Uniwersytet Mikołaja Kopernika

Marek Grochowski

Wybór wektorów referencyjnych dla wybranych metod klasyfikacji

Praca magisterska napisana w
Katedrze Informatyki Stosowanej
Wydziału Fizyki Astronomii i Informatyki Stosowanej
pod opieką dr N. Jankowskiego

Toruń 2003

Spis treści

Wstęp

Klasyfikacja wzorców jest bardzo ważnym obszarem zainteresowań sztucznej inteligencji i inteligencji obliczeniowej. Problem sprowadza się do określenia kategorii badanego obiektu, na podstawie wartości cech, jakimi ten obiekt opisujemy. Istnieją adaptacyjne modele potrafiące "nauczyć się" rozwiązywać powyższy problem w oparciu o wiedzę zawartą w odpowiednio przygotowanym zbiorze treningowym. Dany model klasyfikacji uczymy zazwyczaj na całym dostępnym zbiorze treningowym. Zakładamy więc, że dostępne dane są pewne, tzn. wiernie odzwierciedlają daną populację z której pochodzą badane obiekty. Musimy jednak liczyć się z tym, że dane mogą zawierać błędy pomiarowe i ich usunięcie z procesu uczenia może niejednokrotnie polepszyć wyniki. Poza tym znaczący wkład do nauki modelu może posiadać niewielki podzbiór przypadków treningowych i zastąpienie grupy wektorów jednym lub kilkoma reprezentatywnymi nie zmieni nic, albo prawie nic, w poprawności działania klasyfikatora. Odpowiednia selekcja wektorów referencyjnych może więc poprawić skuteczność klasyfikacji, jednocześnie zmniejszając złożoność obliczeniową stosowanych modeli. Ograniczając liczbę przypadków treningowych nie dopuścimy do "przeuczenia" modelu polepszając późniejszą generalizację. Takie uproszczenie problemu ułatwi śledzenie procesów zachodzących w czasie uczenia oraz może okazac się pomocne przy próbie zrozumienia zależności występujących w strukturze danych. Będziemy więc poszukiwać "super-wektorów" uczących, które wiarygodnie reprezentują wiedzę zawartą w zbiorze treningowym.

Celem tej pracy jest przybliżenie zagadnienia wyboru przypadków referencyjnych ze zbioru treningowego, czyli wektorów, które posłużą jako wzorce treningowe w procesie uczenia klasyfikatora oraz wykazanie wpływu takiego postępowania na działanie niektórych modeli klasyfikujących. Pierwszy rozdział zawiera opis najczęściej stosowanych metod klasyfikacji ze szczególnym zwróceniem uwagi na modele użyte w doświadczeniach. Rozdział drugi porusza problem selekcji wektorów oraz zawiera przegląd stosowanych w tym celu metod. W ostatnim rozdziale pracy znajdują się wyniki eksperymentalne przeprowadzone na szeregu zbiorów danych przy użyciu modeli opisanych w rozdziale drugim.

1. Klasyfikacja

1.1. Zdefiniowanie problemu

Dany jest zbiór, zwany zbiorem treningowym, zawierający obiekty pochodzące z pewnej badanej populacji

$$\mathcal{T} = \{\langle \mathbf{x}_1, c(\mathbf{x}_1)\rangle, \langle \mathbf{x}_2, c(\mathbf{x}_2)\rangle, \ldots, \langle \mathbf{x}_n, c(\mathbf{x}_n)\rangle\} \tag{1.1}$$

Obiekt opisany jest zestawem charakteryzujących go cech, których wartości tworzą wektor

$$\mathbf{x} = [x_1, x_2, ..., x_d] \in \mathcal{R}^d \tag{1.2}$$

Każdy obiekt skojarzony jest z pewną klasą pojęć, tzn. wektorowi $\mathbf{x}_i$ przypisujemy etykietę

$$c(\mathbf{x_i}) \in \mathcal{C} = \{c_1, c_2, ..., c_k\} \tag{1.3}$$

oznaczającą jego przynależność do jednej z k klas. Parę $\langle \mathbf{x}, c(\mathbf{x})\rangle$ nazywamy przypadkiem, próbką lub przykładem. Celem klasyfikacji jest znalezienie odwzorowania

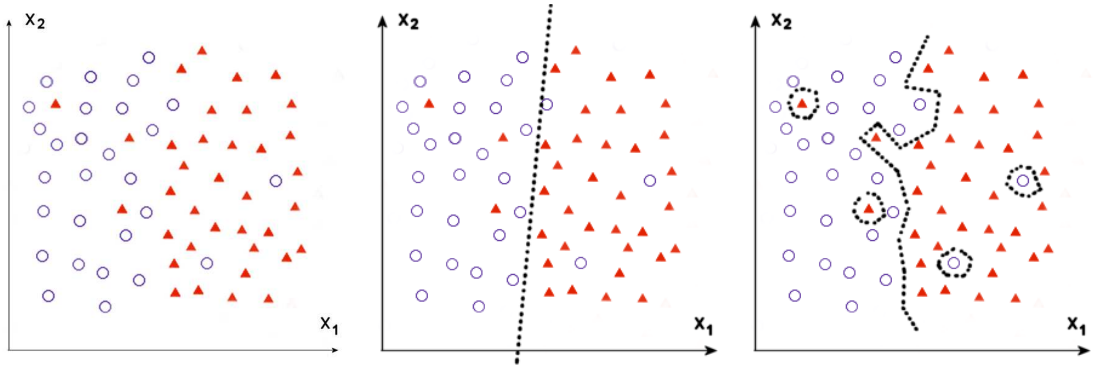
$$f : \mathcal{R}^d \to \mathcal{C} \tag{1.4}$$

Jest to więc szczególny przypadek poszukiwania funkcji aproksymującej, która odwzorowuje $\mathbf{x}$ na dyskretne wartości etykiet klas. Spodziewamy się, że dla dowolnego przypadku $\mathbf{x}$, nie pochodzącego ze zbioru treningowego, klasyfikator będzie potrafił przydzielić ten wektor do jednej z k znanych mu klas $f(\mathbf{x}) = c_i$. Innymi słowy klasyfikator podejmuje decyzję do której klasy należy przypisać dany obiekt. Jeżeli rozkład danych jest deterministyczny (tzn. każde dwa identyczne przypadki należą do tej samej klasy) wówczas istnieje taka funkcja f, która będzie bezbłędnie odwzorowywała wszystkie wektory treningowe $f(\mathbf{x_i}) = c(\mathbf{x_i})$ dla kadego $i = 1, 2, \ldots, n$. Zauważmy, że dla danej funkcji f oraz dowolnego zbioru $\tilde{\mathcal{T}} = \{\langle \tilde{\mathbf{x}}_\mathbf{1}, c(\tilde{\mathbf{x}}_\mathbf{1})\rangle, \ldots, \langle \tilde{\mathbf{x}}_{\tilde{\mathbf{n}}}, c(\tilde{\mathbf{x}}_{\tilde{\mathbf{n}}})\rangle\}$ takiego, że $\tilde{\mathcal{T}} \cap \mathcal{T} = \varnothing$, istnieje funkcja f^* taka, że $f^*(\mathbf{x_i}) = f(\mathbf{x_i})$ dla wszystkich $i = 1, \ldots, n$ ale $f^*(\tilde{\mathbf{x}}_\mathbf{i}) \neq f(\tilde{\mathbf{x}}_\mathbf{i})$ dla każdego $i = 1, \ldots, \tilde{n}$. W zagadnieniu klasyfikacji poszukujemy takiego odwzorowania, które nie tylko potrafi rozpoznać przypadki ze zbioru treningowego, ale bazując na wiedzy z niego uzyskanej, potrafi rozpoznać z możliwie minimalnym błędem przypadki nie należące do tego zbioru. Celem klasyfikacji jest więc znalezienie odwzorowania, które posiada jak najlepsze własności generalizujące.

1.2. Granice i obszary decyzyjne

Z geometrycznego punktu widzenia wektor $\mathbf{x} = [x_1, \ldots, x_d]$ określa punkt w d-wymiarowej przestrzeni cech. Model klasyfikujący, przydzielając każdy punkt z tej przestrzeni do jednej z k klas, przeprowadza podział przestrzeni cech na k obszarów decyzyjnych $R_1, \ldots, R_k$ tak, że wszystkie punkty w danym obszarze R_i przydzielane są do odpowiadającej mu

klasy c_i. Można więc powiedzieć, że klasyfikator niejawnie realizuje pewne funkcje dyskriminacyjne $g_i(\mathbf{x}), i = 1, \ldots, k$ przydzielając wektor $\mathbf{x}$ do klasy c_i jeśli $g_i(\mathbf{x}) > g_j(\mathbf{x})$ dla wszystkich $i \neq j$. Regiony decyzyjne rozdzielone są powierzchniami decyzyjnymi. Równanie $g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0$ definiuje taką płaszczyznę rozdzielającą sąsiadujące regiony decyzyjne R_i i R_j[3]. Kształt powierzchni decyzyjnej zależy od rozkładu danych w przestrzeni cech oraz od modelu i jego złożoności. Problem generalizacji w tym przypadku sprowadza się do poszukiwania powierzchni decyzyjnych, które nie są przesadnie dopasowane do danych treningowych (rys.1.1).



(a) (b) (c)

Rysunek 1.1. Przykład dwuwymiarowego dwuklasowego zbioru treningowego (a), oraz dwa skrajne przypadki powierzchni decyzyjnych (linia przerywana) - (c) przykład modelu przeuczonego, powierzchnia decyzyjna jest zbyt dopasowana do danych treningowych dając słabą generalizację, (b) najprostszy podział za pomocą linii prostej jeżeli tylko nie powoduje zbyt dużego błędu może okazać się dobrym rozwiązaniem

1.3. Uczenie modelu

Każdy model klasyfikujący, który korzysta z wiedzy zawartej w pewnym zbiorze treningowym możemy nazywać modelem uczącym[3]. W procesie uczenia model adaptuje wewnętrzne parametry zależnie od prezentowanych przypadków treningowych. Zbiór treningowy $\mathcal{T}$ w ogólności nie musi zawierać informacji o przynależności danego wektora do odpowiedniej klasy, wówczas model musi umieć wyłonić samodzielnie klasy ze struktury danych (uczenie bez nadzoru). Jeśli mamy wprost podaną wiedzę o przynależności do klasy danego wektora (jak to zostało zdefiniowane w naszym przypadku (1.1)), daje nam to możliwość oszacowania jakości uczenia poprzez określenie pewnej funkcji błędu

$$\tilde{E}[f] = \sum_i E(f(\mathbf{x_i}), c(\mathbf{x_i})) \tag{1.5}$$

Pozwala to na sterowanie kierunkiem uczenia (zwanym uczeniem pod nadzorem) tak aby zminimalizować powyższą funkcję. W najprostszym przypadku dla modeli adaptacyjnych funkcja błędu definiowana jest w postaci błędu średniokwadratowego

$$\tilde{E}[f] = \frac{1}{2} \sum_{i=1}^{n} |f(\mathbf{x}_i) - c(\mathbf{x}_i)| \tag{1.6}$$

1.4. Przegląd metod klasyfikacji

Poniżej opisane zostały pokrótce niektóre modele klasyfikacji. Szczegółowy opis ich działania nie jest tematem niniejszej pracy - bliższe informacje na ten temat zawiera cytowana literatura.

1.4.1. k-NN

Metoda k-NN (*k-Nearest Neighbors*) należy do metod opartych na poszukiwaniu podobieństw między przypadkami. Metody te wywodzą się z teorii rozpoznawania wzorców (*ang. pattern recognition*) a ich działanie polega na poszukiwaniu przypadków najbardziej podobnych, względem pewnej miary podobieństwa $d(\mathbf{x}, \mathbf{x}')$, do przypadku klasyfikowanego. W najprostszym przypadku, stosowanym w k-NN, funkcja $d(\mathbf{x}, \mathbf{x}')$ jest definiowana jako odległość pomiędzy $\mathbf{x}$ i $\mathbf{x}'$ w zadanej metryce. Klasyfikacja k-NN polega na przydzieleniu wektorowi $\mathbf{x}$ etykiety, która występuje najczęściej wśród k najbliższych sąsiadów klasyfikowanego przypadku. W każdym punkcie przestrzeni cech określić możemy prawdopodobieństwo tego, że wektor $\mathbf{x}$ należy do klasy c_i

$$P(c_i|\mathbf{x}) = \frac{k_i}{k} \tag{1.7}$$

gdzie k_i jest liczbą wektorów z sąsiedztwa $\mathbf{x}$ o etykiecie c_i, zaś k liczbą branych pod uwagę sąsiadów. Klasyfikacja polega więc na wyborze klasy dla której to prawdopodobieństwo jest największe. Zdefiniowanie modelu sprowadza się do wyboru metryki i ilości k sąsiadów. Główną klasą metryk dla d-wymiarowych danych jest metryka Minkowskiego:

$$D(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^{\alpha} \right)^{\frac{1}{\alpha}} \tag{1.8}$$

Dla $\alpha = 2$ otrzymujemy najczęściej stosowaną metrykę - euklidesową a dla $\alpha = 1$ metrykę Manhattan. Liczbę sąsiadów wybieramy najczęściej nieparzystą aby uniknąć przypadków remisowych. Wybór ilości sąsiadów można zautomatyzować przeprowadzając serię testów dla różnych wartości k na zbiorze walidacyjnym i wybierając tę wartość dla której ilość pomyłek była najmniejsza. k-NN wymaga przechowywania w pamięci wszystkich wektorów treningowych. Za każdym razem gdy klasyfikujemy przypadek musimy przeszukać całą pamięć (złożoność obliczeniowa $O(dn)$) w poszukiwaniu najbliższych sąsiadów, co dla dużych zbiorów danych może być uciążliwe[3].

1.4.2. NRBF

NRBF (*ang. Normalized Radial Bases Function*) jest klasyfikatorem zainspirowanym sieciami neuronowymi z radialnymi funkcjami bazowymi (sieci RBF) choć jego działanie bliskie jest klasyfikatorom probabilistycznym, gdyż model za pomocą próbek treningowych próbuje oszacować rozkład prawdopodobieństwa $P(c_i|\mathbf{x})$. W miejscu każdego przypadku treningowego umieszczamy centrum funkcji Gaussa o ustalonym parametrze rozmycia σ

$$G(\mathbf{x}; \mathbf{x}_j; \sigma) = e^{-\frac{\|\mathbf{x}-\mathbf{x}_j\|^2}{\sigma^2}} \tag{1.9}$$

Jeśli przeprowadzimy normalizację względem przypadków z całego zbioru treningowego

$$\widetilde{G}(\mathbf{x}; \mathbf{x_t}, \sigma) = \frac{G(\mathbf{x}; \mathbf{x_t}; \sigma)}{\sum_{k=1}^{n} G(\mathbf{x}; \mathbf{x}_k; \sigma)} \tag{1.10}$$

tak aby

$$\sum_{i=1}^{n} \widetilde{G}(\mathbf{x}; \mathbf{x_i}, \sigma) = 1 \tag{1.11}$$

wówczas możemy interpretować wyrażenie

$$P(c_i|\mathbf{x}) = \sum_{\mathbf{x_t} \in \mathcal{T}_{c_i}} \widetilde{G}(\mathbf{x}; \mathbf{x_t}, \sigma) \tag{1.12}$$

jako prawdopodobieństwo przynależności $\mathbf{x}$ do klasy c_i, gdzie $\mathcal{T}_{c_i}$ jest podzbiorem zbioru treningowego zawierającym wszystkie obiekty oznaczone etykietą c_i[6].
Powyższy model jest uproszczoną wersją sieci neuronowej NRBF. Pomija ważenie wpływu węzłów zawierających funkcję Gaussa, ilość węzłów jest stała (równa ilości próbek) i nie ulega zmianie ich położenie. Jedynym parametrem, który pozostaje do ustalenia, jest σ. NRBF zachowuje się podobnie jak metoda k najbliższych sąsiadów, z tą różnica, że rozkład prawdopodobieństwa $P(c_i|\mathbf{x})$ w przypadku NRBF jest ciągły, gdzie dla k-NN ma on charakter dyskretny. Parametr k z metody kNN ma w przypadku NRBF odpowiednik w postaci σ, która decyduje o "zasięgu" funkcji Gaussa.

1.4.3. Ontogeniczna sieć neuronowa IncNet

Biologiczne sieci neuronowe stały się inspiracja do powstania matematycznych modeli przetwarzających informacje, nazywanych sztucznymi sieciami neuronowymi. Podstawowym elementem takiej sieci jest neuron, który realizuje pewną funkcję transferu, decydującą o pobudzeniu neuronu, czyli o sile sygnału powstałego na jego wyjściu, w zależności od sumy sygnałów docierających na wejścia neuronu. Sieci złożone z wielu połączonych ze sobą neuronów potrafią realizować bardzo skomplikowane odwzorowania, przez co znajdują szerokie zastosowanie w systemach klasyfikujących, aproksyujących i sterujących. Dobór architektury sieci neuronowej jest ściśle związany z problemem jaki chcemy za jej pomocą rozwiązać. Do klasyfikacji obiektów wykorzystywane są zwykle jednokierunkowe wielowarstwowe sieci neuronowe (MLP *ang. Multilayer Perceptron*), czyli sieci w których sygnał propaguje się od wejścia sieci do wyjścia bez sprzężeń zwrotnych. Sieć MLP z jedną warstwą ukrytą realizuje funkcję

$$y_i = \tilde{f}_i(\tilde{\mathbf{w}}^{\mathbf{t}}\mathbf{u} + \tilde{w}_0) \tag{1.13}$$

gdzie y_i jest odpowiedzią i-tego neuronu wyjściowego a wektor $\mathbf{u}$ jest sygnałem wyjściowym warstwy ukrytej o k-tej równej postaci

$$u_k = f_k(W\mathbf{x} + \mathbf{w_0}) \tag{1.14}$$

Uczenie sieci polega na odpowiedniej adaptacji wartości wag $W, \mathbf{w_0}, \tilde{\mathbf{w}}, \tilde{\mathbf{w}}_\mathbf{k}$ połączeń między warstwami, zgodnie z pewna regułą uczenia. Odpowiedni dobór funkcji transferu $f(\cdot)$, realizowanej przez neurony, może bardzo ułatwić problem. Najczęściej w sieciach MLP stosuje się funkcję sigmoidalną $\sigma(x) = \frac{1}{1-e^{ax}}$. Interesującą klasę modeli neuronowych stanowią sieci z radialnymi funkcjami transferu. Sieci RBF (*ang. Radial Basis Function*) realizują odwzorowanie

$$f(\mathbf{x}; \mathbf{w}) = \sum_i w_i \cdot G(\mathbf{x}; \mathbf{x_i}, \sigma) \tag{1.15}$$

gdzie funkcją transferu może być na przykład funkcja Gaussa (1.9). O sile sygnału wyjściowego neuronu decyduje odległość $\mathbf{x}$ od centrum $\mathbf{x_i}$ funkcji radialnej.

Sieci neuronowe z jedną warstwą ukrytą wystarczają do rozwiązania większości problemów aproksymacyjnych i klasyfikacyjnych - sieć posiadająca dwie warstwy ukryte stanowi uniwersalny aproksymator[9].

Incremental Network (IncNet) jest siecią jednowarstwową (architektura identyczna z sieciami RBF), w pełni ontogeniczną; rośnie, maleje i łączy neurony w trakcie uczenia, tak aby jak najlepiej dopasować swoją strukturę do złożoności danych treningowych [7]. Jako reguła uczenia zastosowany jest rozszerzony filtr Kalmana a funkcjami transferu najczęściej są funkcje bicentralne, utworzone z kombinacji funkcji sigmoidalnych

$$Bi(\mathbf{x};\mathbf{t},\mathbf{b},\mathbf{s}) = \prod_{i=1}^{n} \sigma(e^{s_i}(x_i - t_i + e^{b_i}))(1 - \sigma(e^{s_i}(x_i - t_i - e^{b_i}))) \tag{1.16}$$

Funkcje takie charakteryzują się dużą elastycznością i pozwalają estymować złożone powierzchnie decyzyjne przy użyciu niewielkiej liczby węzłów[8].

1.4.4. Drzewo decyzyjne SSV

Drzewa decyzyjne zbudowane są z węzłów, realizujących testy na cechach wektorów w ten sposób, że każdy z takich testów dzieli przypadki na grupy, próbując rozseparować z jak największa dokładnością wektory należące do różnych klas. Każdy z kolejnych testów, dzieli przypadki na coraz mniejsze grupy, aż do momentu całkowitego rozseparowania wzorców z różnych klas. Końcowe odgałęzienia (liście drzewa) skojarzone są z poszczególnymi etykietami klas. Budowanie drzewa (uczenie modelu) przeprowadzane jest według pewnego kryterium, które stara się ocenić przydatność danego podziału - w najprostszym przypadku obliczany jest błąd klasyfikacji dla danego podziału i wybierany jest ten, któremu odpowiada najmniejszy błąd. Rozrośnięte drzewa należy przyciąć - sytuacja w której każdy liść zawiera pojedyńczy przypadek nie zapewnia dobrej generalizacji. Klasyfikacja wzorca polega na podążaniu od korzenia drzewa do odpowiedniego liścia zgodnie przesłankami spełnianymi (bądź nie) w poszczególnych węzłach.

Kryterium podziału SSV (*ang. Separability of Split Value*) stara się rozdzielić jak największą ilość par obiektów pochodzących z różnych klas. Sprowadza się to do poszukiwania maksimum wyrażenia

$$\begin{aligned} SSV(s) = 2 \sum_{c \in \mathcal{C}} |LS(s,b,\mathcal{T}) \cap \mathcal{T}_c| \cdot |RS(s,b,\mathcal{T}) \cap (\mathcal{T} - \mathcal{T}_c)| \\ - \sum_{c \in \mathcal{C}} min(|LS(s,b,\mathcal{T}) \cap \mathcal{T}_c|, |RS(s,b,\mathcal{T}) \cap \mathcal{T}_c|) \end{aligned} \tag{1.17}$$

gdzie s jest wartością podziału cechy b (dla ciągłych wartości jest to punkt, dla nominalnych - podzbiór wartości cechy), $LS(s,f,\mathcal{T})$ i $RS(s,f,\mathcal{T})$ to lewa i prawa strona uzyskane po podziale cechy b dla danego zbioru $\mathcal{T}$[4].

