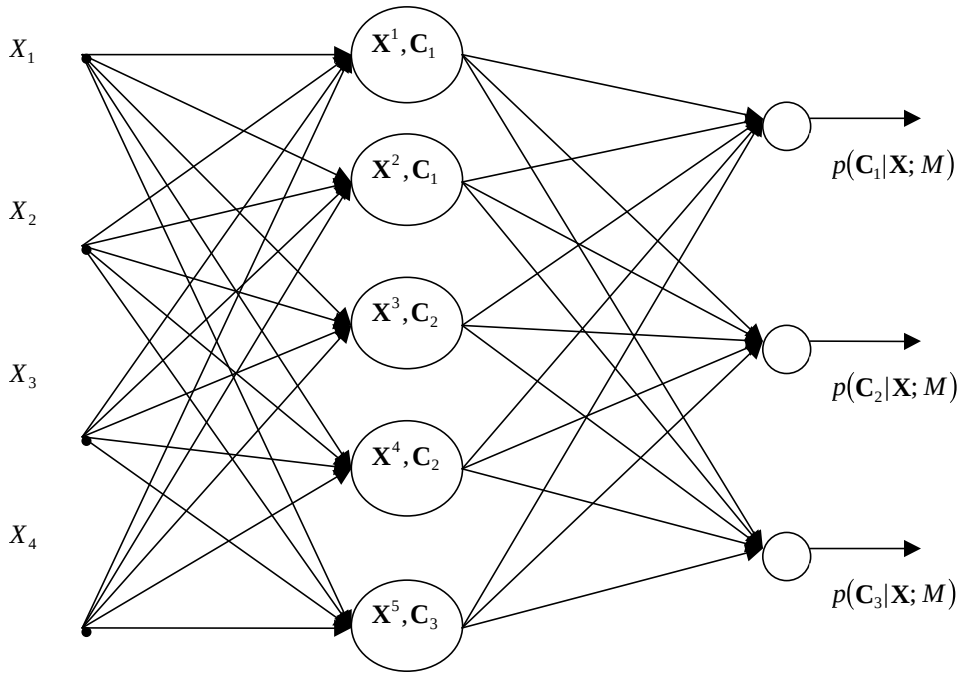
Realizacja sieciowa

Algorytm metody najbliższych sąsiadów może zostać przedstawiony w postaci sieci, która ma węzły ukryte, obliczające odległości $d(\mathbf{X}, \mathbf{R}^p)$. *k* węzłów o najmniejszej odległości oblicza etykietę klasy $h(\mathbf{X}, \mathbf{R}^p) = C_j$ a pozostałe węzły dają $h_j(\mathbf{X}, \mathbf{R}^p) = 0$. Warstwa wyjściowa sieci oblicza prawdopodobieństwa korzystając ze wzorów:

$$P(\mathbf{C}_i|\mathbf{X}; M) = \sum_j \mathbf{W}_{ij} \cdot h_j(\mathbf{X}), \quad p(\mathbf{C}_i|\mathbf{X}; M) = \frac{P(\mathbf{C}_i|\mathbf{X}; M)}{\sum_i P(\mathbf{C}_i|\mathbf{X}; M)} \tag{14}$$

Wagi $\mathbf{W}_{ij}$ dla połączeń z jednostkami wyjściowymi, obliczającymi prawdopodobieństwa dla poszczególnych klas $\mathbf{C}_i$, przyjmują wartości $\mathbf{W}_{ij} = \mathbf{S}(\mathbf{C}_i, \mathbf{C}_j)/\mathbf{C}_j$, gdzie $\mathbf{S}(\cdot)$ jest macierzą oceniającą podobieństwo pomiędzy klasami. W przypadku tradycyjnej metody *k*-NN zamiast tej macierzy używa się delty Kroneckera. Każdy wektor należący do *k* najbliższych sąsiadów wnosi wkład o wartości $\mathbf{S}(\mathbf{C}_i, \mathbf{C}_j)$ do prawdopodobieństwa dla klasy $\mathbf{C}_i$. Struktura sieci przedstawiona jest na rysunku powyżej. Jako funkcję kosztu podlegającą optymalizacji można przyjąć:

$$E(\mathbf{W}; M) = \sum_{\mathbf{X}} \sum_i R(\mathbf{C}_i, \mathbf{C}(\mathbf{X}))\left(p(\mathbf{C}_i|\mathbf{X}; M) - \delta(\mathbf{C}_i, \mathbf{C}(\mathbf{X}))\right)^2 \tag{15}$$



Rys. 1. Sieciowe uogólnienie metody k-NN.