1.4.5. System neurorozmyty FSM

Feature Space Mapping jest siecią neuronową, o architekturze podobnej do sieci RBF, choć w działaniu przypomina systemy poszukujące reguł logicznych w danych - w ogólności reguł logiki rozmytej. Model neuronowy FSM składa się z jednej warstwy neuronów ukrytych z lokalnymi faktoryzowalnymi funkcjami transferu

$$G(\mathbf{x};\mathbf{r},\sigma) = \prod_{t} G_t(x_i; r_i, \sigma_i) \tag{1.18}$$

Dzięki zastosowaniu takich funkcji każdy wymiar możemy rozpatrywać oddzielnie. Jeśli $G_i(x_i; r_i, \sigma)$ zdefiniujemy za pomocą funkcji Gaussa (1.9) wówczas neurony realizują w każdym wymiarze funkcje przynależności $G_i : \mathcal{X} \to [0, 1]$, określającą w jakim stopniu dany przypadek możemy kojarzyć z danym neuronem (czyli z centrum funkcji Gaussa w pobliżu którego położone są wartości odpowiednich cech badanych wektorów). Klasyfikacja wektora $\mathbf{x}$ polega na wyborze najbardziej pobudzonego neuronu, któremu przypisana jest odpowiednia etykieta klasy

$$FSM(\mathbf{x}) = C(\arg\max_i(G_i(\mathbf{x}; \mathbf{r}, \sigma))) \tag{1.19}$$

Iloczyn (1.18) możemy rozumieć jako regułę logiczną, wówczas wektor $\mathbf{x}$ jest kojarzony z neuronem (i odpowiednią klasą) gdy spełnia odpowiadająca mu regułę. Innymi funkcjami faktoryzowalnymi pozwalającymi na ekstrakcję reguł rozmytych, stosowanymi w sieci FSM, są funkcje bicentralne (**??**) lub funkcje trapezoidalne. FSM może także realizować klasyczne reguły logiczne jeśli zastosujemy jako funkcje transferu funkcję prostokątną. Wstępna klasteryzacja danych wykożystywana jest do ustalenia ilości węzłów oraz inicjalizacji położeń centrów funkcji transferu, które są optymalizowane w późniejszym procesie uczenia.

1.4.6. SVM

Jeśli funkcje dyskryminującą zdefiniujemy w postaci

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 \tag{1.20}$$

wówczas powierzchnie decyzyjne rozdzielające klasy będą płaszczyznami. Takie podejście nazywamy liniową dyskryminacja (LDA)[3]. Możliwych płaszczyzn rozdzielających dwie sąsiednie klasy może być wiele - najlepszą generalizację powinna zapewniać płaszczyzna z największym marginesem. Marginesem nazwiemy nieujemną wartość b spełniającą

$$|g(\mathbf{x}_i) - b| \geqslant 0, \qquad \forall \mathbf{x}_i \in \mathcal{T} \tag{1.21}$$

Przy określeniu tej granicy decydujący wpływ mają tylko niektóre wektory, nazywane wektorami wspierającymi (*ang. support vectors*), dla których $g(\mathbf{x}) = b$. LDA nie daje dobrych rezultatów w przypadkach gdy mamy do czynienia z problemem, w którym klasy nie są separowalne liniowo.

Maszyna wektorów wspierających SVM (*ang. Support Vector Machine*) opiera się na tym samym pomyśle co linowa dyskryminacja z maksymalizacją marginesu z tym, że przypadki nie muszą być separowalne w przestrzeni wejściowej, lecz w pewnej wyżej wymiarowej przestrzeni [3][11]. Zwiększenie wymiaru przestrzeni zwiększa jej pojemność liniową, tzn. zawsze istnieje przestrzeń w której, po zmapowaniu do niej wektorów za pomocą pewnej nieliniowej transformacji $\varphi(\cdot)$, dowolne dwie klasy będą separowalne za pomocą hiperpłaszczyzny. Niech $\mathbf{y_k} = \varphi(\mathbf{x_k})$ będzie przypadkiem przetransformowanym do wielowymiarowej przestrzeni, wówczas najbardziej optymalnej hiperpłaszczyzny rozdzielającej dwie klasy poszukujemy minimalizując wyrażenie

$$L(\alpha) = \sum_{k=1}^{n} \alpha_k - \frac{1}{2}\sum_{k,j}^{n} \alpha_k\alpha_j z_k z_j \mathbf{y}_j^t\mathbf{y}_k \tag{1.22}$$

z więzami

$$\sum_{k=1}^{n} z_k\alpha_k = 0 \qquad \alpha_k \geqslant 0, \quad k = 1, \ldots, n \tag{1.23}$$

gdzie $z_k = \pm 1$ zależy od klasy do której należy k-ty przypadek. Funkcja $\varphi(\cdot)$ określona jest przez iloczyn skalarny

$$\mathbf{y}_j^t \mathbf{y}_k = \varphi(\mathbf{x}_j)^t \varphi(\mathbf{x}_k) = k(\mathbf{x}_j, \mathbf{x}_k) \tag{1.24}$$

gdzie $k(\mathbf{x}_j, \mathbf{x}_k)$ nazywane jest funkcją jądra i najczęściej wybierane w postaci funkcji gaussowskiej (1.9). Funkcja dyskryminacyjna przybiera teraz postać

$$g(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i) + w_0' \tag{1.25}$$

Wektory wspierające w tym przypadku to wektory dla których $\alpha_k > 0$.

1.5. Podsumowanie

W powyższym rozdziale omówione zostały ogólnie najczęściej stosowane metody klasyfikacji. Z punktu widzenia pracy najistotniejsze wydają się metody korzystające z przykładów treningowych bezpośrednio podczas klasyfikacji, takie jak k-NN oraz NRBF. Dla tych metod najłatwiej jest zaplanować strategię wyboru wektorów i przewidzieć jej wpływ na skuteczność działania modelu a zredukowanie ilości wektorów prowadzi w tym przypadku do zmniejszenia kosztu obliczeniowego i pamięciowego procesu klasyfikacji.

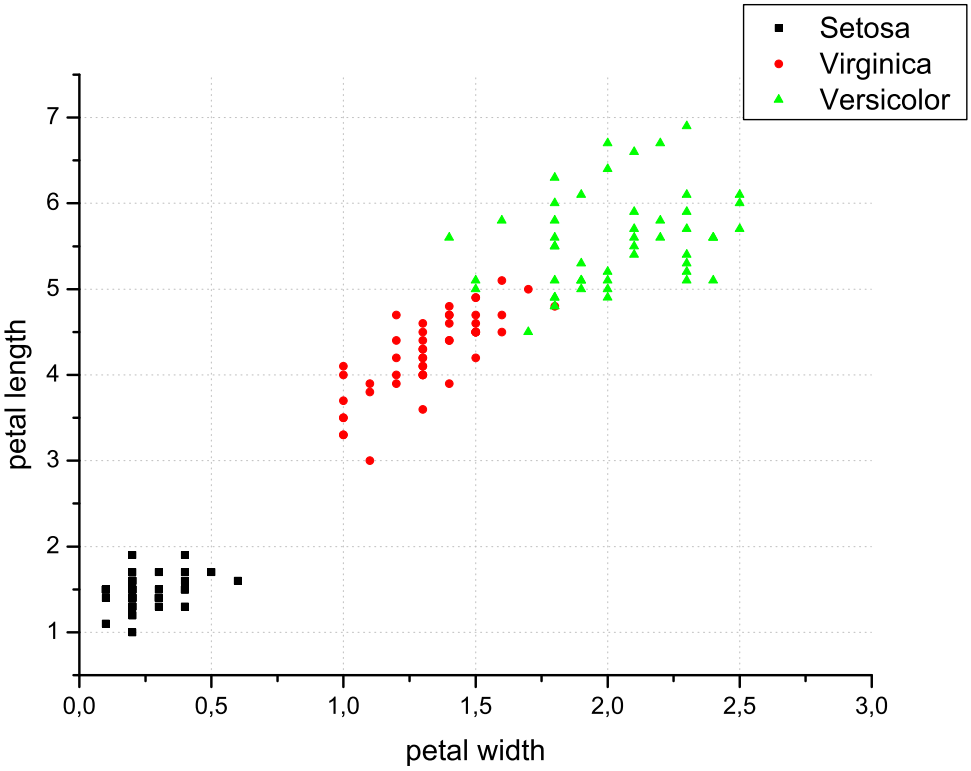
2. Strategie szukania wektorów referencyjnych

2.1. Zdefiniowanie problemu

Szukamy optymalnie małego zbioru $\mathcal{S}$ przypadków reprezentujących tą samą populację co próbki treningowe $\mathcal{T}$, zapewniającego poprawną klasyfikację zarówno przypadków ze zbioru $\mathcal{T}$ jak i nowych przypadków, z błędem nie większym (lub niewiele większym) niż w wyniku uczenia klasyfikatora na całym dostępnym zbiorze treningowym. Innymi słowy, chcemy przeprowadzić selekcję wektorów ze zbioru danych treningowych, pozostawiając w nim tylko wektory najbardziej użyteczne, pozbywając się wektorów mało wartościowych lub szkodliwych w sensie późniejszej klasyfikacji. Jeśli szukany zbiór ma być podzbiorem zbioru treningowego ($\mathcal{S} \subseteq \mathcal{T}$), wówczas dla zbioru $\mathcal{T}$ zawierającego n przypadków problem sprowadza się do znalezienia najbardziej optymalnego z $2^n - 1$ możliwych podzbiorów. Możemy też wygenerować nowe przypadki - wektory wirtualne, które w ogólności mogą być dowolnymi punktami w badanej przestrzeni $\mathcal{R}^d$. Złożoność kombinatoryczna problemu zmusza do stosowania różnych heurystyk poszukujących kompromisu pomiędzy maksymalną redukcja ilości wektorów treningowych a zachowaniem dokładności klasyfikacji.

2.2. Struktura zbioru danych

Tak jak nie znaleziono klasyfikatora, który radziłby sobie równie dobrze ze wszystkimi problemami klasyfikacyjnymi, tak też możemy spodziewać się, że kryterium poszukiwania wektorów referencyjnych także powinno być dopasowane do struktury danego problemu. Duża część problemów charakteryzuje się tym, że daną klasę możemy zdefiniować przez obszary o jednorodnym rozkładzie przypadków oznaczonych tą samą etykietą. Przykładem takiego zbioru danych jest Iris (rys.2.1). Dla takich danych możemy mówić o centrach klas i granicach pomiędzy nimi, nie trudno wyróżnić wzorce odstające od danych, leżące daleko do swoich klas lub położone w głębi regionów zajętych przez przypadki z innej klasy, które możemy interpretować jako błędne, stanowiące szum w danych. Zastanówmy się, które wektory w takim przypadku mogą mieć duże znaczenie w klasyfikacji a które mogą być szkodliwe. Usunięcie wektorów leżących wewnątrz jednorodnej klasy nie powinno wpłynąć na dokładność klasyfikacji - granice decyzyjne nadal będą określone przez pozostałe wektory położone bliżej granic. Wydaje się więc, że wektorów referencyjnych powinniśmy poszukiwać w obszarach, na których graniczą ze sobą wektory z różnych klas. Z drugiej strony, gdy usuniemy ze zbioru wektory położone w bezpośrednim sąsiedztwie wektorów z klas przeciwnych, spowodujemy "wygładzenie" granic decyzyjnych, co może polepszyć generalizację, zmniejszając ryzyko przeuczenia się modelu. Możemy posunąć się jeszcze dalej i poszukać prototypów, które reprezentowałyby grupy przypadków, klastry a nawet całe klasy. Istnieją jednak problemy, w których klasy nie są zlokalizowane w grupach położonych w określonych obszarach przestrzeni cech. W najbardziej ekstremalnym przy-



Rysunek 2.1. Wizualizacja zbioru Iris względem dwóch cech określających szerokość i długość płatka

padku wektory z danej klasy mogą być rozrzucone po całej badanej przestrzeni i trudno jest stwierdzić, który przypadek jest ważny a który może być błędny. Na przykład, gdy mamy do czynienia z danymi, których cechy posiadają wartości symboliczne - których nie sposób uporządkować monotonicznie, nie możemy spodziewać się, że dane będą tworzyły grupy przypadków z jednakową etykietą zlokalizowane w określonych miejscach przestrzeni wejściowej. Każdy wektor ze zbioru treningowego może wówczas decydować o kształcie granicy decyzyjnej i problem wyboru wektorów referencyjnych staje się stosunkowo trudny. Brighton i Mellish[2] stwierdzają, że w takich przypadkach najlepiej spisują się metody szukające prototypów w całej dostępnej przestrzeni.

Powyższe rozumowanie dotyczy głównie metod opartych na podobieństwie. Dla kNN usunięcie jednego wektora ma wpływ na klasyfikację przypadków wyłącznie z pewnego sąsiedztwa usuniętego. W przypadku bardziej złożonych modeli takie postępowanie może nie posiadać lokalnego charakteru. Trudno wywnioskować ze struktury danych, bez wnikania w zawiłości działania danego modelu, które wektory są najbardziej istotne na przykład dla drzewa decyzyjnego lub dla metody SVM.
Niewątpliwie struktura danych ma wpływ na wybór strategii poszukiwania wektorów referencyjnych - w różnych problemach różne przypadki mogą odgrywać istotną rolę. Dobry algorytm poszukujący wektorów referencyjnych powinien poradzić sobie z każdym typem danych - oczywiście jeżeli dane są przygotowane dobrze i wiarygodnie reprezentują dany problem.

2.3. Przegląd stosowanych metod

Poniżej znajduje się opis kilkunastu metod stosowanych do wyboru wektorów uczących. Działanie większości z nich opiera się na badaniu zależności między przypadkami a ich sąsiedztwem, przez co naturalnym ich zastosowaniem jest poszukiwanie wektorów referencyjnych dla kNN. Nic nie stoi jednak na przeszkodzie, aby przynajmniej część z nich spróbować uogólnić, aby znalazły zastosowanie również dla innych modeli klasyfikujących.

2.3.1. Filtry szumu

Filtrami szumu[1] nazwiemy metody usuwające ze zbioru treningowego przypadki, które możemy uważać za szkodliwe dla klasyfikatora, czyli wektory błędne lub odstające od struktury danych. Spodziewamy się, że takie postępowanie podniesie jakość danych i poprawi wyniki klasyfikacji. Redukcja objętości zbioru uczącego jest tu stosunkowo niewielka i dlatego metody te często są stosowane jako wstępna filtracja błędnych przypadków z danych przed dalszą selekcją za pomocą innych algorytmów. Filtry szumu sprawdzają się, gdy zbiór treningowy nie zawiera dużo przypadków, które możemy uznać za błędne. Trudności mogą wystąpić, gdy poziom szumu w danych jest wysoki i takie wektory przestają być wyjątkami - wówczas jedne błędne wektory mogą być dobrze klasyfikowane przez inne błędne przypadki. W niektórych zbiorach danych niemożliwe jest stwierdzenie czy dany przypadek jest błędny (np. gdy dane z różnych klas w dużym stopniu się pokrywają) dlatego często niezbędna jest wiedza dotycząca danego zbioru aby ocenić czy możemy w tym przypadku zastosować dany algorytm.

ENN

Wilson w 1972 roku zaproponował prostą regułę edycji danych, według której usuwamy ze zbioru wektor jeśli jego klasa nie zgadza się z większością jego najbliższych k sąsiadów ze zbioru oryginalnego T. W ten sposób pozbywamy się wektorów otoczonych przez przypadki z innej klasy (jest to zazwyczaj "szum") oraz przypadków położonych w pobliżu granic klas (wygładzamy granice decyzyjne), pozostawiając dane wewnątrz obszarów zajętych przez daną klasę (rys.2.2). Złożoność czasu algorytmu jest równa złożoności testu *leave one out* $O(dn^2)$, czyli klasyfikacji każdego wektora ze zbioru przez pozostałe za pomocą k-NN (Wilson używał k=3). Powyższy algorytm w literaturze funkcjonuje jako zasada edycyjna Wilsona lub ENN (*ang. Edited Nearest Neighbor*)[2][13].

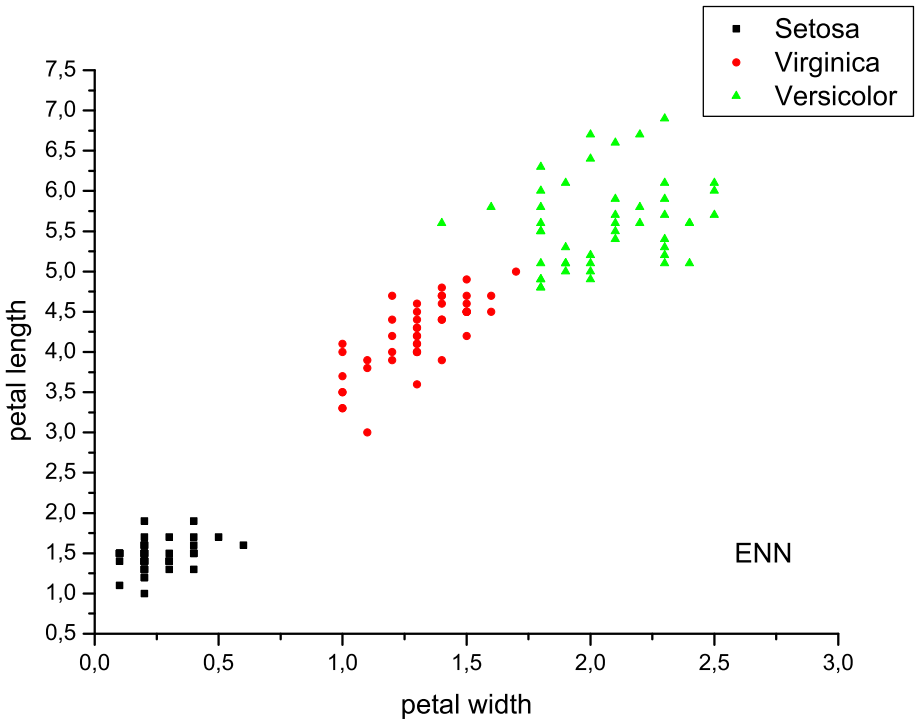
RENN i All kNN

Algorytmy RENN i All kNN są kolejnymi modyfikacjami metody edycyjnej Wilsona. RENN (*ang. Repeated ENN*) powtarza ENN wielokrotnie, aż do zbieżności, czyli do momentu, gdy podczas całej epoki nie zostanie usunięty żaden wektor. Podobne podejście odnaleźć można w algorytmie All k-NN, który polega na powtarzaniu ENN za każdym razem zwiększając ilość branych pod uwagę sąsiadów ($k = 1, 2, ..., l$).

Kryterium spójności - ENRBF

Analogiczną metodę edycji można zaproponować korzystająca z klasyfikatora NRBF. Niech $P(c_i|\mathbf{x}, \mathcal{S})$ określa prawdopodobieństwo, że wektor $\mathbf{x}$ należy do klasy c_i dla zbioru danych $\mathcal{S}$. Jeśli zbiór $\mathcal{S}$ jest dostatecznie gęsty, wtedy usunięcie dowolnego wektora

[1] W literaturze metody te często występują pod nazwą metod edycyjnych (*ang. editing rules*)



Rysunek 2.2. Rezultat działania ENN dla danych Iris

$\mathbf{x}_i$ nie powinno zmieniać w znaczny sposób rozkładu prawdopodobieństwa $P(c_i|\mathbf{x}_i, \mathcal{S}) \approx P(c_i|\mathbf{x}_i, \mathcal{S}^i)$, gdzie $\mathcal{S}^i = \mathcal{S} - \{\mathbf{x}_i, C_i\}$. W przypadku gdy

$$P(c_i|\mathbf{x}_i, \mathcal{S}^i) \ll P(c_j|\mathbf{x}_i, \mathcal{S}^i) \qquad \text{dla każdego } j \neq i \tag{2.1}$$

możemy powiedzieć, że $\mathbf{x}_i$ jest niezgodny (niespójny) ze zbiorem danych $\mathcal{S}$ i możemy go pominąć przy uczeniu klasyfikatora[6]. Zdefiniujmy prawdopodobieństwo $P(c_i|\mathbf{x}, \mathcal{S})$ za pomocą unormowanych funkcji Gaussa (1.10). Reguła edycyjna będzie miała postać: usuń przypadek z $\mathcal{T}$ dla którego zachodzi

$$P(c_i|\mathbf{x}_i, \mathcal{T}^i) < \alpha P(c_j|\mathbf{x}_i, \mathcal{T}^i) \qquad \text{dla każdego } j \neq i. \tag{2.2}$$

Parametr $\alpha \in [0, 1]$ decyduje o ostrości kryterium, czyli o ilości usuniętych wektorów. Dla $\alpha = 0$ żaden wektor nie zostanie usunięty - filtr przepuszcza wszystkie dane. Dla $\alpha = 1$ usunięte zostaną wszystkie przypadki sporne, w których otoczeniu większość przypadków należy do przeciwnej klasy oraz wektory z pobliża granic. Powoduje to wygładzenie granic i rozseparowanie klas. Gdybyśmy zdefiniowali $P(c_i|\mathbf{x}, \mathcal{S})$ zgodnie z prawdopodobieństwem klasyfikacji k-NN (1.7), powyższe kryterium dla $\alpha = 1$ staje się identyczne z modelem ENN. ENRBF (*ang. Edited NRBF*) usuwa wektory niespójne stosując powyższa regułę dla każdego przypadku z $\mathcal{T}$ ($O(dn^2)$).

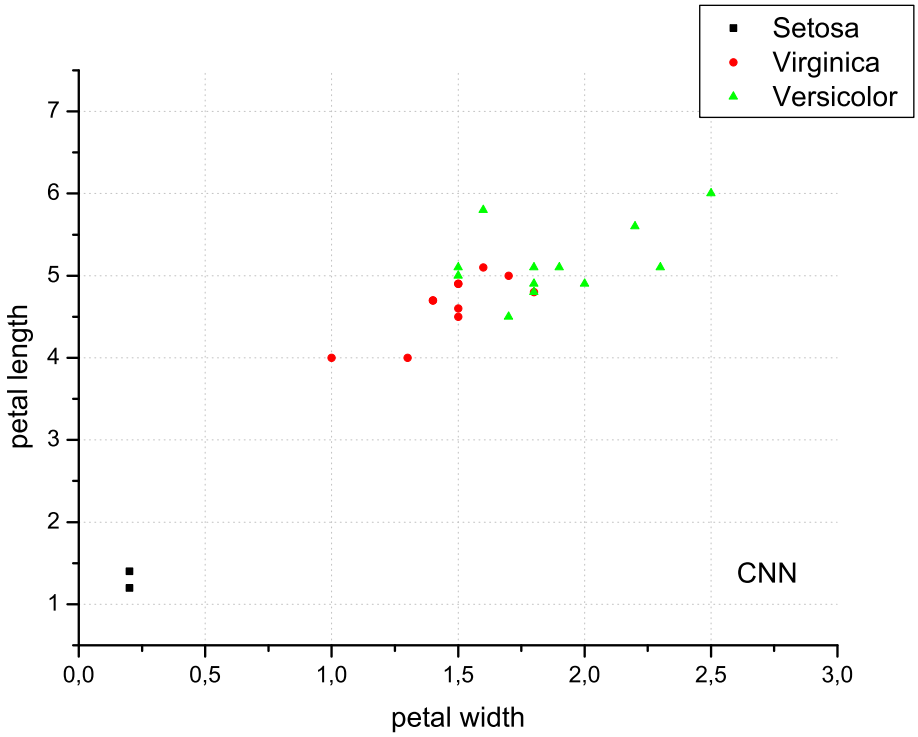
2.3.2. Metody kondensujące dane

Metodami kondensującymi nazwiemy metody, których głównym celem jest maksymalne zmniejszenie objętości zbioru treningowego bez zwiększenia błędu klasyfikacji. Poszu-

kują one w zbiorze treningowym przypadków, których wkład do nauki klasyfikatora jest znikomo mały, przez co niepotrzebnie zwiększają koszt czasu i pamięci a ich usunięcie nie powoduje pogorszenia działania modelu klasyfikującego.

CNN

W 1968 roku Hart zaproponował algorytm CNN (*ang. Condensed Nearest Neighbor*), który poszukuje tzw. minimalnego spójnego podzbioru zbioru treningowego. Podzbiorem spójnym zbioru $\mathcal{T}$ według Harta nazywamy zbiór $\mathcal{S}$, który pozwala na klasyfikację wszystkich wektorów z $\mathcal{T}$ z dokładnością nie mniejszą niż w przypadku, gdy jako zbiór uczący użyjemy oryginalnego zbioru $\mathcal{T}$[2]. CNN jest algorytmem inkrementacyjnym: rozpoczyna poszukiwanie od pustego zbioru $\mathcal{S}$ do którego w sposób losowy dodajemy po jednym wektorze z każdej klasy. Kolejne wektory z $\mathcal{T}$ klasyfikujemy przy użyciu wektorów z $\mathcal{S}$ (Hart stosował k-NN z $k = 1$). Jeśli $\mathbf{x}$ zostanie błędnie zaklasyfikowany wówczas dodajemy go do $\mathcal{S}$. Powtarzamy całą procedurę, aż do momentu gdy wszystkie przypadki z $\mathcal{T}$ będą poprawnie klasyfikowane przez wektory z $\mathcal{S}$. W rezultacie, każdy wektor z $\mathcal{T}$ położony jest w mniejszej odległości od przypadku z tej samej klasy w $\mathcal{S}$ niż od dowolnego wektora z innej klasy w tym zbiorze (rys.2.3). Algorytm nie gwarantuje znalezienia minimalnego



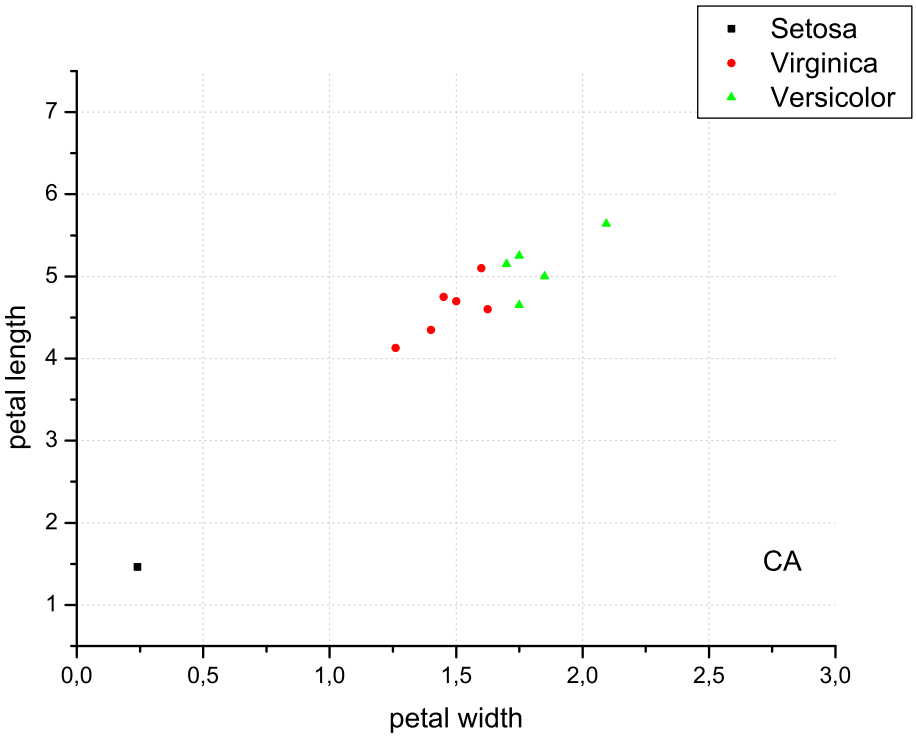
Rysunek 2.3. Rezultat działania CNN dla danych Iris

zbioru spójnego, między innymi dlatego, że otrzymany wynik zależy od kolejności prezentacji przypadków. Poza tym wektory, które podejrzewamy, że są błędne (takie wektory posiadają większość sąsiadów z przeciwnej klasy) zawsze znajdą się w zbiorze $\mathcal{S}$, co z kolei powoduje, że wektory będące ich sąsiadami także muszą się znaleźć w szukanym zbiorze

aby mogły być dobrze klasyfikowane[13].
Identyczne kryterium poszukiwania wektorów referencyjnych zaproponował Aha pod nazwą IB2 ograniczając się do przeprowadzenia jednej epoki po wszystkich przypadkach z $\mathcal{T}$ i w rezultacie nie gwarantując tym samym tego, że otrzymamy zbiór spójny względem zbioru treningowego.

CA

Chang w 1974 roku zaproponował algorytm, który szuka w zbiorze treningowym par wektorów w celu ich połączenia w jeden przypadek. Dwa wektory $\mathbf{p}$ i $\mathbf{q}$ możemy połączyć w jeden gdy oba należą do tej samej klasy oraz jeśli takie połączenie nie "popsuje" spójności zbioru, tzn. nie zwiększy błędu klasyfikacji przypadków w $\mathcal{T}$ za pomocą wektorów referencyjnych w $\mathcal{S}$. W najprostszym przypadku wektor powstały po połączeniu umieszczamy w środku prostej łączącej dwa rozpatrywane wektory w przestrzeni cech. Procedurę powtarzamy, aż do momentu w którym, podczas całej epoki nie zostanie połączona żadna para wektorów (rys.2.4). Duża złożoność czasowa $O(dn^3)$ sprawia, iż algorytm nie



Rysunek 2.4. Rezultat działania CA dla danych Iris

znajduje zastosowania dla dużych baz danych, choć jego ciekawą właściwością jest to, iż przestrzeń poszukiwań wektorów nie ogranicza się tylko do położeń wektorów ze zbioru treningowego[10].

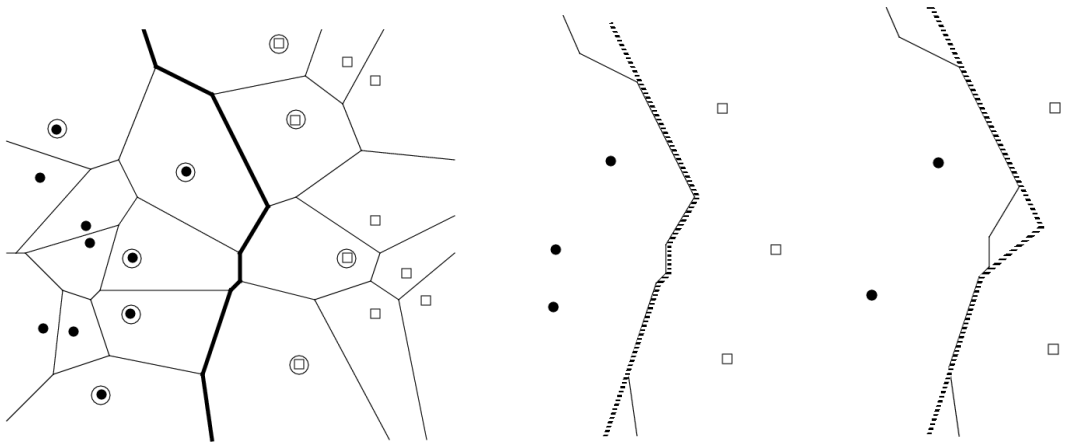
RNN

Algorytm CNN doczekał się wielu modyfikacji, jedną z nich, o której warto wspomnieć, jest dekremencyjna wersja tego algorytmu. Szukając minimalnego zbioru spójnego

możemy odrzucać kolejno wektory z $\mathcal{T}$, testując za każdym razem wpływ takiego postępowania na zachowanie spójności. Jeżeli usunięcie wektora spowoduje wzrost błędu klasyfikacji policzonego dla wszystkich przypadków z $\mathcal{T}$, wówczas taki przypadek wydaje się istotny, dlatego powinniśmy go zachować. W ten sposób, w oparciu o klasyfikację k-NN, działa algorytm RNN (*ang. Reduced Nearest Neighbor*). Jednak takie podejście jest bardzo kosztowne - każde usunięcie wektora musi być uzasadnione testem klasyfikacji na całym zbiorze treningowym[13].

GA i RNGE

W przypadku klasyfikatora minimalnoodległościowego (1-NN) widać, że kształt granicy decyzyjnej jest zdeterminowany wyłącznie przez wektory leżące w bezpośrednim sąsiedztwie swoich "wrogów", czyli wektorów posiadających inną etykietę od przypadku badanego. Jeśli wyznaczymy dla zbioru treningowego diagram Voronoi (rys.2.5a) i oddzielimy obszary w diagramie należące do różnych klas, wówczas otrzymamy powierzchnie odpowiadające powierzchniom decyzyjnym klasyfikatora 1-NN. Zauważmy, że jeśli usu-

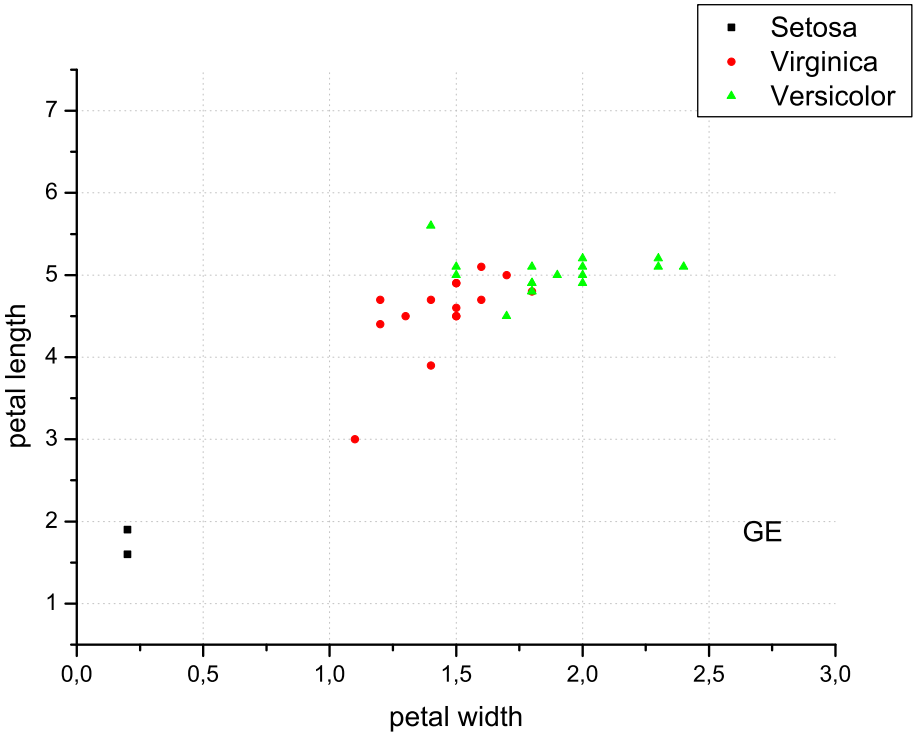


(a) (b) (c)

Rysunek 2.5. Wpływ selekcji wektorów na kształt granicy decyzyjnej klasyfikatora 1-NN przy zastosowaniu edycji sąsiadów z (a) diagramu Voronoi, (b) grafu Gabriela oraz (c) grafu RNG[1]

niemy wektory, które w diagramie sąsiadują wyłącznie z wektorami należącymi do tej samej klasy, granica nie ulegnie zmianie, a co za tym idzie, zachowana zostanie spójność danych względem zbioru treningowego. Poza tym zawartość otrzymanego zbioru nie zależy od kolejności prezentacji wzorców. Niestety, złożoność obliczeniowa algorytmu, zwłaszcza dla wielowymiarowych danych, jest duża. Wyznaczenie diagramu Voronoi może być w najgorszym przypadku rzędu $O(n^{d/2})$), co zmusza do stosowania prostszych metod. Jedną z alternatyw jest poszukiwanie sąsiadów w grafie Gabriela lub w grafie relatywnym[1].
Graf Gabriela dla danego zbioru $\mathcal{D}$ definiujemy w następujący sposób: dla każdej pary punktów (p, q) w zbiorze $\mathcal{D}$, tworzymy sferę $S(p, q)$ tak aby punkty p i q leżały na końcu średnicy sfery. Punkty p i q nazwiemy sąsiadami w grafie Gabriela jeśli wewnątrz takiej sfery nie występuje żaden inny punkt ze zbioru $\mathcal{D}$. Graf Gabriela utworzony jest z odcinków łączących wszystkie takie pary. Algorytm edycyjny GE (*ang. Gabriel Editing*) będzie wyglądał tak samo jak w przypadku diagramu Voronoi - usuwamy ze zbioru wektory, które w grafie Gabriela sąsiadują wyłącznie z wektorami z tej samej klasy. Otrzymany w ten

sposób zbiór będzie zawsze podzbiorem zbioru otrzymanego za pomocą diagramu Voronoi (każda para punktów sąsiadująca ze sobą w grafie Gabriela jest też sąsiadującą parą w grafie Voronoi) w związku z tym nie zostaje zachowana całkowicie granica decyzyjna 1-NN a otrzymany zbiór nie jest już spójny względem zbioru oryginalnego (rys.2.6). Deformacji



Rysunek 2.6. Rezultat działania GE dla danych Iris

ulegają fragmenty powierzchni decyzyjnej położone na krańcach regionów oddzielających dwie klasy (rys.2.5b). Złożoność algorytmu GE jest rzędu $O(dn^3)$ - dla każdej pary $\mathbf{x_p}$ i $\mathbf{x_q}$ ze zbioru treningowego $\mathcal{T}$ sprawdzamy czy

$$d^2(\mathbf{x_p}, \mathbf{x_q}) > d^2(\mathbf{x_p}, \mathbf{x_k}) + d^2(\mathbf{x_q}, \mathbf{x_k}) \qquad \text{dla każdego } k \neq q \neq p \tag{2.3}$$

Jeszcze większą selekcję otrzymamy stosując wersję algorytmu z wykorzystaniem grafu relatywnego sąsiedztwa RNG (*ang. Relative Neighborhood Graph*). Dwa punkty $\mathbf{x_p}$ i $\mathbf{x_q}$ definiujemy jako relatywnie bliskie jeśli

$$d(\mathbf{x_p}, \mathbf{x_q}) \geqslant max[d(\mathbf{x_p}, \mathbf{x_k}), d(\mathbf{x_q}, \mathbf{x_k})] \qquad \text{dla każdego } k \neq q \neq p \tag{2.4}$$

Graf powstaje w wyniku połączenia wszystkich takich punktów. Algorytm selekcji RNGE (*ang. RNG Editing*) daje w rezultacie podzbiór zbioru otrzymanego za pomocą algorytmu GE, co prowadzić może do jeszcze większej degeneracji granicy decyzyjnej (rys.2.5c).

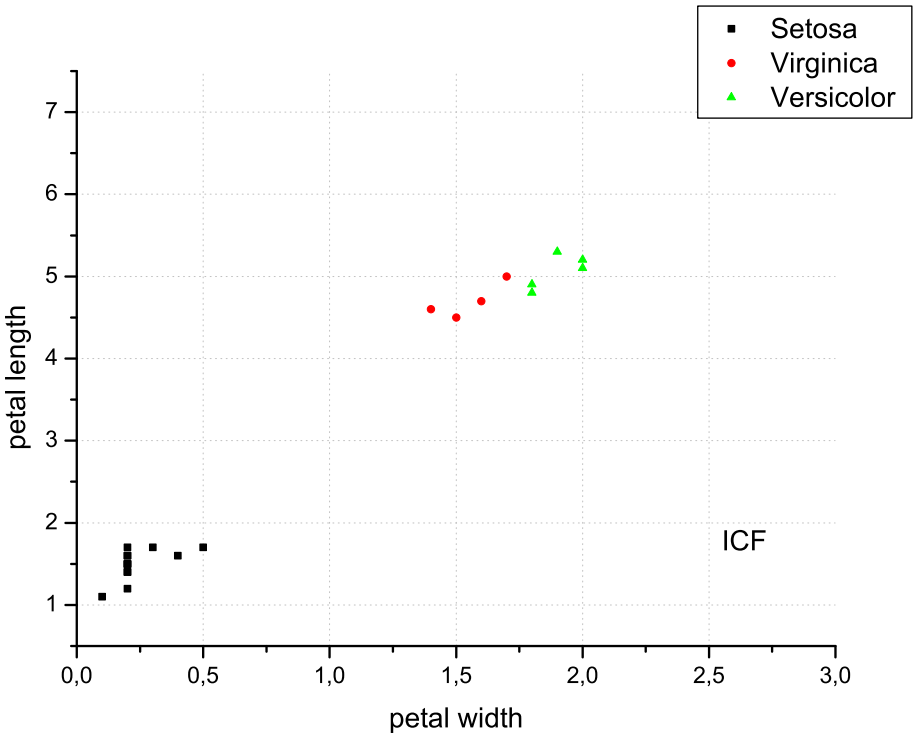
ICF

Kolejna metoda poszukuje wektorów położonych w centrach klas i klastrów zgodnie z heurystyką mówiącą, że takie wektory są dobrze reprezentowane przez ich otoczenie.

Zdefiniujmy *zbiór lokalny* $\mathcal{L}(\mathbf{x})$ wektora $\mathbf{x}$ jako zbiór wszystkich przypadków znajdujących się wewnątrz największej możliwej hipersfery o środku w punkcie $\mathbf{x}$ zawierającej wyłącznie przypadki z klasy tej samej co $\mathbf{x}$. Tak więc, promień hipersfery jest odległością $\mathbf{x}$ od najbliższego przypadku pochodzącego z innej klasy. Wektory posiadające liczebne zbiory lokalne otoczone są przez dużą liczbę przypadków należących do tej samej klasy, więc możemy je usunąć bez obawy pogorszenia klasyfikacji. Problemem stanowi ustalenie momentu w którym powinniśmy zaprzestać usuwania kolejnych wektorów. Brighton i Mellish [2] zaproponowali algorytm ICF (*ang. Iterative Case Filtering Algorithm*) w którym dla każdego z przypadków określamy zbiory:

$$\begin{aligned} \mathcal{C}(\mathbf{x}) &= \{\mathbf{x}' \in \mathcal{T} : \mathbf{x} \in \mathcal{L}(\mathbf{x}')\} \\ \mathcal{R}(\mathbf{x}) &= \{\mathbf{x}' \in \mathcal{T} : \mathbf{x}' \in \mathcal{L}(\mathbf{x})\} \end{aligned} \tag{2.5}$$

Zbiór $\mathcal{C}(\mathbf{x})$ jest równoważny zbiorowi lokalnemu wektora $\mathbf{x}$, zaś $\mathcal{R}(\mathbf{x})$ to zbiór wektorów dla których $\mathbf{x}$ znajduje się w ich zbiorze lokalnym. Zaproponowana zasada edycji polega na usunięciu wektora $\mathbf{x_i}$ jeżeli rozmiar zbioru $\mathcal{R}(\mathbf{x_i})$ jest większy od rozmiaru zbioru $\mathcal{C}(\mathbf{x_i})$. Powyższe kryterium sprawdzane jest dla wszystkich wektorów z $\mathcal{S}$ (na początku $\mathcal{S} = \mathcal{T}$) w każdej iteracji algorytmu ICF, aż do momentu, gdy po kolejnej iteracji nie zostanie usunięty żaden przypadek. W rezultacie otrzymujemy zbiór wektorów, które położone są w pobliżu granic decyzyjnych (rys.2.7). Można zauważyć, że wektory stanowiące szum



Rysunek 2.7. Rezultat działania ICF dla danych Iris

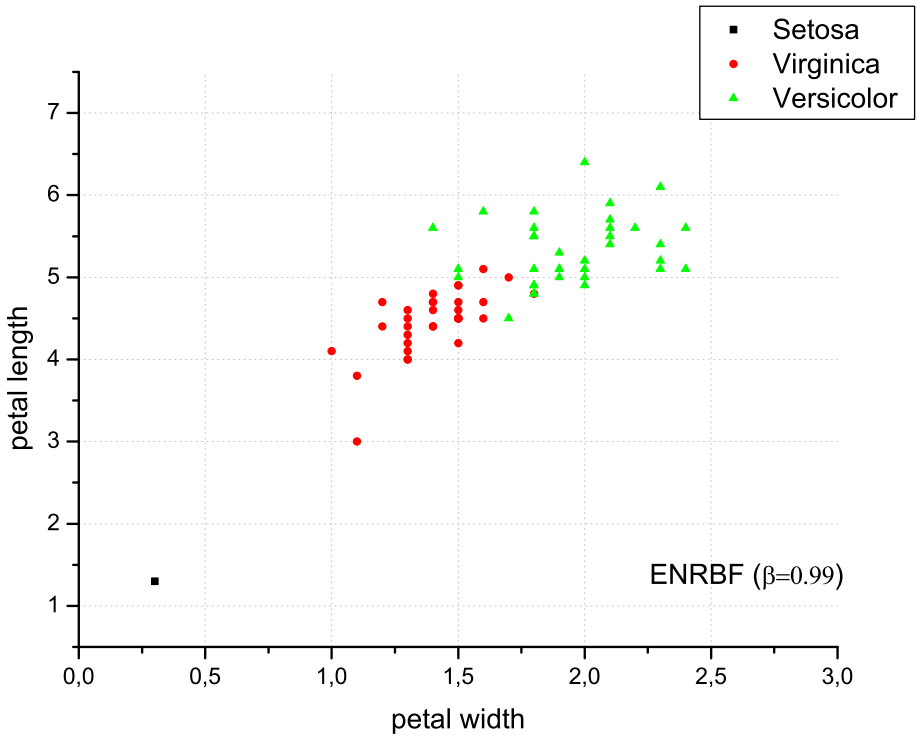
(otoczone przypadkami z innej klasy) są jedynymi elementami swoich zbiorów $\mathcal{R}$ i $\mathcal{C}$, dlatego aby uchronić algorytm przed zachowywaniem takich przypadków, we wstępnej fazie stosujemy filtr szumu - metodę ENN.