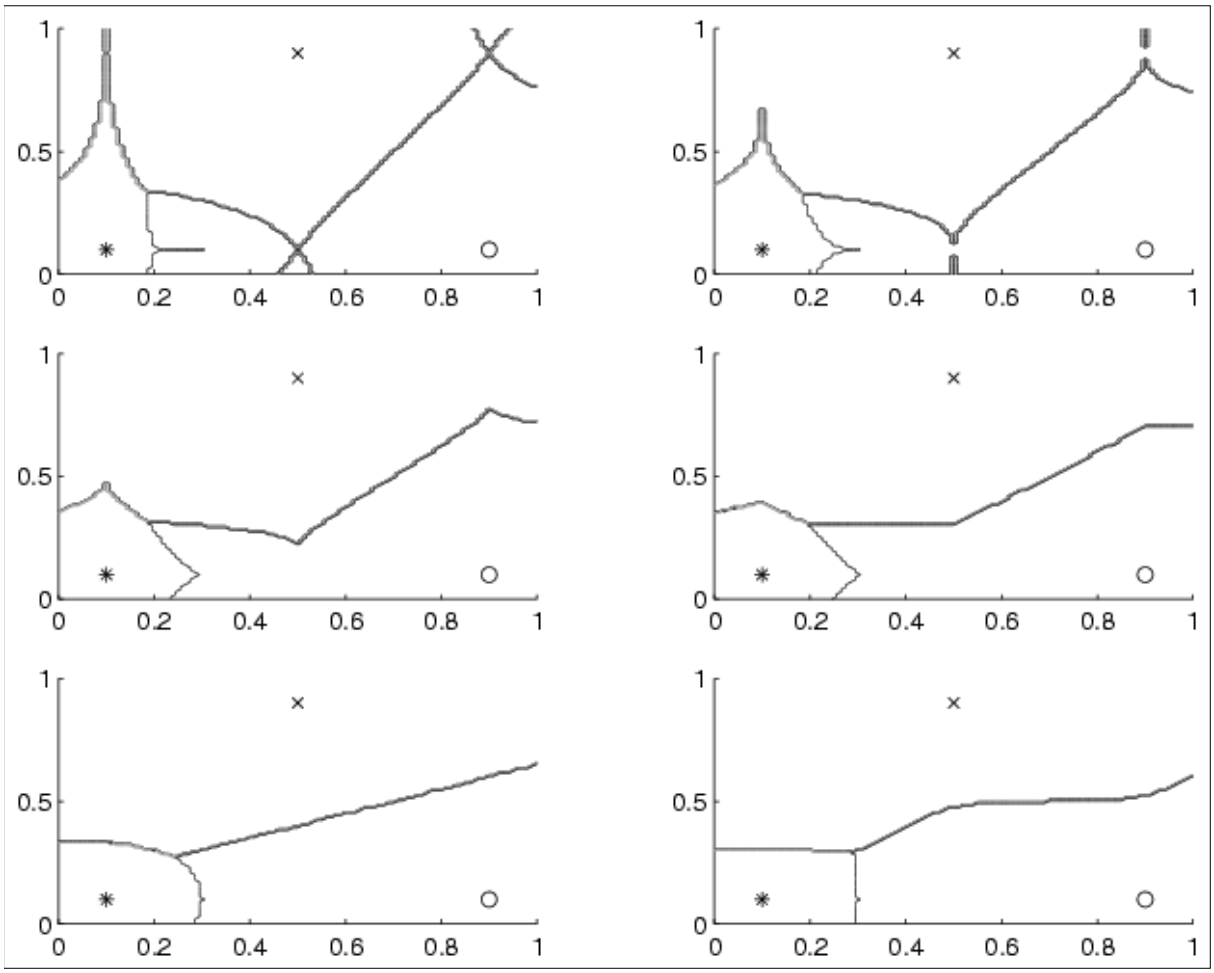
gdzie model M zawiera k jako parametr i $\mathbf{S}(\mathbf{C}_i, \mathbf{C}_j)$ jest macierzą podobieństw międzyklasowych. Chcąc zminimalizować liczbę błędów klasyfikacji, prawdopodobieństwa powinny być zamienione na wartości binarne 0, 1 przez procedurę „wygrywający bierze wszystko". Wagi wyjściowe, równe początkowo $\mathbf{W}_{ij} = \mathbf{S}(\mathbf{C}_i, \mathbf{C}_j)/\mathbf{C}_j$ mogą być traktowane jako parametry adaptacyjne. Wprowadzenie skalowania wpływu wektorów referencyjnych $G(d(\cdot))$ pozwala na zastosowanie gradientowych metod optymalizacji.