ENRBF2

Wspomnieliśmy już, że dla dostatecznie gęstych danych usunięcie jednego wektora nie powinno mocno zmienić prawdopodobieństwa klasyfikacji $P(c_i|\mathbf{x}, \mathcal{S})$ w punkcie $\mathbf{x}$. Dla k-NN usunięcie przypadku $\mathbf{x}$ z wnętrza klasy otoczonego wyłącznie przez przypadki o tej samej etykiecie, nie zmieni wcale lub prawie wcale prawdopodobieństwa $P(c_i|\mathbf{x})$. Dopuki nie spowoduje to drastucznych zmian w rozkładzie prawdopodobieństwa takie postępowanie może byc uzasadnione. Jeśli usunięcie wektora $\mathbf{x}$ spowoduje spore zmniejszenie prawdopodobieństwa przynależności punktu $\mathbf{x}$ do danej klasy znaczy to, że ten wektor jest istotny i należy go zachować. Inaczej mówiąc, możemy usunąć przypadek $\mathbf{x}$ jeśli

$$P(c(\mathbf{x}_i)|\mathbf{x}_i; \mathcal{S})\beta < P(c(\mathbf{x}_i)|\mathbf{x}_i; \mathcal{S}^i) \tag{2.6}$$

gdzie $\beta \in (0, 1)$ decyduje o wielkości dopuszczalnego zmniejszenia prawdopodobieństwa. Dla $\beta = 1$ nie zostanie usunięty żaden wektor, prawdopodobieństwo $P(c(\mathbf{x}_i)|\mathbf{x}_i; \mathcal{S})$ nigdy nie wzrośnie w wyniku usunięcia wektora $\mathbf{x}_i$. Im β mniejsze od 1 tym więcej wektorów ulegnie usunięciu, niestety kosztem większych deformacji rozkładu prawdopodobieństwa w punkcie $\mathbf{x}$ i jego okolicy. Usuwanie wektorów można przeprowadzać pojedynczo, za każ-



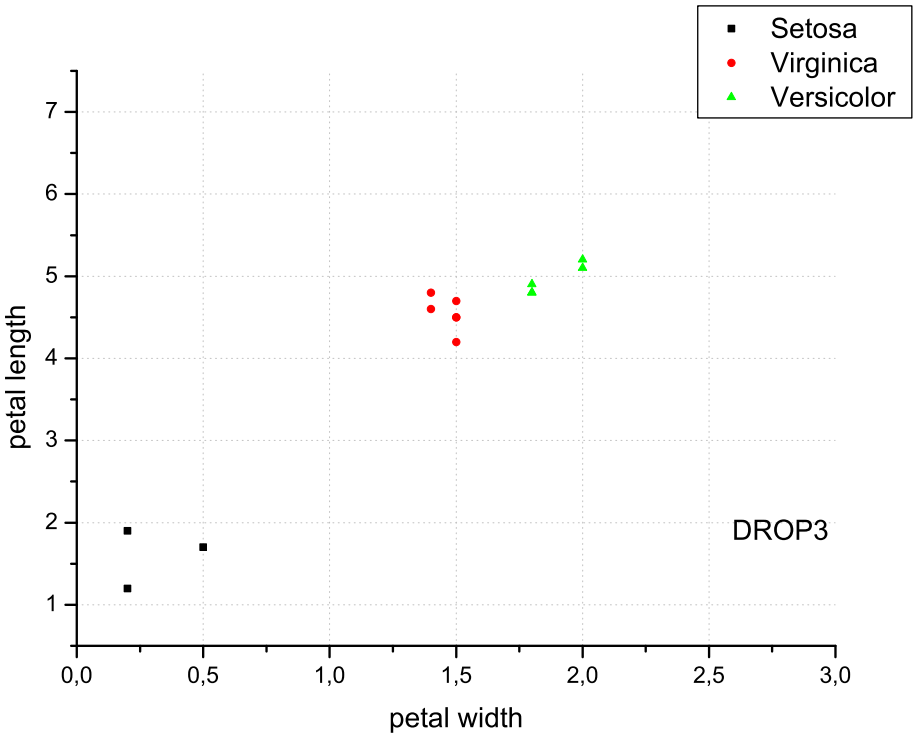
Rysunek 2.8. Rezultat działania ENRBF ($\beta = 0.99$) dla danych Iris

dym razem obliczając prawdopodobieństwo na zmniejszającym się zbiorze referencyjnym - w ten sposób unikniemy przypadku w którym moglibyśmy usunąć zbyt dużo wektorów podczas jednej epoki. Jednak takie podejście jest kosztowne obliczeniowo - wymaga w każdej epoce policzenia odpowiednich prawdopodobieństw dla wszystkich pozostających

w $\mathcal{S}$ przypadków. Przyspieszenie algorytmu możemy uzyskać usuwając całe grupy wektorów spełniających powyższe kryterium, jeśli tylko wektory z takiej grupy są daleko położone od siebie, tzn. usunięcie jednego z nich nie ma praktycznie wpływu na rozkład prawdopodobieństwa w pobliżu pozostałych przypadków zaznaczonych do usunięcia.

DROP 1-5

W przypadku klasyfikatora k-NN usunięcie dowolnego przypadku może co najwyżej spowodować "osłabienie" klasyfikacji wektorów dla których usunięty wektor był jednym z k najbliższych sąsiadów. Niech $\mathcal{A}(\mathbf{x})$ oznacza zbiór wektorów dla których $\mathbf{x}$ jest jednym z k najbliższych sąsiadów. Reguła selekcji zaproponowana przez Wilsona i Martineza [13] jest postaci: usuń wektor $\mathbf{x}$ z $\mathcal{S}$ (na początku bierzemy $\mathcal{S} = \mathcal{T}$) jeśli nie zmniejszy to liczby poprawnie klasyfikowanych wektorów ze zbioru $\mathcal{A}(\mathbf{x})$. Metoda działa analogicznie jak RNN z tym, że nie sprawdzamy za każdym razem dokładności klasyfikacji wszystkich wektorów ze zbioru treningowego, lecz tylko tych ze zbioru $\mathcal{A}(\mathbf{x})$. Algorytm DROP1 (*ang. Decremental Reduction Optimization Procedure*) jest algorytmem dekremencyjnym, czyli zaczynamy z $\mathcal{S} = \mathcal{T}$. Następnie usuwamy kolejne wektory z $\mathcal{S}$ zgodnie z powyższą regułą. Aby przyśpieszyć działanie algorytmu dla każdego przypadku, na początku tworzone są listy zawierające $k + 1$ sąsiadów oraz odpowiadające im listy $\mathcal{A}$. Jest to najbardziej czasochłonna operacja algorytmu $O(dn^2)$. Za każdym razem, gdy dany przypadek zostaje usunięty, następuje aktualizacja odpowiednich list ($O(dn)$). Kosztem pamięci zużytej na przechowywanie wszystkich tych list zmniejszamy koszt klasyfikacji kNN danego przypadku do $O(k)$. Algorytm usuwa wektory stanowiące szum (usunięcie takiego wektora powinno poprawić klasyfikację wektorów dla których był on jednym z najbliższych sąsiadów) jak i wektory położone wewnątrz klastrów. DROP1 jest czuły na kolejność prezentacji, co zostało uwzględnione w DROP2, gdzie najpierw wektory zostają uszeregowane według odległości od najbliższego "wroga". Taki sposób sortowania sprawia, że najpierw rozpatrywane do usunięcia są wektory z wnętrz klas a na końcu najbardziej konfliktowe przypadki z obszarów granicznych. DROP1 dodatkowo może w niektórych przypadkach powodować usunięcie całych klastrów (w najbardziej patologicznym przypadku nawet całych klas), dlatego w DROP2 reguła edycyjna jest bardziej ostrożna: usuwamy $\mathbf{x}$ z $\mathcal{S}$ jeśli nie zmniejszy to liczby poprawnie klasyfikowanych wektorów dla których $\mathbf{x}$ był jednym z k najbliższych sąsiadów w $\mathcal{T}$. Wstępne posortowane danych wydaje się dobrym rozwiązaniem problemu niestabilności rozwiązania, wynikającej z losowej kolejności prezentacji wzorców. Musimy jednak pamiętać o tym, że wektory o małej odległości od najbliższego przypadku z przeciwnej klasy, nie muszą być wektorami granicznymi, lecz mogą to być przypadki błędne, położone wewnątrz regionów zajętych w większości przez przypadki z przeciwnej klasy. Dlatego, przed posortowaniem, dobrze jest pozbyć się szumu. DROP3 (rys.2.9) jest rozszerzeniem DROP2, w którym zastosowano we wstępnej fazie algorytm ENN filtrujący szum. Algorytm DROP4 używa bardziej ostrożnego kryterium przy usuwaniu szumu - przypadek zostaje usunięty jeśli spełnia regułę edycyjną Wilsona (ENN) oraz regułę użytą w algorytmach DROP (czyli nie zwiększa liczby błędnej klasyfikacji wektorów dla których badany jest jednym z k najbliższych sąsiadów). Algorytm DROP5 jest kolejną modyfikacją DROP2, w której najpierw posortowane dane prezentujemy w kolejności odpowiadającej rosnącej odległości od najbliższego "wroga" - co odpowiada wstępnej fazie filtrującej szum. Następnie powtarzany jest DROP2 dla odwrotnej kolejności wzorców, aż do zbieżności.



Rysunek 2.9. Rezultat działania DROP3 dla danych Iris

2.3.3. Metody prototypowe

Metodami prototypowymi nazwane zostały modele, które poszukują najbardziej użytecznych, wzorcowych, przypadków, które z wysoką dokładnością potrafią reprezentować często spore grupy wektorów za pomocą jednego lub kilku "super-wektorów" referencyjnych. Zyskujemy przez to drastyczne uproszczenie problemu, nie mając jednak gwarancji na zachowanie poprawności klasyfikacji.

LVQ

Kwantyzacja wektorów (*ang. Vectors Quantization*) jest metodą zaproponowaną przez Kohonena[5], która za pomocą ustalonej liczby wektorów prototypowych (zwanych w tym przypadku wektorami kodującymi) stara się odwzorować rozkład danych ze zbioru treningowego. Każdy z wektorów kodujących reprezentuje inny obszar przestrzeni cech. Ich położenie ustalane jest w procesie uczenia (LVQ *ang. Learning VQ*) pod nadzorem. Jeśli oznaczymy zbiór wektorów kodujących (zwanych księgą szyfrów (*ang. code book*))

$$\mathcal{V} = \{\{c(\mathbf{v}_j), \mathbf{v}_j\}; j = 1, ..., L\} \tag{2.7}$$

wówczas dla przypadku $\mathbf{x}$ adaptacji ulega położenie najbliższego wektora kodującego zgodnie z regułą

$$\mathbf{v}_j(t+1) = \begin{cases} \mathbf{v}_j(t) + \alpha_t \left[\mathbf{x}_i - \mathbf{v}_j(t)\right] & \text{jeśli} \quad c(\mathbf{v}_j) = c(\mathbf{x}_i) \\ \mathbf{v}_j(t) - \alpha_t \left[\mathbf{x}_i - \mathbf{v}_j(t)\right] & \text{jeśli} \quad c(\mathbf{v}_j) \neq c(\mathbf{x}_i) \end{cases} \tag{2.8}$$

Położenie pozostałych wektorów nie ulega zmianie. Stała uczenia $\alpha_t \in (0,1)$ powinna maleć ze wzrostem t liczby prezentacji przypadków ze zbioru treningowego.
Trudnością związaną z LVQ jest ustalenie liczby wektorów prototypowych. Chcielibyśmy aby algorytm szukający wektorów referencyjnych potrafił dla każdego zbioru danych samodzielnie ustalić ilość potrzebnych przypadków. LVQ może jednak znaleźć zastosowanie jako metoda optymalizująca położenie wektorów referencyjnych otrzymanych za pomocą innych algorytmów.

MC1 i RMHC

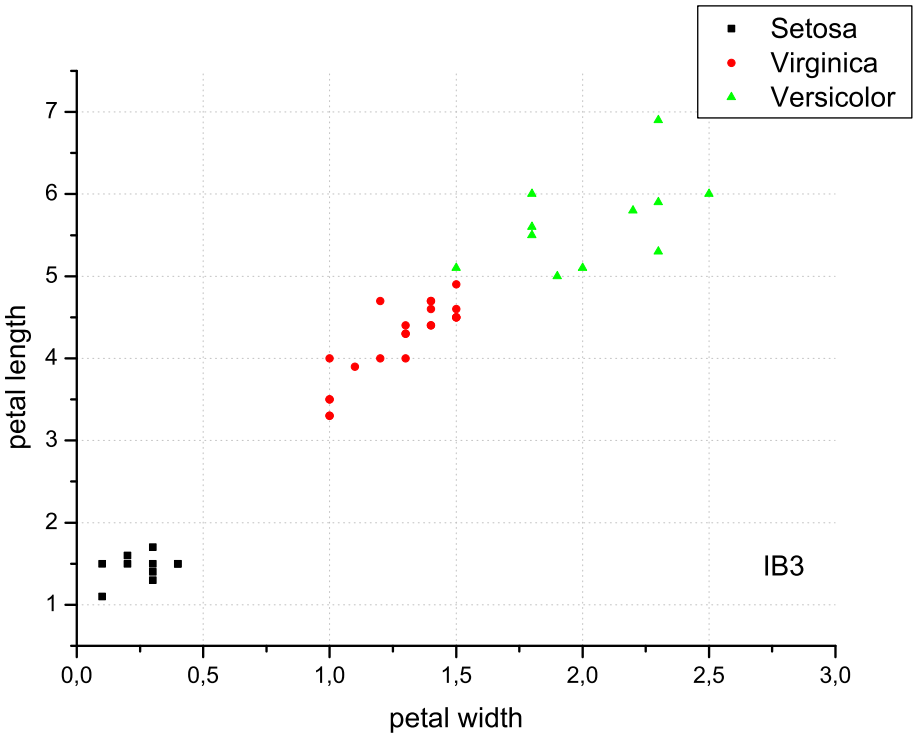
Bardzo prostą i szybką metodą jest metoda stochastycznego szukania połączonego z metodami wspinaczki zaproponowaną przez Skalaka [12]. Algorytm MC1 (*ang. Monte Carlo 1*) wybiera losowo l wektorów jako zbiór referencyjny dla klasyfikatora. Następnie przeprowadzany jest test klasyfikacji na zbiorze $\mathcal{T}$. Operację powtarzamy wielokrotnie, losując w każdej epoce nowy zestaw wektorów i zapamiętując tylko ten dla którego błąd klasyfikacji był najmniejszy. Algorytm RMHC (*ang. Random Mutation Hill Climbing*) działa podobnie, z ta różnicą, że proces szukania inspirowany jest algorytmami ewolucyjnymi. Zbiór l wektorów szukanych reprezentowany jest w postaci ciągu bitów, w którym w każdej epoce ulega mutacji jeden losowo wybrany bit, co w istocie sprowadza się do zamiany losowo wybranej pary w $\mathcal{S}$ i $\mathcal{T}$. Zostaje zachowana tylko ta sekwencja dla której funkcja przetrwania (czyli poprawność klasyfikacji) osiągnęła największą wartość. Nie mamy gwarancji znalezienia najlepszych prototypów, lecz dla wielu zbiorów danych powyższa metoda daje dobre wyniki. Problemem pozostaje, podobnie jak w przypadku LVQ, wybór ilości szukanych wektorów referencyjnych oraz ilości losowań (mutacji). Skalak stosował w doświadczeniach l = ilości klas, klasyfikatorem był k-NN z $k = 1$ dla $m = 100$ liczby epok.

IB3

Algorytm IB3, zaproponowany w 1992 roku przez Aha, zachowuje w zbiorze $\mathcal{S}$ tylko wektory statystycznie "akceptowalne". Rozpoczynamy od umieszczenia losowo wybranego wektora w $\mathcal{S}$. Następnie, każdy przypadek z $\mathcal{T}$ jest dodawany do $\mathcal{S}$ jeśli jest błędnie klasyfikowany przez "akceptowalne" wektory z $\mathcal{S}$ (gdy takie wektory nie występują w tym zbiorze, wówczas wybierane są losowo). W celu określenia, czy dany wektor jest akceptowalny, definiowana jest górna i dolna granica przedziału zaufania

$$C_{max/min} = \frac{p + \frac{z^2}{2n} \pm z\sqrt{\frac{p(p-1)}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}} \tag{2.9}$$

gdzie p oznacza prawdopodobieństwo odniesienia sukcesu w n próbach a z jest poziomem zaufania. Dla danego wektora z $\mathcal{S}$ we wzorze (2.9) p oznacza częstość poprawnej klasyfikacji, czyli stosunek liczby poprawnych klasyfikacji do n liczby wszystkich przeprowadzonych klasyfikacji wektora od momentu wprowadzenia go do $\mathcal{S}$. Granice przedziału (2.9) obliczamy też w stosunku do klasy z której pochodzi badany przypadek, wówczas, p oznacza stosunek liczby rozpatrywanych dotychczas przypadków z tej klasy do n wszystkich rozpatrywanych przypadków. Wektor w zbiorze $\mathcal{S}$ jest akceptowalny, jeśli dolna granica przedziału zaufania dla tego przypadku jest większa od górnej granicy przedziału zaufania dla klasy z której pochodzi dany przypadek, przy poziomie zaufania 90% ($z = 0.9$). Podobnie przypadek jest usuwany (jest nieakceptowalny), jeśli górna granica przedziału zaufania dla tego przypadku jest statystycznie niższa (dla poziomu zaufania 70%($z = 0.7$)) od dolnej granicy przedziału zaufania dla klasy z której pochodzi ten przypadek[13].



Rysunek 2.10. Rezultat działania IB3 dla danych Iris

ELH, ELGrow i Explore

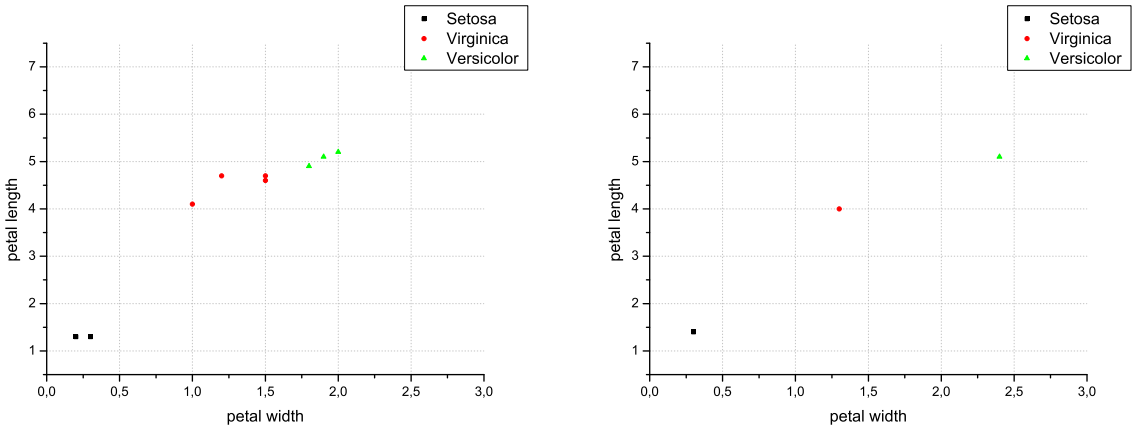
Cameron-Jones zaproponował stosowanie funkcji kosztu postaci

$$J(m,n,x) = F(m,n) + m\log_2(c) + F(x,n-m) + x\log_2(c-1) \tag{2.10}$$

gdzie n jest liczbą wektorów w $\mathcal{T}$, m liczbą wektorów w $\mathcal{S}$, c liczbą klas, x liczbą wektorów które zostały źle zaklasyfikowane przez k-NN a $F(m,n)$ jest kosztem kodowania n dostępnych przypadków za pomocą m wektorów zdefiniowanym następująco:

$$F(m,n) = \log^*(\sum_{j=0}^{m} c_j^n) = \log^*\left(\sum_{j=0}^{m} \frac{n!}{j!(n-j)!}\right) \tag{2.11}$$

gdzie $\log^*(x)$ oznacza sumę po dodatnich wyrażeniach $\log_2(x), \log_2(\log_2 x)$, itd.[13].
Algorytm ELH (*ang. Encoding Length Heuristic*) dodaje kolejne wektory z $\mathcal{T}$ do pustego na początku zbioru $\mathcal{S}$ jeśli tylko powoduje to zmniejszenie kosztu $J(\cdot)$ w stosunku do sytuacji w której byśmy tego nie zrobili. ELGrow (*ang. Encoding Length Grow*) po zakończonej fazie ELH dodatkowo przeprowadza redukcję elementów zbioru $\mathcal{S}$, odrzucając dany przypadek jeżeli zaowocuje to obniżeniem kosztu (2.10). Kolejny algorytm z tej rodziny - *Explore*, stanowi rozszerzenie algorytmu ELGrow poprzez połączenie ze stochastycznym szukaniem minimum kosztu. Mianowicie, po zakończeniu ELGrow, wykonywane jest 1000 mutacji polegających na losowym odrzucaniu lub dodawaniu pojedyńczych wektorów jeśli tylko taka operacja powoduje zmniejszenie kosztu. Powyższe metody dają w rezultacie bardzo małe zbiory referencyjne (rys.2.11), zwłaszcza ELGrow i Explore generują zbiory zawierające często tylko po jednym przypadku z każdej klasy.



Rysunek 2.11. Rezultat działania ELH (po lewej) i Explore (po prawej) na zbiór danych Iris

DEL

DEL (*ang. Decremental Encoding Length*) to kolejny algorytm zaproponowany przez Wilsona i Martineza [13] w którym zastosowano kryterium ELH. Algorytm przypomina DROP3 z tym, że przypadek jest usuwany zgodnie z zasadą stosowaną w metodach Cameron-Jonesa, tzn. jeśli tylko nie zwiększy to kosztu (2.10). Jest to więc dekremencyjna wersja algorytmu ELH.

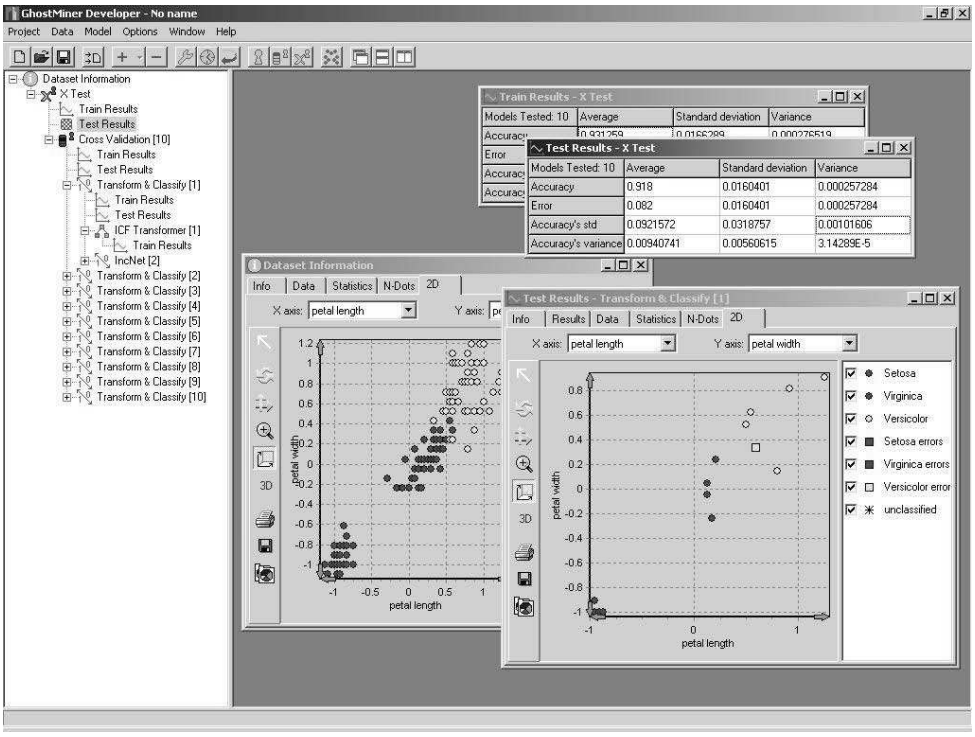
2.4. Podsumowanie

Opisane tu algorytmy dają pewien ogląd stosowanych podejść przy szukaniu wektorów referencyjnych. Różnorodnych metod, poszukujących prototypy i redukujących objętości zbiorów treningowych, istnieje bardzo dużo. Funkcjonują też różnorodne ich modyfikacje i nie sposób wymienić tu wszystkie. W pracy skupiono się na metodach, które zostały zaimplementowane i wykorzystane w eksperymentach numerycznych, opisanych w następnym rozdziale.

3. Eksperymenty numeryczne

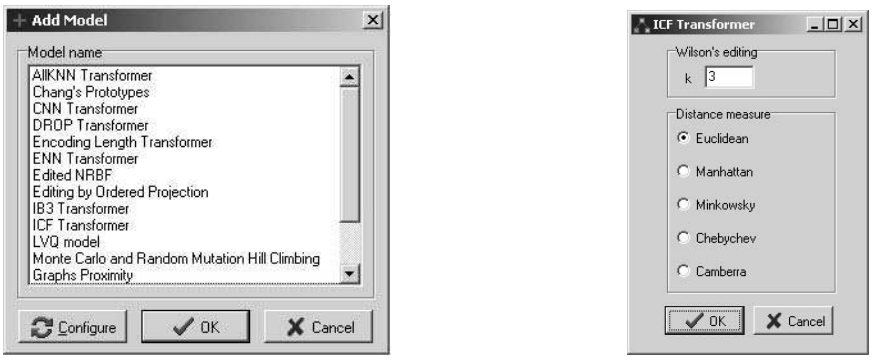
3.1. Implementacja i obliczenia

Opisane w poprzednim rozdziale metody wyboru wektorów uczących, zostały zaimplementowane w języku C++ i dołączone w postaci dodatkowych modułów (bibliotek DLL) do programu Ghost Miner. Część kodów źródłowych poszczególnych modeli (min. IB3, ELH, Explore, ElGrow, DEL, DROP1-5) oparta została na źródłach pochodzących z programu "Drop", napisanego w języku C przez Wilsona i Martineza, udostępnionego pod adresem: `http://axon.cs.byu.edu/~randy/pubs/ml/drop/`. Modele GE i RNGE zaimplementowane zostały w oparciu o źródła w języku Java, dostępne pod adresem: `http://cgm.cs.mcgill.ca/~godfried/teaching/projects.pr.98/sergei/`
Zastosowany w obliczeniach program Ghost Mainer (rys.3.1) służący do eksploracji danych, udostępnia wiele narzędzi do preparacji i wizualizacji danych a także szereg modeli klasyfikujących. Każda metoda wyboru wektorów referencyjnych dołączona została do tego programu



Rysunek 3.1. Program do eksploracji danych Ghost Miner

w postaci funkcji transformującej dane (rys.3.2), której argumentem w każdym przypadku jest zbiór treningowy zaś wynikiem działania (wartością zwracaną) - zbiór wektorów referencyjnych. Tak otrzymany zbiór, uczestniczył następnie w procesie uczenia jednego z modeli klasyfikacyjnych dostępnych w programie "Ghost Miner". Dokładność nauczonego modelu sprawdzana była podczas klasyfikacji przypadków pochodzących z odpowiednio przygotowanego zbioru testowego.



Rysunek 3.2. Konsola wyboru metod transformujących zbiór treningowy (po lewej) i przykładowa konsola służąca do ustalenia parametrów modelu ICF

Metody wyboru wektorów zastosowane zostały dla siedmiu zbiorów danych (zob. Dodatek). Wszystkie dane przed rozpoczęciem obliczeń zostały znormalizowane zgodnie ze wzorem

$$x_i' = \frac{x_i - \overline{x}}{\frac{1}{2}(x_{max} - x_{min})} \tag{3.1}$$

gdzie x_{max} i x_{min} oznaczają największą i najmniejszą wartość i-tej cechy. Podczas normalizacji 5% ($\alpha = 0.05$) wektorów o najbardziej skrajnych wartościach danej cechy zostało pominiętych w celu uniknięcia szkodliwego wpływu danych "odstających" od struktury zbioru uczącego na normalizację. Jeśli dane nie posiadały odpowiedniego zbioru testowego, lecz były w postaci jednego zbioru danych, wówczas stosowana była dziesięciokrotna kroswalidacja (10CV), tzn. dane dzielone były losowo na 10 równych liczebnie podzbiorów i w każdej z 10 prób jeden z nich traktowany był jako zbiór testowy, pozostałe 90% przypadków uczestniczyło w procesie uczeniu klasyfikatora. Wszystkie obliczenia powtórzone zostały dziesięciokrotnie.

3.2. Wyniki klasyfikacji

W poniższych tabelach umieszczone są wyniki otrzymane dla poszczególnych zbiorów danych. Pierwsza kolumna zawiera, wyrażony w procentach, stosunek $|\mathcal{S}|/|\mathcal{T}|$ ilości wektorów otrzymanych po zastosowaniu danego algorytmu do ilości wektorów w całym dostępnym zbiorze treningowym, stanowiący wartość średnią dla 100 testów. Pozostałe kolumny zawierają wyniki klasyfikacji przeprowadzone przy pomocy sześciu metod klasyfikacji: k-NN z metryką euklidesową z automatycznym wyborem k (dla $k = 1, \ldots, 10$ wybieramy takie które daje najmniejszy błąd dla testu 10CV), NRBF z metryką euklidesową i "rozmyciem" funkcji Gaussa $\sigma = 4$, FSM z gaussowskimi funkcjami transferu, IncNet również z gaussowskimi funkcjami transferu, SSV ze strategią szukania pierwszy-najlepszy, SVM z gaussowskimi funkcjami jądra.
Pierwszy wiersz w tabeli zawiera wyniki klasyfikacji dla przypadku gdzie zbiorem referencyjnym był cały zbiór treningowy $\mathcal{S} = \mathcal{T}$. W przypadkach oznaczonych $*$ zastosowanie danej metody klasyfikacji stało się niemożliwe z powodu zbyt dużej selekcji wektorów. Na przykład, metody ELGrow i Explore zazwyczaj pozostawiają w zbiorze treningowym po kilka wektorów w każdej z klas. Tak małe zbiory treningowe sprawiają kłopoty w trakcie klasteryzacji stosowanej w FSM a także przy próbie kroswalidacji takiego zbioru np. przy obcinaniu drzewa decyzyjnego SSV, lub przy automatycznym wyborze k dla k-NN. Dla tak małych zbiorów referencyjnych ilość sąsiadów w kNN wybierana była ręcznie $k = 1$. We wszystkich metodach selekcji wykorzystana została metryka euklidesowa. Ilość sąsiadów w metodach korzystających z k-NN wynosiła $k = 3$ (analogicznie do badań przeprowadzonych przez Wilsona i Martineza w [13]). Dla ENRBF ustalono

$\sigma = 1$ i $\alpha = 0.2$. ENRBF99 oznacza zastosowanie kryterium ENRBF2 (2.6) pozwalającego na usunięcie wektora jeśli nie spowoduje to zmniejszenie prawdopodobieństwa jego klasyfikacji do 99% ($\beta = 0.99$) jego wartości z przed selekcji. ENRBF95 odpowiada ENRBF2 odpowiednio z $\beta = 0.95$. W metodach MC1 i RMHC rozmiar zbioru referencyjnego równy był liczbie klas w $\mathcal{T}$ a ilość iteracji $m = 100$ (analogicznie jak w [12]). Dla porównania z tymi dwiema metodami wykonany został analogiczny test LVQ inicjowany pojedynczymi przypadkami z każdej klasy. Współczynnik uczenia dla LVQ wynosił $\alpha = 0.02$ a ilość iteracji $m = 20$.

Metody filtrujące szum, w przypadku zbioru danych Wine, Iris i Wisconson Breast Cancer dały w sumie niewielką redukcję zbioru oryginalnego, w niektórych przypadkach polepszając dokładność klasyfikacji. Najostrożniejszym kryterium selekcji szumu okazał się ENRBF, z kolei najbardziej radykalny w tej grupie AllkNN, dla danych Pima Indian Diabetes, uznał za błędne blisko połowę przypadków a ich usunięcie poprawiło klasyfikacje k-NN o kilka procent, nie szkodząc innym klasyfikatorom. W tym samym przypadku ENRBF usunął tylko 2.22% wektorów osiągając porównywalną dokładność klasyfikacji jak dla zbioru otrzymanego z AllkNN. W sumie filtry szumu spełniły swoją rolę, poprawiając zazwyczaj dokładność klasyfikacji o parę procent, tylko w niektórych przypadkach można zauważyć niewielki spadek dokładności.

Wśród pozostałych metod wyróżnić należy metody ICF, IB3, DROP3 i NRBF99, które w dużym stopniu redukują rozmiar zbioru treningowego, bez większych szkód dla klasyfikacji. Skuteczność tych metod jest jednak uzależniona od użytego klasyfikatora oraz różni się dla poszczególnych zbiorów danych.

Dla zbioru Wine metody GE i ENRBF99, dając około 60 procentową redukcję, nie zaszkodziły poważnie żadnemu z zastosowanych klasyfikatorów. Metody bardziej selekcyjne, jak ICF, IB3, CNN i DROP2-5, dobrze wypadły dla klasyfikatorów kNN i NRBF. Zadziwiające, że zbiór referencyjny otrzymany metodą Explore, zachowując tylko 1.95% wektorów (po jednym przypadku na klasę), zapewnia bardzo dobrą klasyfikację dla k-NN, NRBF. Również inne metody szukające pojedynczych wektorów reprezentujących całą klasę (MC1, RMHC, LVQ1) potrafiły w tym przypadku zachować wysoki poziom poprawności klasyfikacji. Zmniejszenie ilości wektorów treningowych negatywnie wpłynęło na SVM - im mniejszy zbiór referencyjny tym gorszą otrzymujemy klasyfikację.

Dla zbioru Pima Indian Diabetes wpływ redukcji ilości przypadków treningowych miał mniej szkodliwy wpływ na klasyfikator SVM. Metody Explore i LVQ redukując objętość zbioru do 0.3%, spowodowały zmniejszenie dokładności klasyfikacji SVM o około 8% a w przypadku k-NN i NRBF dając w rezultacie nawet kilkuprocentowy wzrost poprawności klasyfikacji. ENRBF99, pozostawiając 75.71% wektorów uczących, "nie zaszkodził" żadnemu klasyfikatorowi. Inne metody, dające większą redukcję, powodowały czasem kilkuprocentowy spadek dokładności klasyfikacji. W najgorszym przypadku, dla metody DROP1, błąd klasyfikacji SVM uległ zwiększeniu o ponad 20%, FSM o około 12% a SSV o 6%.

W przypadku danych Czerniak można zauważyć ekstremalnie różną skuteczność działania poszczególnych klasyfikatorów. Drzewo decyzyjne SSV zapewnia 100% poprawność klasyfikacji, gdzie k-NN i NRBF w około połowie przypadków podejmują mylną decyzję odnośnie klasy danego wzorca. Wybór wektorów referencyjnych dowolną metodą (oprócz DROP1) nie spowodował zmniejszenia dokładności SSV. 17.20% przypadków otrzymanych metodą DEL wystarcza aby zapewnić 100% dokładność dla tego klasyfikatora. Metoda GE nie usunęła żadnego przypadku ze zbioru treningowego, oznacza to, iż wszystkie przypadki występujące w tym zbiorze danych są granicznymi wektorami w diagramie Voronoi. Klasyfikacja SVM na zredukowanym zbiorze referencyjnym prowadziła za każdym razem do sporego zmniejszenia dokładności klasyfikacji. Metody wybierające po kilka prototypów, takie jak LVQ, Explore, MC1 i RMHC, w przypadku klasyfikacji k-NN i NRBF, spowodowały zmniejszenie o kilka procent ilości popełnianych błędów. Interesujący jest fakt, że LVQ1 spowodowało dosyć duży wzrost poprawności klasyfikacji kNN, NRBF oraz FSM.

Dla zbioru Cleveland Heart Disease większość metod wyboru wektorów referencyjnych nie spowo-

dowała znacznego wzrostu błędu klasyfikacji. Wyjątek stanowi DROP1, który negatywnie wpłynął na wyniki wszystkich klasyfikatorów. Bardzo małe zbiory referencyjne również w tym przypadku wystarczają aby zapewnić dostatecznie dużą dokładność klasyfikacji a niekiedy (Explore i LVQ1 dla kNN i NRBF) nawet powodując jej kilkuprocentowy wzrost.
Dane Appendictis charakteryzują się tym, że klasa większościowa zawiera aż 80% przypadków z całego zbioru treningowego, dlatego dobrze działający klasyfikator dla takich danych nie może popełniać błędu większego niż 20%. Tak nierównomierny rozkład danych stał się powodem niestabilnego zachowania niektórych metod opisanych w niniejszej pracy. Zbiór referencyjny otrzymany metodami Explore, ElGrow i DEL zawierał zazwyczaj tylko kilka przypadków z klasy większościowej. Podobnie dla DROP1 i IB3 zdarzało się, że wszystkie przypadki z klasy mniejszościowej zostały usunięte ze zbioru uczącego. ENRBF99 zredukował ilość wektorów do 30.89% nie powodując praktycznie żadnych szkód przy klasyfikacji dowolną z zaprezentowanych metod. LVQ, MC1 i RMHC ponownie wykazały, że jeden wektor może często być dobrym reprezentantem całej klasy.
Również dla danych Wisconsin Breast Cancer metody prototypowe wykazały wysoką skuteczność. Pozostałe modele dobrze spisały się dla kNN i NRBF powodując w najgorszym przypadku kilkuprocentowy wzrost błędu klasyfikacji. Bardzo dobrze wypadły metody GE i RNGE zachowując wysoką dokładność wszystkich zastosowanych metod klasyfikacji.
Dla danych Iris klasyfikator SVM okazał się czuły ze względu na ilość przypadków uczących i można zauważyć wręcz proporcjonalny spadek dokładności klasyfikacji wraz ze wzrostem redukcji zbioru uczącego. Jedynie ENRBF99, wybierając 71.69% wektorów do uczenia SVM, potrafił nie zaszkodzić temu klasyfikatorowi. Z pośród innych metod dobrze wypadł GE oraz IB3. Większość modeli potrafiło wygenerować poprawne zbiory referencyjne dla klasyfikatorów k-NN, NRBF i IncNet. W tym przypadku metody LVQ, MC1 i RMHC spowodowały niewielki wzrost błędu klasyfikacji, choć inna metoda - Explore, wybierając równie mały zbiór referencyjny (2.24%), nie wpłynęła negatywnie na poprawność klasyfikacji.