Dla wielu baz danych (zwłaszcza dla obrazów) taka sieć powinna działać lepiej niż sieci typu MLP i inne klasyfikatory, jako że rezultaty powinny być nie gorsze niż te, otrzymane z klasycznej metody k-NN (por. Michie i inni 1994).

Pojedynczy neuron, dostarczający hiperpłaszczyzny granicznej, można zastąpić odpowiednio umiejscowionym wektorem referencyjnym. Używając różnych funkcji odległości możemy w znacznej mierze wpływać na kształt granic decyzji. Jeśli mamy tylko jeden wektor referencyjny $\mathbf{R}_i$ dla każdej klasy przynależność do danej klasy określona jest przez funkcję dyskryminacyjną. Dla dwóch klas ma ona postać:

$$z(\mathbf{X}) = W_1 D(\mathbf{X}, \mathbf{R}_1) - W_2 D(\mathbf{X}, \mathbf{R}_2) - \theta$$

gdzie próg θ jest parametrem adaptacyjnym. Razem z wagami oraz położeniami wektorów referencyjnych daje to $2n+3$ parametry adaptacyjne (dla n-wymiarowych wektorów). Przedstawiając obliczenia funkcji dyskryminacyjnych w postaci sieci należy wprowadzić węzeł wyjściowy, sumujący wartości funkcji dyskryminujących po prototypach przypisanych do poszczególnych klas:



$$z(\mathbf{X}) = \sum_{i \in C_1} W_i D(\mathbf{X}, \mathbf{R}_i) - \sum_{i \in C_2} W_i D(\mathbf{X}, \mathbf{R}_i) - \theta$$

Skalowanie całej sumy zamiast poszczególnych odległości zmniejsza liczbę parametrów adaptacyjnych upraszczając kształty możliwych do otrzymania granic decyzji. Wektory referencyjne można wybrać z bazy treningowej i zoptymalizować za pomocą algorytmu LVQ lub minimalizując odpowiednią funkcję kosztu. Na rysunku przedstawiono granice decyzji dla przypadku dwuwymiarowego, w którym wyróżniono 3 prototypy, po jednym dla klasy. Waga pierwszego prototypu jest 3 razy większa niż pozostałe wagi. Poszczególne rysunki odpowiadają różnym wartościom wykładników dla odległości Minkowskiego.

6. Przykładowe wyniki

Nawet w swojej najprostszej postaci metoda najbliższych sąsiadów dała w empirycznych porównaniach projektu STATLOG (Michie i inni, 1994) w 4 z 22 baz danych najlepsze rezultaty i w 3 następnych prawie najlepsze rezultaty. Dotyczy to zwłaszcza baz danych związanych z obrazami. Proste modyfikacje tej metody, takie jak selekcja cech, optymalizacja liczby sąsiadów i funkcji odległości pozwoliły nam na osiągnięcie najlepszych rezultatów w ponad połowie wszystkich przypadków.