Zbiór danych : Wine							
Metoda	$\lvert\mathcal{S}\rvert/\lvert\mathcal{T}\rvert$	k-NN	NRBF	FSM	IncNet	SSV	SVM
$\mathcal{S}=\mathcal{T}$	100.00 ± 0.00	96.28 ± 4.62	95.38 ± 4.73	95.67 ± 5.13	97.98 ± 3.14	91.28 ± 7.44	98.32 ± 2.88
ENN	95.51 ± 0.87	94.88 ± 4.80	96.24 ± 4.81	95.78 ± 4.81	96.72 ± 4.01	90.56 ± 6.21	97.00 ± 3.76
RENN	95.10 ± 0.81	95.16 ± 4.59	96.29 ± 3.95	95.61 ± 5.05	97.14 ± 3.95	90.10 ± 7.41	96.84 ± 4.13
AllKNN	92.31 ± 1.11	94.94 ± 5.12	95.67 ± 4.50	94.84 ± 5.25	97.59 ± 3.67	90.51 ± 7.24	96.66 ± 4.72
ENRBF	98.94 ± 0.31	96.02 ± 4.42	95.89 ± 4.15	95.54 ± 4.82	97.81 ± 3.05	90.49 ± 7.27	97.91 ± 3.07
CNN	16.50 ± 2.28	90.93 ± 6.78	91.93 ± 5.61	84.00 ± 11.09	93.48 ± 6.95	81.30 ± 12.69	67.00 ± 16.09
CA	4.33 ± 0.77	85.97 ± 8.90	84.07 ± 11.17	50.48 ± 12.31	81.91 ± 1.03	*	51.54 ± 13.93
DROP1	5.49 ± 0.83	90.29 ± 7.26	90.89 ± 6.96	65.79 ± 15.77	88.15 ± 8.93	*	52.83 ± 14.33
DROP2	11.94 ± 1.62	93.53 ± 5.41	93.92 ± 5.65	71.24 ± 13.43	89.72 ± 8.44	75.98 ± 13.17	50.39 ± 11.93
DROP3	11.96 ± 1.53	93.79 ± 5.76	94.83 ± 5.61	70.35 ± 14.47	90.57 ± 8.19	74.07 ± 12.15	45.25 ± 7.55
DROP4	12.19 ± 1.47	93.32 ± 5.45	95.05 ± 5.53	68.97 ± 13.89	90.82 ± 8.75	74.65 ± 10.33	47.16 ± 10.29
DROP5	9.99 ± 2.12	93.96 ± 5.56	94.32 ± 4.75	73.72 ± 14.40	91.24 ± 6.32	71.26 ± 11.90	86.12 ± 10.09
ICF	8.55 ± 1.59	90.72 ± 6.76	90.67 ± 6.87	67.60 ± 12.26	92.01 ± 5.51	65.12 ± 15.25	48.28 ± 15.19
GE	66.60 ± 2.00	94.69 ± 5.25	93.74 ± 5.29	94.89 ± 4.91	96.91 ± 4.63	92.06 ± 6.19	95.89 ± 4.18
RNGE	23.29 ± 1.39	92.90 ± 5.59	93.97 ± 5.58	83.27 ± 9.63	92.40 ± 5.84	86.94 ± 7.99	74.92 ± 13.87
IB3	18.07 ± 5.45	94.04 ± 6.35	94.26 ± 5.58	88.42 ± 9.61	92.80 ± 5.96	78.42 ± 11.13	90.60 ± 9.10
ENRBF99	60.65 ± 1.92	95.27 ± 5.02	97.92 ± 3.12	95.73 ± 4.84	97.63 ± 3.28	91.34 ± 6.77	96.47 ± 4.15
ENRBF95	33.96 ± 2.27	94.70 ± 4.46	97.24 ± 4.07	91.10 ± 7.98	96.68 ± 4.02	89.75 ± 6.89	94.41 ± 5.32
ELH	9.64 ± 1.95	92.24 ± 5.68	93.00 ± 6.58	61.72 ± 14.89	87.19 ± 9.74	71.50 ± 13.04	49.37 ± 10.82
ELGrow	2.05 ± 0.41	88.92 ± 7.31	88.52 ± 7.76	*	86.62 ± 8.71	*	37.83 ± 11.03
Explore	1.95 ± 0.39	95.38 ± 4.41	94.54 ± 5.46	*	92.81 ± 8.29	*	34.72 ± 5.63
DEL	4.90 ± 0.77	90.27 ± 7.09	90.54 ± 6.45	*	88.71 ± 8.03	*	47.54 ± 13.24
LVQ	1.87 ± 0.0	90.56 ± 7.11	90.69 ± 6.72	*	91.39 ± 6.43	*	33.15 ± 1.81
MC1	1.87 ± 0.0	89.63 ± 7.95	88.88 ± 7.42	*	89.21 ± 8.45	*	33.27 ± 2.15
RMHC	1.87 ± 0.0	89.53 ± 7.70	89.50 ± 8.16	*	90.31 ± 8.20	*	33.83 ± 3.15

Tabela 3.1. Wyniki klasyfikacji danych Wine

Zbiór danych : Pima Indian diabetes							
Metoda	$\lvert\mathcal{S}\rvert/\lvert\mathcal{T}\rvert$	k-NN	NRBF	FSM	IncNet	SSV	SVM
$\mathcal{S}=\mathcal{T}$	100.00 ± 0.0	69.95 ± 4.54	73.01 ± 4.81	73.12 ± 5.00	75.40 ± 4.76	73.76 ± 4.85	79.62 ± 0.60
ENN	73.15 ± 0.99	74.88 ± 4.24	66.00 ± 1.52	72.53 ± 4.90	75.75 ± 4.54	75.16 ± 4.35	77.25 ± 3.95
RENN	69.18 ± 1.14	74.50 ± 3.94	65.91 ± 1.19	70.62 ± 4.62	74.66 ± 4.36	75.07 ± 4.26	77.00 ± 3.66
AllKNN	54.28 ± 1.16	74.41 ± 4.60	66.38 ± 1.76	71.71 ± 5.13	75.58 ± 4.65	73.94 ± 4.47	77.05 ± 3.73
ENRBF	97.88 ± 0.31	71.36 ± 4.85	66.43 ± 1.36	73.63 ± 3.74	75.93 ± 4.33	73.69 ± 3.98	77.34 ± 4.58
CNN	50.53 ± 1.32	64.24 ± 5.75	69.66 ± 2.96	69.30 ± 5.92	73.24 ± 4.94	72.46 ± 4.50	76.19 ± 4.27
DROP1	6.76 ± 1.53	64.27 ± 6.62	68.96 ± 8.18	61.22 ± 8.19	72.11 ± 5.64	66.01 ± 8.31	57.90 ± 13.96
DROP2	17.84 ± 1.69	69.75 ± 5.18	70.00 ± 3.09	70.04 ± 4.65	75.33 ± 3.96	72.76 ± 4.69	74.85 ± 4.17
DROP3	12.99 ± 1.38	72.58 ± 4.68	73.74 ± 4.79	70.55 ± 4.89	76.03 ± 4.16	72.46 ± 5.68	74.94 ± 4.51
DROP4	17.77 ± 1.92	72.30 ± 4.94	71.22 ± 3.44	71.57 ± 4.54	75.72 ± 4.16	73.38 ± 5.43	76.16 ± 4.67
DROP5	18.99 ± 1.80	70.52 ± 4.61	71.93 ± 3.71	72.22 ± 4.90	76.17 ± 4.89	71.76 ± 5.37	76.56 ± 5.03
ICF	9.57 ± 0.74	71.16 ± 4.65	71.45 ± 4.33	66.36 ± 5.18	74.20 ± 4.65	70.76 ± 6.22	67.52 ± 3.76
GE	67.85 ± 1.27	64.17 ± 5.03	66.25 ± 1.73	69.45 ± 3.96	74.01 ± 4.24	71.61 ± 4.94	76.37 ± 3.69
RNGE	59.50 ± 1.05	67.23 ± 4.99	68.95 ± 2.67	71.67 ± 4.88	73.49 ± 4.56	72.74 ± 5.10	76.39 ± 4.42
IB3	7.82 ± 0.97	69.12 ± 4.99	72.04 ± 5.43	68.42 ± 5.64	74.04 ± 4.88	70.33 ± 6.10	66.56 ± 12.16
ENRBF99	75.71 ± 1.11	69.74 ± 5.42	69.76 ± 3.16	73.87 ± 4.82	76.26 ± 4.47	75.26 ± 4.11	77.06 ± 4.46
ENRBF95	53.55 ± 0.91	63.85 ± 6.08	59.56 ± 8.29	66.75 ± 5.72	70.99 ± 6.28	70.50 ± 5.88	76.33 ± 4.76
ELH	22.21 ± 2.05	67.33 ± 4.75	71.63 ± 4.86	67.98 ± 5.95	73.27 ± 5.65	70.59 ± 5.33	74.29 ± 3.90
ELGrow	0.32 ± 0.15	69.26 ± 6.22	69.32 ± 5.10	*	66.14 ± 7.61	*	68.20 ± 6.80
Explore	0.32 ± 0.10	72.45 ± 5.62	76.17 ± 4.60	*	66.85 ± 7.85	*	72.21 ± 7.46
DEL	1.13 ± 0.48	70.02 ± 5.96	71.10 ± 6.02	51.11 ± 19.52	71.35 ± 5.52	*	67.00 ± 8.47
LVQ	0.29 ± 0.0	72.52 ± 4.40	72.67 ± 3.92	*	72.21 ± 4.11	*	72.43 ± 4.23
MC1	0.29 ± 0.0	69.78 ± 4.98	70.22 ± 4.83	*	67.70 ± 7.29	*	69.76 ± 5.38
RMHC	0.29 ± 0.0	70.09 ± 5.70	70.49 ± 5.48	*	67.81 ± 7.12	*	69.69 ± 5.80

Tabela 3.2. Wyniki klasyfikacji danych Pima Indian diabetes

Zbiór danych : Czerniak							
Metoda	$\lvert\mathcal{S}\rvert/\lvert\mathcal{T}\rvert$	k-NN	NRBF	FSM	IncNet	SSV	SVM
$\mathcal{S}=\mathcal{T}$	100.0 ± 0.0	46.15 ± 0.0	50.00 ± 0.0	91.92 ± 7.12	65.76 ± 4.60	100.00 ± 0.0	76.92 ± 0.0
ENN	79.20 ± 0.0	57.69 ± 0.0	57.69 ± 0.0	79.61 ± 8.89	68.84 ± 4.94	100.00 ± 0.0	73.07 ± 0.0
RENN	76.40 ± 0.0	57.69 ± 0.0	57.69 ± 0.0	79.61 ± 7.03	63.46 ± 5.51	100.00 ± 0.0	69.23 ± 0.0
AllKNN	70.40 ± 0.0	57.69 ± 0.0	57.69 ± 0.0	95.76 ± 5.57	64.53 ± 4.79	100.00 ± 0.0	73.07 ± 0.0
ENRBF	93.20 ± 0.0	46.15 ± 0.0	50.00 ± 0.0	91.53 ± 6.22	63.07 ± 3.24	100.00 ± 0.0	65.38 ± 0.0
CNN	42.96 ± 0.87	41.53 ± 7.86	38.07 ± 7.35	96.92 ± 5.06	48.07 ± 13.10	100.00 ± 0.0	48.84 ± 3.16
CA	16.71 ± 0.90	45.00 ± 10.58	31.92 ± 9.07	80.38 ± 10.79	30.0 ± 3.97	100.0 ± 0.0	29.23 ± 5.50
DROP1	18.24 ± 2.55	41.53 ± 9.38	50.38 ± 4.94	83.84 ± 9.73	37.30 ± 12.30	97.30 ± 8.51	35.76 ± 6.79
DROP2	28.40 ± 0.0	42.30 ± 0.0	42.30 ± 0.0	79.23 ± 15.61	30.76 ± 2.56	100.00 ± 0.0	26.92 ± 0.0
DROP3	21.54 ± 2.05	50.00 ± 11.88	54.23 ± 7.78	87.69 ± 5.37	43.46 ± 18.31	100.00 ± 0.0	33.84 ± 4.36
DROP4	27.28 ± 0.30	39.23 ± 10.99	37.30 ± 9.07	88.07 ± 6.89	40.00 ± 10.60	100.00 ± 0.0	39.61 ± 2.59
DROP5	26.80 ± 0.0	53.84 ± 0.0	53.84 ± 0.0	88.46 ± 4.79	43.84 ± 7.73	100.00 ± 0.0	50.00 ± 0.0
ICF	20.0 ± 0.0	61.53 ± 0.0	69.23 ± 0.0	70.00 ± 5.37	58.84 ± 8.51	100.00 ± 0.0	38.46 ± 0.0
GE	100.0 ± 0.0	46.15 ± 0.0	50.00 ± 0.0	91.53 ± 7.20	65.38 ± 6.01	100.00 ± 0.0	76.92 ± 0.0
RNGE	54.8 ± 0.0	46.15 ± 0.0	46.15 ± 0.0	89.23 ± 8.65	58.09 ± 12.87	100.00 ± 0.0	50.00 ± 0.0
IB3	28.95 ± 4.05	51.92 ± 14.30	48.84 ± 7.26	94.61 ± 7.29	45.76 ± 13.25	100.00 ± 0.0	46.53 ± 10.16
ENRBF99	56.33 ± 0.78	39.61 ± 9.07	36.53 ± 9.11	96.92 ± 3.97	46.53 ± 13.50	100.00 ± 0.0	46.92 ± 2.43
ENRBF95	41.95 ± 0.68	31.58 ± 8.06	34.53 ± 5.06	95.00 ± 6.29	41.53 ± 6.48	100.00 ± 0.0	13.16 ± 1.85
ELH	29.99 ± 1.48	37.69 ± 9.02	43.07 ± 8.46	83.84 ± 12.53	41.15 ± 7.70	100.00 ± 0.0	42.69 ± 3.82
ELGrow	2.37 ± 0.74	60.38 ± 12.82	60.76 ± 13.17	*	55.00 ± 9.60	*	26.53 ± 7.35
Explore	2.82 ± 0.95	53.46 ± 12.08	53.84 ± 11.46	*	61.53 ± 5.43	*	24.23 ± 2.59
DEL	17.20 ± 0.0	61.53 ± 0.0	53.84 ± 0.0	58.07 ± 9.83	27.69 ± 6.73	100.00 ± 0.0	38.46 ± 0.0
LVQ1	1.60 ± 0.0	67.69 ± 2.68	68.46 ± 3.53	*	70.00 ± 2.43	*	24.23 ± 1.85
MC1	1.60 ± 0.0	53.46 ± 1.42	53.07 ± 8.65	*	60.38 ± 7.02	*	23.84 ± 1.62
RMHC	1.60 ± 0.0	63.84 ± 9.62	61.53 ± 12.69	*	54.64 ± 9.55	*	25.38 ± 7.29

Tabela 3.3. Wyniki klasyfikacji danych Czerniak

Zbiór danych : Cleveland heart disease							
Metoda	$\lvert\mathcal{S}\rvert/\lvert\mathcal{T}\rvert$	k-NN	NRBF	FSM	IncNet	SSV	SVM
$\mathcal{S}=\mathcal{T}$	100.0 ± 0.0	74.39 ± 7.55	79.39 ± 7.11	80.34 ± 6.14	82.11 ± 6.45	75.87 ± 8.41	81.55 ± 5.79
ENN	80.31 ± 1.24	79.61 ± 7.59	79.2 ± 7.59	79.43 ± 6.88	80.24 ± 7.70	77.44 ± 7.55	81.08 ± 6.96
RENN	76.94 ± 1.42	79.27 ± 6.85	78.94 ± 7.3	81.05 ± 6.75	80.19 ± 6.94	76.62 ± 7.53	79.85 ± 6.96
AllKNN	62.72 ± 1.86	78.48 ± 7.25	78.78 ± 7.7	79.62 ± 8.82	80.65 ± 7.52	77.1 ± 6.65	81.28 ± 7.05
ENRBF	89.84 ± 0.76	78.03 ± 7.24	80.06 ± 7.7	80.46 ± 7.05	82.58 ± 7.49	79.6 ± 5.71	82.24 ± 5.95
CNN	42.73 ± 2.07	70.9 ± 7.11	77.26 ± 7.85	73.67 ± 7.84	78.61 ± 8.24	71.39 ± 8.58	81.16 ± 6.54
DROP1	4.95 ± 2.21	67.45 ± 10.93	69.6 ± 10.6	62.53 ± 14.23	63.16 ± 11.51	62.87 ± 15.94	67.22 ± 12.43
DROP2	15.91 ± 2.76	76.95 ± 7.93	77.58 ± 8.48	76.53 ± 6.61	78.53 ± 6.77	71.67 ± 8.41	81.15 ± 6.54
DROP3	12.46 ± 2.31	77.76 ± 7.51	78.71 ± 7.46	77.23 ± 7.94	79.63 ± 8.35	72.12 ± 7.77	81.17 ± 6.77
DROP4	15.43 ± 2.47	77.87 ± 7.00	78.14 ± 7.22	77.55 ± 8.2	80.69 ± 7.78	70.33 ± 8.7	80.52 ± 6.2
DROP5	16.52 ± 2.49	77.31 ± 6.97	78.12 ± 7.09	79.01 ± 6.05	81.77 ± 7.71	73.46 ± 7.1	82.05 ± 7.24
ICF	11.80 ± 1.47	75.84 ± 7.83	77.57 ± 6.61	75.07 ± 8.12	76.04 ± 10.0	69.76 ± 8.73	79.63 ± 6.82
GE	99.07 ± 0.58	76.33 ± 7.45	80.88 ± 6.33	80.17 ± 6.23	81.74 ± 6.39	76.53 ± 7.78	81.63 ± 6.51
RNGE	55.17 ± 1.73	74.08 ± 8.6	79.9 ± 6.49	77.57 ± 7.77	80.11 ± 8.2	71.95 ± 8.62	81.93 ± 6.03
IB3	11.16 ± 1.90	76.86 ± 7.15	77.73 ± 7.13	77.36 ± 8.65	76.22 ± 9.82	72.47 ± 7.37	79.65 ± 6.74
ENRBF99	53.94 ± 3.32	78.98 ± 6.59	80.71 ± 7.2	79.41 ± 6.36	81.49 ± 6.81	76.41 ± 8.24	81.93 ± 5.87
ENRBF95	38.27 ± 2.81	77.90 ± 7.83	79.95 ± 6.65	77.16 ± 7.55	82.06 ± 7.13	74.06 ± 8.14	82.46 ± 7.17
ELH	17.02 ± 2.72	72.31 ± 8.08	76.29 ± 7.23	71.97 ± 8.56	74.92 ± 9.5	71.82 ± 8.51	79.84 ± 7.03
ELGrow	0.73 ± 0.09	73.1 ± 8.31	73.02 ± 9.4	*	73.54 ± 10.27	*	72.59 ± 10.16
Explore	0.73 ± 0.00	80.08 ± 7.2	79.17 ± 7.71	*	80.16 ± 6.57	*	79.5 ± 8.69
DEL	1.51 ± 0.14	76.67 ± 6.91	77.07 ± 7.82	54.96 ± 10.68	74.99 ± 8.85	*	74.02 ± 9.68
LVQ	0.73 ± 0.0	80.86 ± 7.02	81.19 ± 7.21	*	81.64 ± 7.04	*	80.79 ± 7.99
MC1	0.73 ± 0.0	79.0 ± 8.03	78.78 ± 6.95	*	79.38 ± 7.2	*	79.11 ± 6.17
RMHC	0.73 ± 0.0	79.03 ± 6.29	78.7 ± 6.69	*	79.62 ± 8.21	*	78.85 ± 7.51

Tabela 3.4. Wyniki klasyfikacji danychdla zbioru Cleveland heart disease

Zbiór danych : Appendictis							
Metoda	$\lvert\mathcal{S}\rvert/\lvert\mathcal{T}\rvert$	k-NN	NRBF	FSM	IncNet	SSV	SVM
$\mathcal{S}=\mathcal{T}$	100.0 ± 0.0	83.00 ± 84.41	86.63 ± 9.56	85.89 ± 9.63	86.97 ± 8.19	86.31 ± 9.98	87.71 ± 8.31
ENN	84.65 ± 1.70	89.03 ± 7.84	85.28 ± 8.58	87.90 ± 8.25	86.93 ± 9.45	85.48 ± 9.35	86.83 ± 10.06
RENN	84.62 ± 1.71	88.35 ± 8.74	85.73 ± 8.83	86.82 ± 8.59	86.58 ± 8.43	84.80 ± 9.54	86.62 ± 9.00
AllKNN	75.38 ± 2.22	87.65 ± 8.54	85.48 ± 7.51	86.22 ± 9.16	86.41 ± 8.82	84.41 ± 10.75	88.15 ± 8.51
ENRBF	92.59 ± 0.81	87.06 ± 8.98	84.97 ± 9.07	87.22 ± 8.59	86.91 ± 8.42	88.71 ± 9.46	86.85 ± 8.94
CNN	39.42 ± 3.09	71.67 ± 13.52	85.02 ± 8.91	75.85 ± 11.28	85.74 ± 9.09	81.75 ± 13.55	80.07 ± 6.28
CA	23.88 ± 2.21	80.50 ± 7.76	82.88 ± 8.34	64.62 ± 16.84	81.60 ± 9.77	71.95 ± 16.18	80.47 ± 6.77
DROP1	5.88 ± 1.83	78.02 ± 22.16	84.16 ± 12.14	72.44 ± 22.15	80.26 ± 18.03	*	81.96 ± 15.31
DROP2	7.24 ± 1.63	83.70 ± 12.02	86.39 ± 8.80	60.22 ± 29.65	83.08 ± 11.88	*	82.54 ± 8.19
DROP3	7.09 ± 1.24	85.95 ± 9.39	86.88 ± 8.79	64.40 ± 24.36	85.91 ± 8.97	*	80.88 ± 8.12
DROP4	7.28 ± 1.37	86.02 ± 10.26	85.27 ± 9.16	66.10 ± 23.70	84.75 ± 10.07	*	80.99 ± 7.05
DROP5	7.59 ± 1.05	87.25 ± 9.06	86.76 ± 10.45	74.95 ± 21.05	85.50 ± 10.18	*	80.55 ± 7.78
EOP	85.85 ± 2.58	79.45 ± 12.37	85.78 ± 8.61	86.38 ± 8.25	85.81 ± 9.10	86.30 ± 10.39	87.50 ± 8.30
ICF	5.47 ± 1.53	87.89 ± 8.63	85.87 ± 8.16	*	83.82 ± 12.81	*	79.40 ± 12.94
GE	43.56 ± 2.97	76.29 ± 12.70	86.18 ± 9.38	67.75 ± 3.23	86.60 ± 7.80	86.54 ± 10.26	84.85 ± 8.63
RNGE	36.95 ± 3.10	78.12 ± 11.78	85.84 ± 9.81	61.73 ± 12.74	85.86 ± 10.24	86.10 ± 10.33	80.38 ± 6.55
IB3	4.53 ± 2.46	79.30 ± 14.11	79.00 ± 15.08	*	79.55 ± 14.73	*	76.39 ± 16.37
ENRBF99	30.89 ± 2.65	87.12 ± 9.15	87.03 ± 9.09	82.00 ± 10.63	86.60 ± 7.77	85.21 ± 9.88	87.27 ± 7.68
ENRBF95	18.21 ± 2.03	86.75 ± 9.52	86.73 ± 9.64	75.29 ± 18.26	85.65 ± 8.96	85.10 ± 12.82	82.82 ± 17.57
ENRBF90	13.30 ± 1.95	84.51 ± 9.83	83.53 ± 12.86	59.07 ± 21.90	84.97 ± 10.24	83.43 ± 14.04	64.02 ± 29.21
ELH	15.33 ± 4.55	87.58 ± 11.85	85.08 ± 9.07	87.12 ± 8.40	83.06 ± 11.37	*	77.34 ± 13.91
ELGrow	1.42 ± 0.62	80.67 ± 8.82	80.11 ± 7.65	*	80.57 ± 7.52	*	81.22 ± 7.79
Explore	1.63 ± 0.58	81.89 ± 8.35	81.57 ± 7.97	*	81.38 ± 8.58	*	82.77 ± 7.79
DEL	3.03 ± 1.52	80.59 ± 12.65	82.68 ± 9.97	*	82.97 ± 10.31	*	82.37 ± 9.82
LVQ	2.1 ± 0.01	85.28 ± 10.40	86.13 ± 12.17	*	85.81 ± 11.26	*	85.94 ± 1.81
MC1	2.1 ± 0.01	86.54 ± 8.62	86.38 ± 10.08	*	84.51 ± 13.11	*	87.34 ± 9.23
RMHC	2.1 ± 0.01	85.12 ± 9.01	87.04 ± 9.15	*	84.22 ± 13.22	*	86.75 ± 9.52

Tabela 3.5. Wyniki klasyfikacji danych dla zbioru danych Appendictis

Zbiór danych : Wisconsin breast cancer							
Metoda	$\lvert\mathcal{S}\rvert/\lvert\mathcal{T}\rvert$	k-NN	NRBF	FSM	IncNet	SSV	SVM
$\mathcal{S}=\mathcal{T}$	100.0 ± 0.0	94.86 ± 4.16	96.29 ± 3.58	96.52 ± 2.07	96.56 ± 2.29	95.63 ± 2.56	96.85 ± 2.09
ENN	96.26 ± 0.35	96.65 ± 2.28	95.78 ± 2.49	95.86 ± 2.36	96.62 ± 2.02	93.73 ± 5.10	96.96 ± 1.94
RENN	95.85 ± 0.39	96.57 ± 2.06	95.84 ± 2.27	95.98 ± 2.32	96.58 ± 1.96	94.87 ± 2.37	96.93 ± 1.89
AllKNN	92.71 ± 0.63	96.80 ± 2.21	95.56 ± 2.09	95.66 ± 2.39	96.59 ± 2.12	94.59 ± 2.67	96.86 ± 1.91
ENRBF	97.69 ± 0.79	95.91 ± 2.03	95.64 ± 2.47	95.65 ± 2.25	96.58 ± 2.17	94.80 ± 2.32	96.74 ± 2.03
CNN	10.92 ± 0.81	91.25 ± 3.25	96.68 ± 2.00	93.77 ± 3.22	95.89 ± 2.40	90.05 ± 4.90	96.56 ± 2.59
DROP1	1.47 ± 0.57	87.56 ± 15.27	90.69 ± 9.70	*	*	*	77.35 ± 12.69
DROP2	3.34 ± 0.79	94.61 ± 2.87	95.98 ± 1.99	84.79 ± 8.18	95.18 ± 3.38	89.18 ± 5.24	95.47 ± 4.06
DROP3	3.00 ± 0.64	96.08 ± 2.35	96.18 ± 2.15	88.35 ± 7.78	96.38 ± 2.43	87.98 ± 8.01	96.16 ± 2.14
DROP4	3.32 ± 0.70	95.75 ± 2.69	96.12 ± 2.23	86.82 ± 9.56	95.78 ± 2.96	88.29 ± 5.35	95.25 ± 4.12
DROP5	4.5 ± 0.90	95.14 ± 2.43	96.05 ± 2.10	93.33 ± 5.11	96.48 ± 2.39	85.12 ± 11.69	96.58 ± 1.93
ICF	2.78 ± 1.06	92.03 ± 3.98	92.41 ± 4.77	74.02 ± 19.46	90.04 ± 9.17	87.71 ± 10.81	84.05 ± 17.87
GE	46.79 ± 3.71	94.93 ± 2.50	95.75 ± 2.44	95.76 ± 2.50	96.35 ± 2.30	92.57 ± 3.13	96.18 ± 2.32
RNGE	10.79 ± 0.71	95.42 ± 2.04	96.58 ± 2.28	94.36 ± 2.92	94.43 ± 6.18	90.71 ± 5.44	95.75 ± 3.18
IB3	4.27 ± 0.67	96.14 ± 2.52	96.76 ± 1.99	94.02 ± 3.10	94.47 ± 3.07	86.79 ± 5.96	95.87 ± 4.68
ENRBF99	15.43 ± 1.25	95.69 ± 2.27	96.74 ± 2.09	95.79 ± 2.25	96.15 ± 2.64	87.79 ± 9.70	96.83 ± 2.08
ENRBF95	8.60 ± 0.83	95.29 ± 2.32	95.85 ± 2.86	92.69 ± 6.85	93.35 ± 5.54	71.84 ± 26.75	81.67 ± 16.82
ELH	3.19 ± 0.66	91.66 ± 4.53	93.84 ± 3.74	86.70 ± 9.36	92.96 ± 5.81	85.06 ± 7.59	76.16 ± 25.61
ELGrow	0.32 ± 0.0	92.16 ± 4.11	92.69 ± 3.48	*	93.95 ± 4.03	*	92.70 ± 3.48
Explore	0.32 ± 0.0	95.13 ± 2.56	95.19 ± 2.43	*	95.50 ± 4.58	*	95.46 ± 2.67
DEL	0.88 ± 0.18	95.02 ± 3.17	95.37 ± 2.73	*	95.86 ± 3.46	*	78.72 ± 14.95
LVQ	0.32 ± 0.0	92.86 ± 2.69	92.81 ± 2.78	*	96.09 ± 2.43	*	94.49 ± 2.74
MC1	0.32 ± 0.0	94.98 ± 2.65	94.71 ± 2.74	*	96.48 ± 1.95	*	96.65 ± 2.33
RMHC	0.32 ± 0.0	96.39 ± 2.05	96.49 ± 2.18	*	95.49 ± 3.17	*	96.22 ± 2.14

Tabela 3.6. Wyniki klasyfikacji danych Wisconsin breast cancer

Zbiór danych : Iris							
Metoda	$\lvert\mathcal{S}\rvert/\lvert\mathcal{T}\rvert$	k-NN	NRBF	FSM	IncNet	SSV	SVM
$\mathcal{S}=\mathcal{T}$	100.0 ± 0.0	95.26 ± 4.92	95.53 ± 5.34	95.80 ± 5.13	95.40 ± 5.19	95.40 ± 5.39	90.40 ± 6.90
ENN	95.28 ± 0.73	95.93 ± 5.23	96.13 ± 4.62	94.80 ± 4.96	95.73 ± 5.17	95.26 ± 5.64	87.93 ± 7.79
RENN	95.21 ± 0.67	95.80 ± 5.25	95.80 ± 4.23	95.40 ± 5.77	95.46 ± 5.10	95.06 ± 5.71	87.86 ± 7.54
AllKNN	94.04 ± 1.24	95.80 ± 5.24	95.20 ± 5.89	95.26 ± 5.11	95.66 ± 5.14	94.53 ± 5.20	86.53 ± 9.40
ENRBF	100.0 ± 0.0	95.73 ± 4.59	95.73 ± 4.37	95.80 ± 5.06	95.06 ± 4.95	94.66 ± 5.66	88.86 ± 7.62
CNN	18.40 ± 2.55	93.60 ± 5.03	95.8 ± 8.19	91.73 ± 4.83	95.80 ± 15.26	80.46 ± 7.13	63.33 ± 5.37
CA	7.8 ± 1.23	94.13 ± 6.03	94.26 ± 5.26	86.86 ± 10.60	95.06 ± 5.20	*	47.40 ± 14.89
DROP1	7.64 ± 1.48	91.26 ± 7.45	91.73 ± 7.36	75.26 ± 13.34	90.86 ± 8.50	*	38.60 ± 11.97
DROP2	10.91 ± 1.61	94.66 ± 5.53	95.06 ± 5.04	86.00 ± 9.90	93.40 ± 5.59	85.06 ± 9.14	37.00 ± 7.93
DROP3	11.33 ± 1.38	95.06 ± 4.93	96.06 ± 4.59	88.53 ± 8.61	92.86 ± 6.47	84.86 ± 8.55	39.33 ± 11.65
DROP4	11.44 ± 1.52	94.53 ± 5.67	95.00 ± 5.03	88.13 ± 9.64	92.20 ± 6.66	85.00 ± 8.61	41.80 ± 14.37
DROP5	11.11 ± 1.44	93.87 ± 5.88	94.33 ± 6.11	88.13 ± 10.38	94.13 ± 5.51	72.60 ± 12.22	40.53 ± 11.34
ICF	9.55 ± 2.29	91.66 ± 10.38	93.13 ± 7.83	78.86 ± 13.32	91.80 ± 8.64	*	42.00 ± 14.09
GE	36.55 ± 1.73	94.80 ± 5.29	95.00 ± 5.23	93.00 ± 6.45	95.00 ± 5.33	94.66 ± 6.32	64.86 ± 11.81
RNGE	18.02 ± 1.35	82.93 ± 15.66	93.13 ± 5.88	81.46 ± 12.55	93.66 ± 7.59	66.33 ± 11.26	35.66 ± 6.42
IB3	19.97 ± 6.19	94.46 ± 5.83	95.13 ± 5.10	93.66 ± 6.71	95.40 ± 5.48	87.46 ± 11.86	58.20 ± 11.89
ENRBF99	71.69 ± 1.33	95.66 ± 4.97	95.80 ± 5.18	95.60 ± 5.51	95.20 ± 5.38	93.86 ± 6.64	90.80 ± 7.24
ENRBF95	20.44 ± 1.49	93.76 ± 6.80	94.93 ± 5.20	94.06 ± 6.04	96.40 ± 4.33	83.66 ± 14.58	64.53 ± 3.90
ELH	8.77 ± 1.94	93.13 ± 6.27	94.20 ± 6.07	85.06 ± 11.55	92.80 ± 5.57	*	48.20 ± 16.50
ELGrow	2.25 ± 0.15	89.80 ± 7.77	88.53 ± 7.82	*	87.93 ± 8.57	*	34.00 ± 2.10
Explore	2.24 ± 0.22	96.06 ± 4.38	95.53 ± 5.25	*	93.00 ± 6.39	*	33.33 ± 0.0
DEL	5.73 ± 1.52	93.40 ± 6.97	94.60 ± 5.81	68.86 ± 16.16	91.40 ± 7.58	*	33.73 ± 1.26
LVQ	2.22 ± 0.0	88.55 ± 1.11	88.33 ± 7.52	*	88.06 ± 7.88	*	33.33 ± 0.0
MC1	2.22 ± 0.0	90.67 ± 8.24	90.60 ± 7.72	*	90.00 ± 7.43	*	33.33 ± 0.0
RMHC	2.22 ± 0.0	91.67 ± 6.97	90.26 ± 7.47	*	89.60 ± 8.16	*	33.33 ± 0.0

Tabela 3.7. Wyniki klasyfikacji dla danych Iris

3.3. Optymalizacja położeń wektorów poprzez LVQ

W poniższych tabelkach zebrane zostały wyniki klasyfikacji dla kilku metod wyboru wektorów referencyjnych w połączeniu z LVQ, który dodatkowo przeprowadzał adaptację położeń wyselekcjonowanych wektorów. LVQ był inicjowany wektorami wybranymi ze zbiory treningowego przez daną metodę selekcji. Współczynnik uczenia wynosił $\alpha = 0.02$ a ilość iteracji $m = 20$.

Zastosowanie optymalizacji położeń wektorów referencyjnych metodą LVQ w niektórych przypadkach okazuje się być bardzo korzystne. Na przykład, klasyfikatory uczone na zbiorach otrzymanych za pomocą ICF, DROP3 i IB3, wykazywały mniejszą poprawność klasyfikacji bez zastosowania optymalizacji LVQ. W pozostałych przypadkach "douczanie" zbioru referencyjnego miało niewielki wpływ na zmianę poprawności klasyfikacji.

Zbiór danych : Wine						
Metoda	k-NN	NRBF	FSM	IncNet	SSV	SVM
ENN+LVQ	94.78 ± 5.01	95.19 ± 4.54	94.92 ± 4.86	97.35 ± 3.50	90.85 ± 6.70	96.47 ± 4.32
ENRBF+LVQ	95.46 ± 4.45	95.58 ± 4.38	95.21 ± 4.43	97.31 ± 3.96	90.74 ± 6.84	97.76 ± 3.71
CNN+LVQ	91.96 ± 6.90	94.78 ± 4.96	88.06 ± 10.27	91.94 ± 7.00	79.45 ± 13.86	88.99 ± 8.24
DROP3+LVQ	94.41 ± 5.12	95.61 ± 4.37	74.35 ± 14.17	89.66 ± 8.05	73.62 ± 13.71	82.10 ± 9.10
ICF+LVQ	93.83 ± 4.92	94.94 ± 4.68	73.66 ± 13.29	93.59 ± 6.63	63.20 ± 17.08	86.70 ± 11.05
RNGE+LVQ	93.88 ± 5.74	93.12 ± 5.79	93.76 ± 6.09	94.11 ± 5.43	86.11 ± 9.08	93.69 ± 5.74
Ib3+LVQ	95.16 ± 4.04	95.61 ± 4.74	95.72 ± 4.44	94.14 ± 6.55	79.60 ± 11.66	93.03 ± 6.04
ELH+LVQ	95.62 ± 4.54	94.70 ± 5.30	63.20 ± 15.76	85.17 ± 9.77	65.70 ± 18.65	72.55 ± 14.50

Tabela 3.8. Wyniki klasyfikacji danych Wine

Zbiór danych : Pima Indian diabetes						
Metoda	k-NN	NRBF	FSM	IncNet	SSV	SVM
ENN+LVQ	74.44 ± 4.59	73.29 ± 4.20	74.40 ± 4.65	75.44 ± 4.34	74.93 ± 4.45	77.08 ± 4.03
ENRBF+LVQ	73.89 ± 5.05	73.68 ± 4.01	74.60 ± 4.63	75.70 ± 4.89	73.45 ± 4.10	77.23 ± 4.28
CNN+LVQ	69.34 ± 4.99	71.68 ± 3.52	69.47 ± 4.97	74.70 ± 4.28	72.42 ± 4.69	76.79 ± 4.38
DROP3+LVQ	70.73 ± 5.14	74.69 ± 4.33	73.06 ± 4.41	75.38 ± 5.01	72.30 ± 5.39	75.81 ± 4.96
ICF+LVQ	65.78 ± 5.93	72.82 ± 5.77	70.46 ± 5.42	74.23 ± 4.82	70.17 ± 6.09	75.14 ± 4.29
RNGE+LVQ	70.54 ± 5.05	73.77 ± 4.21	72.47 ± 5.43	73.57 ± 1.09	73.00 ± 4.85	76.47 ± 4.41
Ib3+LVQ	68.57 ± 6.22	69.13 ± 6.15	69.48 ± 6.23	72.25 ± 6.63	70.35 ± 5.78	71.95 ± 7.32
ELH+LVQ	64.83 ± 6.22	71.88 ± 5.69	69.17 ± 5.74	73.47 ± 5.84	70.64 ± 6.56	75.59 ± 4.82

Tabela 3.9. Wyniki klasyfikacji danych Pima Indian diabetes

Zbiór danych : Czerniak						
Metoda	k-NN	NRBF	FSM	IncNet	SSV	SVM
ENN+LVQ	55.00 ± 3.64	53.84 ± 0.0	72.69 ± 10.16	67.30 ± 7.95	100.00 ± 0.0	73.07 ± 0.0
ENRBF+LVQ	46.15 ± 0.0	50.00 ± 0.0	80.38 ± 5.86	61.92 ± 4.60	100.00 ± 0.0	65.38 ± 0.0
CNN+LVQ	48.84 ± 4.07	53.84 ± 2.56	83.46 ± 17.86	48.84 ± 12.16	100.00 ± 0.0	50.00 ± 0.0
DROP3+LVQ	57.30 ± 8.59	61.92 ± 6.13	63.46 ± 5.51	47.30 ± 9.60	100.00 ± 0.0	35.76 ± 3.64
ICF+LVQ	69.61 ± 2.18	70.00 ± 3.03	89.23 ± 10.38	51.92 ± 7.08	100.00 ± 0.0	35.38 ± 1.62
RNGE+LVQ	52.31 ± 2.89	52.69 ± 0.0	97.69 ± 4.13	68.07 ± 9.07	100.00 ± 0.0	50.00 ± 0.0
Ib3+LVQ	55.00 ± 7.26	52.30 ± 11.05	79.61 ± 13.81	45.38 ± 17.18	100.00 ± 0.0	40.38 ± 5.51
ELH+LVQ	54.61 ± 3.97	49.23 ± 4.36	67.30 ± 10.45	43.07 ± 10.99	100.00 ± 0.0	45.00 ± 3.64

Tabela 3.10. Wyniki klasyfikacji danych Czerniak

Zbiór danych : Cleveland heart disease						
Metoda	k-NN	NRBF	FSM	IncNet	SSV	SVM
ENN+LVQ	79.07 ± 7.04	78.78 ± 6.70	79.10 ± 6.80	80.32 ± 6.78	77.47 ± 7.74	80.48 ± 7.40
ENRBF+LVQ	78.92 ± 6.61	79.89 ± 6.51	80.09 ± 7.05	81.67 ± 7.14	79.00 ± 7.73	81.91 ± 6.80
CNN+LVQ	75.22 ± 7.27	72.42 ± 8.31	74.82 ± 7.48	79.80 ± 7.19	71.75 ± 7.91	80.58 ± 7.11
DROP3+LVQ	77.55 ± 6.76	79.03 ± 7.83	77.58 ± 6.68	81.29 ± 6.82	71.29 ± 8.73	80.68 ± 6.78
ICF+LVQ	77.43 ± 8.12	79.53 ± 6.11	76.20 ± 6.89	79.54 ± 7.81	69.04 ± 8.82	79.59 ± 6.98
RNGE+LVQ	78.74 ± 6.62	72.51 ± 7.93	76.87 ± 8.44	80.83 ± 6.04	72.62 ± 7.23	81.94 ± 6.89
Ib3+LVQ	78.78 ± 7.28	78.77 ± 7.56	77.17 ± 7.66	79.84 ± 7.33	72.64 ± 8.26	80.75 ± 1.62
ELH+LVQ	74.84 ± 8.32	76.12 ± 7.95	73.37 ± 8.31	78.06 ± 7.73	70.25 ± 9.35	80.10 ± 6.62

Tabela 3.11. Wyniki klasyfikacji danych Cleveland heart disease

Zbiór danych : Appendictis						
Metoda	k-NN	NRBF	FSM	IncNet	SSV	SVM
ENN+LVQ	87.60 ± 9.36	88.12 ± 8.57	86.83 ± 8.74	86.94 ± 9.76	85.18 ± 10.82	86.78 ± 8.90
ENRBF+LVQ	86.38 ± 9.17	87.42 ± 10.28	86.57 ± 8.55	87.20 ± 7.40	88.11 ± 8.89	86.90 ± 9.83
CNN+LVQ	86.00 ± 10.08	84.54 ± 9.48	80.91 ± 10.52	85.85 ± 8.74	82.65 ± 10.59	80.17 ± 6.35
DROP3+LVQ	84.00 ± 9.82	84.39 ± 12.31	86.52 ± 10.45	86.75 ± 10.57	86.61 ± 10.37	81.75 ± 11.63
ICF+LVQ	86.93 ± 9.66	86.60 ± 9.46	*	88.10 ± 2.11	*	81.40 ± 9.02
RNGE+LVQ	85.86 ± 9.94	84.92 ± 8.52	78.79 ± 12.03	85.79 ± 10.25	85.95 ± 10.32	80.36 ± 6.33
Ib3+LVQ	80.89 ± 11.31	82.35 ± 9.58	*	80.98 ± 13.87	*	79.03 ± 13.54
ELH+LVQ	75.90 ± 13.53	79.38 ± 12.35	80.56 ± 14.17	83.38 ± 12.26	78.90 ± 15.80	79.50 ± 11.07

Tabela 3.12. Wyniki klasyfikacji danych dla zbioru danych Appendictis

Zbiór danych : Wisconsin breast cancer						
Metoda	k-NN	NRBF	FSM	IncNet	SSV	SVM
ENN+LVQ	96.72 ± 2.07	96.66 ± 1.89	96.06 ± 2.61	96.58 ± 1.93	93.02 ± 7.85	96.84 ± 1.97
ENRBF+LVQ	96.52 ± 2.24	96.37 ± 0.23	95.93 ± 2.26	96.53 ± 2.09	94.94 ± 2.11	96.69 ± 2.34
CNN+LVQ	96.10 ± 2.66	94.86 ± 1.77	95.99 ± 2.16	96.07 ± 2.06	91.19 ± 4.86	96.82 ± 2.10
DROP3+LVQ	95.03 ± 2.14	96.16 ± 2.14	93.70 ± 3.41	95.07 ± 2.37	90.49 ± 3.52	96.78 ± 0.51
ICF+LVQ	95.89 ± 2.67	96.02 ± 2.60	93.21 ± 4.00	94.52 ± 4.62	86.82 ± 11.99	95.57 ± 2.60
RNGE+LVQ	95.80 ± 2.69	94.32 ± 2.90	94.16 ± 3.11	95.64 ± 3.47	91.91 ± 4.16	96.92 ± 2.18
Ib3+LVQ	96.72 ± 2.47	96.73 ± 2.08	93.84 ± 4.22	95.74 ± 1.34	88.67 ± 5.31	96.96 ± 0.19
ELH+LVQ	95.61 ± 2.18	95.36 ± 2.67	92.66 ± 3.75	95.50 ± 2.54	83.73 ± 7.53	94.26 ± 7.30

Tabela 3.13. Wyniki klasyfikacji danych Wisconsin breast cancer

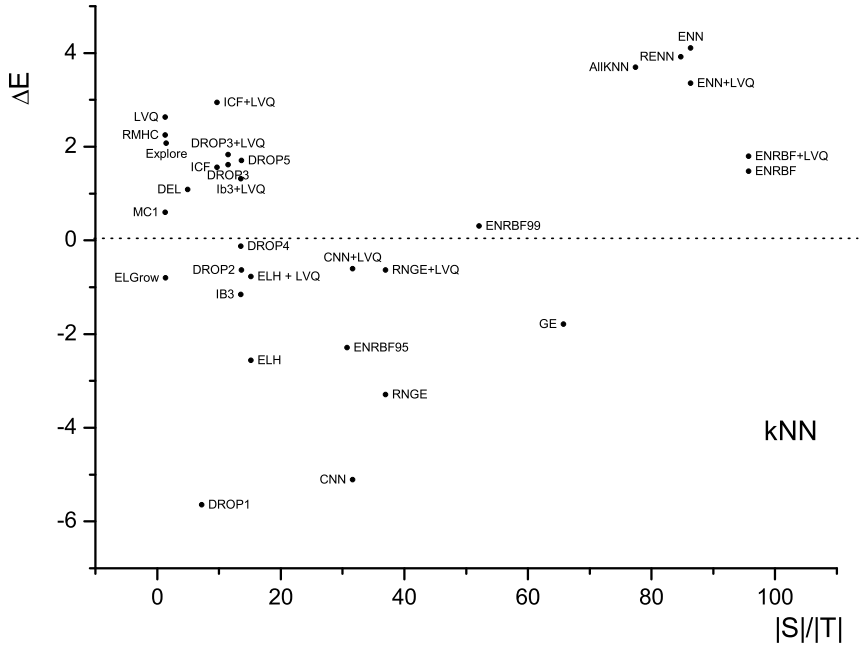
Zbiór danych : Iris						
Metoda	k-NN	NRBF	FSM	IncNet	SSV	SVM
ENN+LVQ	95.80 ± 4.74	95.80 ± 5.21	95.60 ± 5.78	95.73 ± 5.33	95.00 ± 5.17	88.00 ± 7.92
ENRBF+LVQ	95.2 ± 5.17	95.66 ± 5.40	95.33 ± 5.34	95.20 ± 4.90	94.53 ± 5.57	88.93 ± 8.51
CNN+LVQ	88.20 ± 11.67	95.53 ± 5.07	88.53 ± 10.98	94.33 ± 6.39	80.80 ± 15.48	64.86 ± 4.69
DROP3+LVQ	93.66 ± 5.86	94.80 ± 5.01	90.40 ± 9.79	95.40 ± 4.92	85.86 ± 8.53	41.80 ± 11.61
ICF+LVQ	91.06 ± 10.49	90.26 ± 9.39	85.73 ± 23.67	91.40 ± 12.04	79.13 ± 13.47	44.00 ± 14.76
RNGE+LVQ	78.33 ± 15.01	91.40 ± 5.59	89.26 ± 10.26	93.06 ± 7.84	65.40 ± 9.62	33.80 ± 1.47
Ib3+LVQ	94.06 ± 5.86	93.66 ± 5.29	92.33 ± 7.91	94.86 ± 5.16	88.06 ± 9.88	62.80 ± 10.13
ELH+LVQ	93.06 ± 5.91	94.33 ± 5.81	83.13 ± 13.70	94.46 ± 5.74	78.46 ± 12.71	45.80 ± 14.55

Tabela 3.14. Wyniki klasyfikacji dla danych Iris

3.4. Dalsze wnioski

Chcąc unaocznić wpływ opisanych metod działanie poszczególych klasyfikatorów, otrzymane wyniki, dla wszystkich zastosowanych zbiorów danych, zostały uśrednione i przedstawione na odpowiednim wykresie. Oś pozioma odpowiada średniemu poziomowi redukcji liczby wektorów (wyrażonemu w procentach) dla wszystkich zastosowanych zbiorów danych, zaś oś pionowa wyznacza średnie odchylenie (dla użytych zbiorów danych) poprawności klasyfikacji ΔE po zastosowaniu jednej z metod redukcji od poprawności uzyskanej w przypadku zastosowania w procesie uczenia oryginalnego zbioru treningowego. Położenie danej metody na wykresie określa jej przydatność dla danego klasyfikatora. Metody położone powyżej linii przerywanej ($\Delta E > 0$) to metody dla których poprawność klasyfikacji wzrosła po selekcji wektorów. Metody po lewej generują najmniejsze zbiory referencyjne, po prawej - dają najmniejszą redukcję.