Metoda	%, zbiór trening.	%, zbiór testowy	Uwagi
IB2-IB4	81.2-85.5	43.6-44.6	WEKA, nasze obliczenia
Klasyfikator Bayesowski	--	46.6	WEKA, nasze obliczenia
1R (regułowy)	58.4	50.3	WEKA, nasze obliczenia
T2 (reguły z drzewa decyzji)	67.5	53.3	WEKA, nasze obliczenia
FOIL (indukcja, logiczny)	99	60.1	WEKA, nasze obliczenia
FSM, 49 klasycznych reguł Logicznych	83.5	63.2	FSM, nasze obliczenia
LDA (liniowa dyskryminacja)	68.4	65.0	nasze obliczenia
DLVQ (38 węzłów)	100	66.0	nasze obliczenia
C4.5 – reguły z drzewa decyzji	64.5	66.3	nasze obliczenia
Najlepsze rozmyte MLP	75.5	66.3	Mitra i inni (1997)
MLP z algorytmem RPROP		68.0	nasze obliczenia
MLP – Korelacja Kaskadowa		71.0	nasze obliczenia
Rozmyta sieć neuronowa	100	75.5	Hayashi i inni (1990)
C4.5 drzewo decyzji	94.4	75.5	nasze obliczenia
FSM, funkcje Gaussa	93	75.6	nasze obliczenia
FSM, 60 funkcji trójkątnych	93	75.8	nasze obliczenia
IB1c (minimalno-odległościowa)	--	76.7	WEKA, nasze obliczenia
1-NN, Manhattan	79.1	77.9	nasze obliczenia
K* method	--	78.5	WEKA, nasze obliczenia
1-NN, bez 4 cech, Manhattan	76.9	80.4	nasze obliczenia, KG
1-NN, ważone (ASA)	83.4	82.8	nasze obliczenia, KG

Dla niektórych baz danych, np. problemów z wątrobą („hepatobiliary disorders”, zbiór otrzymany z Tokyo Dental and Medical University (Hayashi i inni 1990), zawierający 536 przypadków, 163 wektory testowe, 9 cech i 4 klasy) osiągnięte przez nas wyniki są znacznie lepszy niż wyniki otrzymane za pomocą kilkunastu innych metod zastosowanych w tym przypadku (WEKA jest pakietem programów otrzymanym z Waikato University, Nowa Zelandia). Najlepsze wyniki udało się osiągnąć stosując najpierw selekcję a potem skalowanie pozostałych cech. Wynik o zbliżonej dokładności dał też komitet sieci FSM (Duch i inni 1997). Model ten mieści się również w schemacie metod SBM.

7. Podsumowanie

Schemat metod opartych na podobieństwie jest bardzo bogaty, zawiera w sobie w naturalny sposób większość modeli sieci neuronowych i innych klasyfikatorów. Jego zastosowania nie ograniczają się do zagadnień związanych z klasyfikacją. Opisaliśmy tu możliwości uzupełniania wzorców (*pattern completion*), pozwalające wykorzystać modele SBM jako pamięci hetero lub autoasocjacyjne. Modele SBM można również zastosować do zagadnień wymagających wielowymiarowej aproksymacji, wprowadzając różne metody interpolacji przy wykorzystaniu stosowanych metryk.

Mamy nadzieję, że badania prowadzone w tym kierunku przyniosą dwojakie korzyści. Z jednej strony badania te powinny doprowadzić do integracji i lepszego zrozumienia związków

pomiędzy wieloma metodami inteligencji obliczeniowej, wyrastającymi z technik rozpoznawania wzorców (*pattern recognition*), metod statystycznych, uczenia maszynowego, logiki rozmytej i sieci neuronowych. Z drugiej strony rozwijamy oprogramowanie, które dobierając najbardziej odpowiednią metodę należącą do schematu SBM do analizowanych danych powinno dać rezultaty statystycznie nieodróżnialne od najlepszych, jakie można uzyskać. Przyczyną stosunkowo niewielkiej popularności metod opartych na podobieństwie jest przede wszystkim brak powszechnie dostępnego oprogramowania.

Podziękowania: Za wsparcie finansowe w ramach grantu nr. 8 T11F 014 14 jesteśmy wdzięczni Komitetowi Badań Naukowych. Jesteśmy również wdzięczni Rafałowi Adamczakowi za dostarczenie wyników dotyczących uzupełniania danych za pomocą sieci FSM oraz wykonanie obliczeń porównawczych za pomocą pakietu WEKA.