Dla k-NN (rys. 3.3) największy spadek poprawności (około 5%) klasyfikacji dają DROP1 i

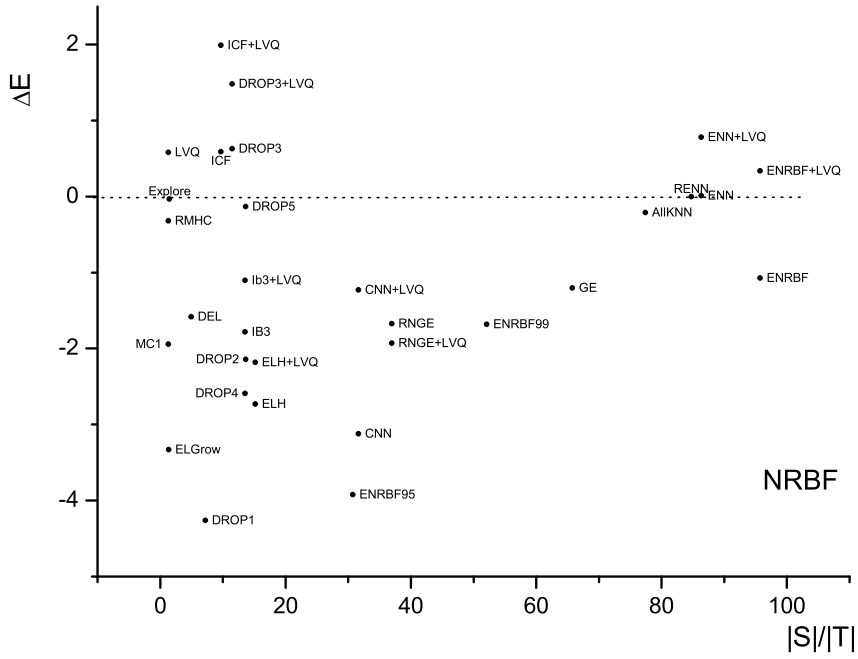


Rysunek 3.3. Średnia zmiana poprawności ΔE klasyfikacji kNN po redukcji ilości wektorów w zbiorze treningowym ($\Delta E = 0$ odpowiada dokładności 79.98%)

CNN. Metody nazwane filtrami szumu spełniają swoje zadanie, zmniejszając zazwyczaj ilość mylnie zaklasyfikowanych przypadków. Zadziwiająco dużo metod o znacznym stopniu redukcji umiejscowiło się powyżej $\Delta E = 0$. Zastąpienie całej klasy jednym przypadkiem (LVQ, MC1, RMHC) dało w tym wypadku bardzo dobre wyniki. Optymalizacja położeń za pomocą LVQ podniosła poprawność klasyfikacji we wszystkich (poza ENN) przypadkach.

W przypadku NRBF (rys. 3.4) również wystąpiło sporo metod, które wraz z dużą selekcją, dawały niewielkie straty dokładności klasyfikacji. DROP3 i ICF redukując objętość zbioru treningowego poniżej 20%, podniosły poprawność klasyfikacji. Reszta metod wypadła słabiej, jednak żaden z modeli nie spowodował większego niż 5% spadku dokładności klasyfikacji.

W przypadku FSM (rys. 3.5) można zauważyć tendencję spadku poprawności klasyfikacji dla coraz mniejszych zbiorów wektorów użytych do uczenia. IB3, RNGE i CNN sa jedynymi metodami, które dając redukcje poniżej 50% zbioru oryginalnego, nie spowodowały spadku dokładności



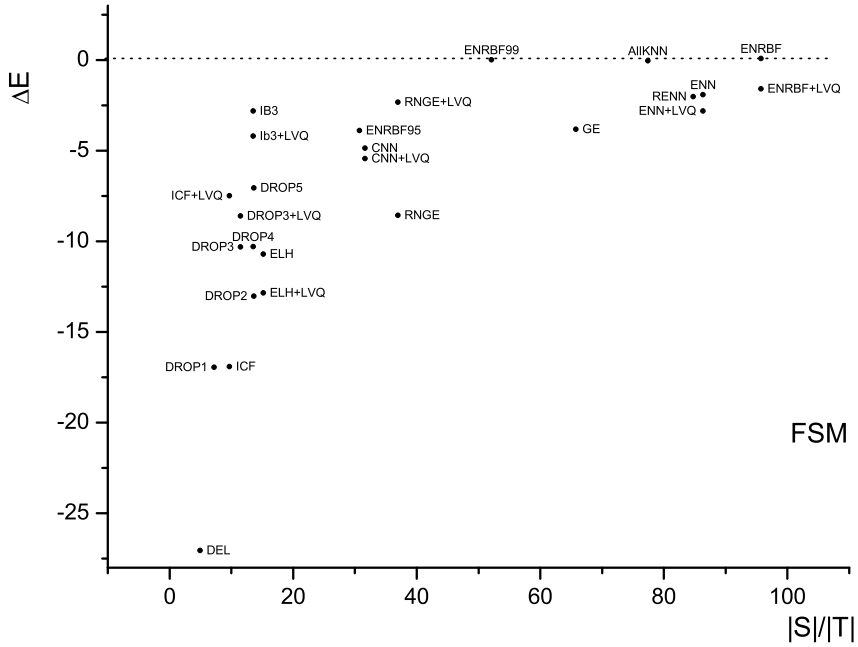
Rysunek 3.4. Średnia zmiana poprawności ΔE klasyfikacji NRBF po redukcji ilości wektorów w zbiorze treningowym ($\Delta E = 0$ odpowiada dokładności 82.32%)

klasyfikacji większego niż 5%. Dobre wyniki osiągnął ENRBF99 (zachowując średnio połowę wektorów, bez straty dokładności). Optymalizacja położeń za pomocą LVQ nie przyniosła w tym przypadku poprawy działania, oprócz DROP3 i ICF.

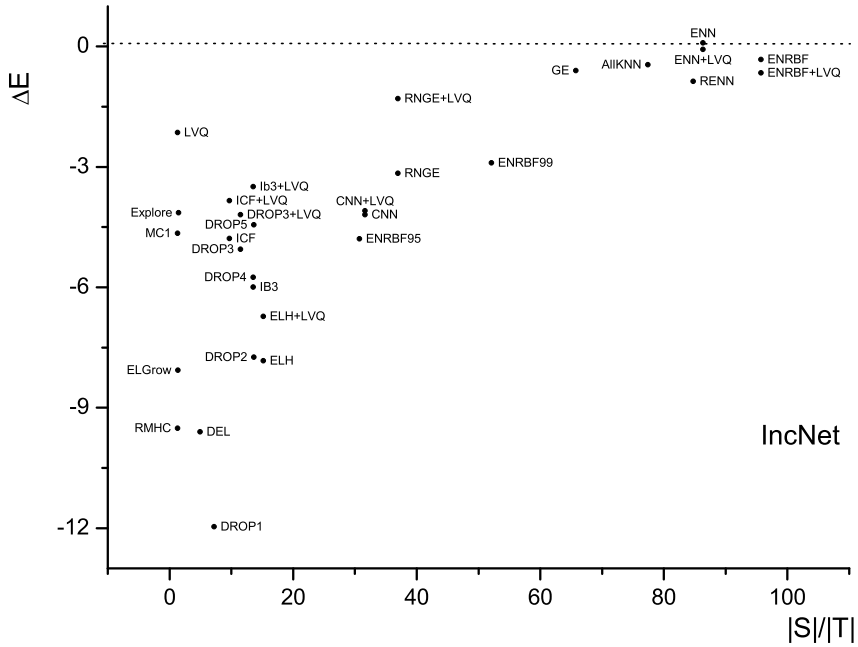
IncNet (rys.3.6) jest również czuły na ilość wektorów uczących. Pomoc ze strony LVQ poprawia sytuację niektórych metod ale tylko niewielka ich część nie spowodowała spadku dokładności mniejszego niż 5%. Zadziwiająco dobrze wypadł LVQ, dopasowujący po jednym wektorze dla klasy, powodując średnio spadek o 2.14% liczby poprawnie klasyfikowanych przypadków.

Dla drzewa decyzyjnego SSV (rys. 3.7) metody redukujące ilość wektorów uczących poniżej 50% spowodowały znaczny spadek poprawności klasyfikacji. Z całej grupy takich metod najlepszy okazał się IB3+LVQ, zmniejszając poprawność klasyfikacji o 4.09%.

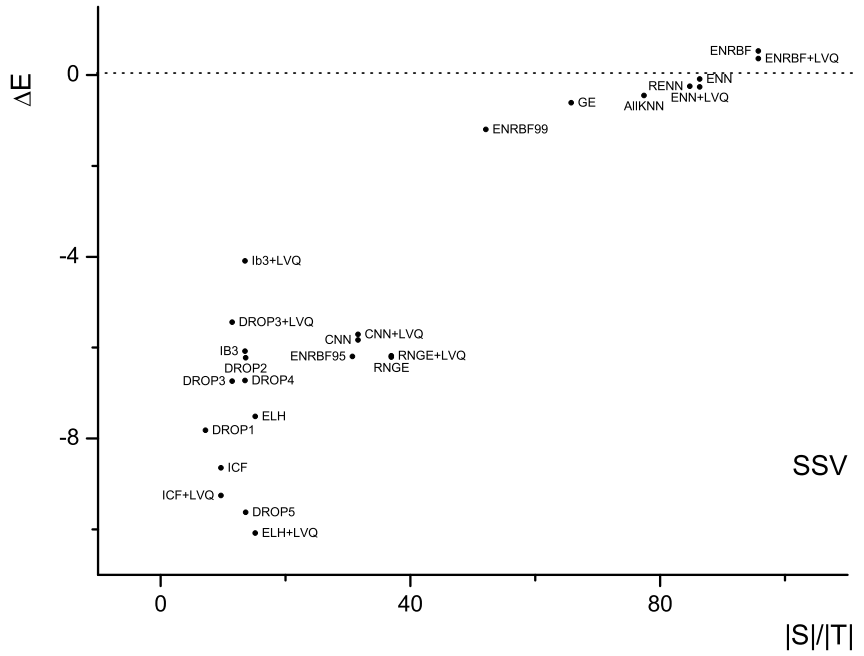
W przypadku SVM (rys. 3.8) selekcja wektorów okazała się najmniej korzystna. Można zauważyć proporcjonalny spadek dokładności klasyfikacji wraz ze zmniejszaniem rozmiaru zbioru uczącego. Co prawda, optymalizacja położeń poprawia trochę sytuację, jednak dla większości metod spadek liczby poprawnie klasyfikowanych przypadków jest większy od 10%.



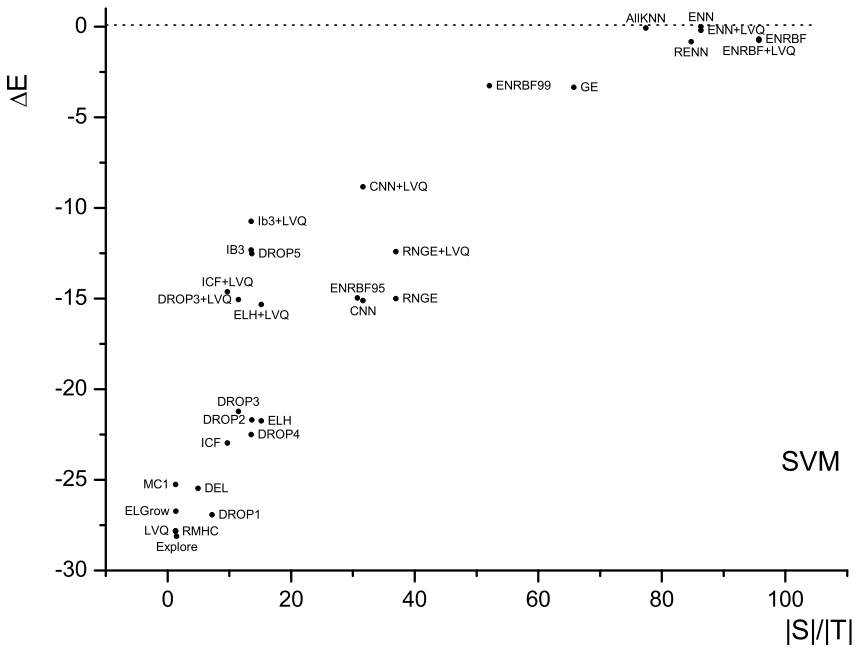
Rysunek 3.5. Średnia zmiana poprawności ΔE klasyfikacji FSM po redukcji ilości wektorów w zbiorze treningowym ($\Delta E = 0$ odpowiada dokładności 88.47%)



Rysunek 3.6. Średnia zmiana poprawności ΔE klasyfikacji IncNet po redukcji ilości wektorów w zbiorze treningowym ($\Delta E = 0$ odpowiada dokładności 85.74%)



Rysunek 3.7. Średnia zmiana poprawności ΔE klasyfikacji SSV po redukcji ilości wektorów w zbiorze treningowym ($\Delta E = 0$ odpowiada dokładności 88.32%)



Rysunek 3.8. Średnia zmiana poprawności ΔE klasyfikacji SVM po redukcji ilości wektorów w zbiorze treningowym ($\Delta E = 0$ odpowiada dokładności 85.73%)

Podsumowanie

W pracy poruszone zostało zagadnienie wyboru wektorów referencyjnych dla modeli klasyfikujących. Szereg zaprezentowanych metod daje ogląd stosowanych podejść wywodzących się z dziedziny rozpoznawania wzorców oraz uczenia maszynowego. Skuteczność działania stosowanych metod uzależniona jest w dużej mierze od problemu przed jakim stawiamy dany model. Podejście, dające dobre wyniki w przypadku niektórych zbiorów danych treningowych, może się nie sprawdzić w dla innych zbiorów. Część prezentowanych metod stosowana może być wyłącznie dla klasyfikatorów k-NN i NRBF. Wyróżnić należy algorytmy GA oraz ENRBF2, wybierające zbiory dobrze reprezentujące przypadki treningowe dla wszystkich zastosowanych metod klasyfikacji. Modele ICF oraz DROP3 dają dobre rezultaty dla klasyfikatorów k-NN i NRBF przy stosunkowo dużej redukcji ilości wektorów, zwłaszcza jeśli poprawimy położenie otrzymanych portotypów metodą LVQ. Zauwarzalny jest pozytywny wpływ na jakość klasyfkacji zastosowania optymalizacji położeń wektorów referencyjnych przy pomocy LVQ. Interesującym faktem jest duża skuteczność metod, które ograniczają rozmiar zbioru referencyjnego do bardzo małej liczby przypadków. Wyróżnić należy tu metody Explore, LVQ1 i ICF. Niekiedy jeden wektor reprezentujący całą klasę okazuje się wystarczająco dobrym źródłem wiedzy dla klasyfikatora, głównie dla modeli k-NN lub NRBF. Małe zbiory prtotypowe ułatwiają ekstrakcję reguł i zależności występujących w strukturze danych, które możemy odnieść do całego zbioru treningowego.

Dodatek: Dane użyte w eksperymentach

Appendictis Database
106 wektorów w 2 klasach (85 przypadki ostrego zapalenia wyrostka, 21 reszta)
8 cech o wartościach ciągłych
Pochodzenie danych: Shalom Weiss[1]

Wisconsin Breast Cancer Databases
699 wektorów, 2 klasy (458 przypadków złośliwych raka i 241 przypadków łagodnych)
9 cech o dyskretnych wartościach
Pochodzenie danych: UCI Machine Learning Repository [2]

Czerniak
Zbiór treningowy:
250 wektorów, 4 klasy (przypadków łagodnych 62, odmiana błękitna 64, podejrzenie choroby 62, rak złośliwy 62)
14 cech (2 o wartościach ciągłych, 12 binarnych)
Zbiór testowy:
26 wektorów (odpowiednio 6, 7, 6, i 7 wektorów)
Pochodzenie danych: Wojewódzki Szpital Specjalistyczny w Rzeszowie

Cleveland Heart Disease Databases
303 wektorów, 5 klas (164 zdrowi, oraz cztery stopnie zaawansowania choroby 55, 36, 35, 13)
13 cech (4 ciągłych, 9 nominalnych)
6 wartości brakujących
Pochodzenie danych: UCI Machine Learning Repository

Pima Indians Diabetes Database
768 wektorów, 2 klasy (500 klasa 1, 268 klasa 2)
8 cech o wartościach dyskretnych
Pochodzenie danych: UCI Machine Learning Repository

Wine Recognition Database
178 wektorów w 3 klasach (odpowiednio 59 , 71 i 48 wektorów)
13 ciągłych cech
Pochodzenie danych: UCI Machine Learning Repository

Iris Plant Database
250 wektorów, 3 klasy (Setosa 50, Virginica 50, Versicolor 50)
4 cechy o wartościach ciągłych
Pochodzenie danych: UCI Machine Learning Repository

[1] http://www.phys.uni.torun.pl/kmk/projects/datasets.html
[2] http://www.ics.uci.edu/~mlearn/MLRepository.html

Bibliografia

[1] Bimay K. Bhattacharya, Godfried T. Toussaint Ronald S. Poulsen. Application of proximity graphs to editing nearest neighbor decision rules.

[2] Henry Brighton, Chris Mellish. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, 6:153—172, 2002.

[3] Richard O. Duda, Peter E. Hart, David G. Stork. *Pattern Classification.* John Wiley Sons, Inc., wydanie drugie, 1997.

[4] Krzysztof Grąbczewski, Włodzisław Duch. The separability of split value. *5th Conference on Neural Networks and Soft Computing, Zakopane*, s. 201–208, 2000.

[5] Simon Haykin. *Neural Networks.* Macmillan Publishing Company, New York, 1994.

[6] N. Jankowski. Data regularization. L. Rutkowski, R. Tadeusiewicz, redaktorzy, *Neural Networks and Soft Computing*, s. 209–214, Zakopane, Poland, 2000.

[7] Norbert Jankowski. Approximation and classification in medicine with incnet neural networks. *Machine Learning and Applications. Workshop on Machine Learning in Medical Applications*, s. 53–58, 1999.

[8] Norbert Jankowski, Włodzisław Duch. Ontogeniczne sieci neuronowe. 1999.

[9] Timothy Masters. *Sieci neuronowe w praktyce.* Wydawnictwo Naukowo-Techniczne, Warszawa, 1996.

[10] Ramon A. Mollineda, Francesc J. Ferri, Enrique Vidal. Marge-based prototype selection for nearest neighbor classification. *Proceedings of 4th World Multiconference on Systemics, Cybernetics and Informatics*, 7:640–645, 2000.

[11] Bernard Schölkopf, Alexsander J. Smola. *Learning with Kernels.* The MIT Press, 2002.

[12] David B. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. *International Conference on Machine Learning*, s. 293–301, 1994.

[13] D. Randall Wilson, Tony R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.