Literatura

Breiman L., *The heuristics of instability in model selection.* Annals of Statistics 24 (1996) 2350-2383

Breiman L., *Bias-Variance, regularization, instability and stabilization*. W: C. Bishop, ed. Neural Networks and Machine Learning. Springer 1998

Daelemans W., Zavrel J., Ko van der Sloot, Antal van den Bosh, (1998) TiMBL: Tilburg *Memory Based Learner*, version 1.0, Reference Guide, ILK Technical Report – ILK 98-03, http://ilk.kub.nl/~ilk/papers/ilk9803.ps.gz

Duan, Q., V.K. Gupta i S. Sorooshian, (1993): *A Shuffled Complex Evolution Approach for Effective and Efficient Global Minimization*, Journal of Optimization Theory and Applications, 76 (1993) 501-521

Duch W, Adamczak R, Jankowski N. (!997): *Initialization of adaptive parameters in density networks*, Third Conference on Neural Networks and Their Applications, Kule, Październik 1997, pp. 99-104

Duch W., Adamczak R., Jankowski N. (1997a): *New developments in the Feature Space Mapping model*, 3rd Conference on Neural Networks and Their Applications, Kule, Październik 1997, pp. 65-70;

Duch W, Adamczak R, (1998): *Statistical methods for construction of neural networks.* International Conference on Neural Information Processing, ICONIP'98, Kitakyushu, Japonia, Październik 1998, Vol. 2, pp. 629-642

Duch W i Jankowski N, (1999): *Survey of neural transfer functions*, Neural Computing Surveys 2 (1999) 163-213

Duch W., Diercksen G.H.F., (1995): *Feature Space Mapping as a universal adaptive system*. Computer Physics Communications 87 (1995) 341-371

Duch W. i Grudziński K. (1999): *Weighting and selection of features in Similarity Based Methods*. Intelligent Information Systems VIII, Ustroń, Poland, 14-18.06.1999, pp. 32-36

Duch W. i Grudziński K. (1999a): *Search and global minimization in similarity-based methods.* International Joint Conference on Neural Networks (IJCNN'99), Washington, July 1999, praca #742

Duch W. i Korczak J.: *Optimization and global minimization methods suitable for neural networks*, Neural Computing Surveys (wysłane)

Fernandez M., Hernandez C., (1999): *Neural networks input selection by using the training set*, Proc. of the 1999 IEEE International Joint Conference on Neural Networks, Washington D.C., 1999 (w druku)

Gupta H.V., Hsu K. i S. Sorooshian, *Superior training of artificial neural networks using weight-space partitioning*, W Proc. of International Conference on Neural Networks, pp. 1919-1923, Houston, USA, 1997

Hayashi Y., Imura A., Yoshida, K. (1990): *Fuzzy neural expert system and its application to medical diagnosis*, w: 8th International Congress on Cybernetics and Systems, New York City 1990, pp. 54-61

Ingberg L., (1996) *Adaptive simulated annealing (ASA): Lessons learned*, J. Control and Cybernetics, 25 (1996) 33-54

Laaksonen J. i E. Oja (1996), *Classification with Learning k-Nearest Neighbors*. W: Proc. of ICNN'96, Washington, D.C., June 1996, pp. 1480-1483

Merz C.J., *Using Correspondence Analysis to Combine Classifiers*. Machine Learning 36 (1999) 33-58

Mertz C.J., Murphy P. M., *UCI repository*,
http://www.ics.uci.edu/pub/machine-learning-databases

Michie D., Spiegelhalter D.J., Taylor C.C., *Machine learning, neural and statistical classification*, Elis Horwood, London 1994

Neal R.M, *Bayesian Learning in Neural Networks*. Springer Lecture Notes in Statistics vol. 118 (1996)

Nelder, J.A., and Mead, R., Computer Journal 7 (1965) 308-313

Rubin D.B., *Multiple imputation after 18+ years*. J. of the American Statistical Association 91 (1996) 473-489

Rohwer R. i Morciniec M., *A Theoretical and Experimental Account of n-tuple Classifier Performance*, Neural Computation 8 (1996) 657-670

Wilson D.R. i Martinez T.R., *Improved Heterogeneous Distance Functions*, Journal of Artificial Intelligence Research 6 (1997) 1-34